Chapter 9
Computational Argumentation for Supporting Learning Processes: Applications and Challenges

Carlos Chesñevar, César A. Collazos, and Ana Maguitman

Abstract This book chapter analyzes different applications and challenges of computational argumentation for modeling different aspects of learning processes. Some of the topics included are argument-based recommender systems for educational purposes; argument-based shared knowledge for computer-supported collaborative learning (CSCL) and argument-based opinion mining for eliciting students' knowledge based on information items corresponding to different topics of study. We also identify and discuss salient challenges associated with argumentation in the current state of the art. The chapter is organized to be self-contained, including an overview of the key elements in computational argumentation. Our contribution is intended to provide a reference point for researchers working on intelligent techniques for educational processes who are interested in incorporating argumentation as a metaphor for modeling intelligent decision making in Intelligent Tutoring Systems (ITS), Computer-Supported Collaborative Learning (CSCL) systems, and other related areas.

Keywords Argumentation · Intelligent tutoring systems · Computer-supported collaborative learning · Opinion mining · Recommender systems · Shared knowledge awareness · Learning systems

C. Chesñevar (✉) · A. Maguitman
Departamento de Ciencias E Ingeniería de La Computación (DCIC UNS), Instituto de Ciencias E Ingeniería de La Computación (ICIC CONICET UNS), Universidad Nacional del Sur – Campus Palihue, San Andrés, 800 - 8000 Bahía Blanca, Argentina
e-mail: cic@cs.uns.edu.ar

A. Maguitman
e-mail: agm@cs.uns.edu.ar

C. A. Collazos
Universidad del Cauca - Popayán, Popayán, Colombia
e-mail: collazos@unicauca.edu.co

M. Ivanović et al. (eds.), *Handbook on Intelligent Techniques in the Educational Process*, Learning and Analytics in Intelligent Systems 29,
https://doi.org/10.1007/978-3-031-04662-9_9

9.1 Introduction

Computational argumentation is a discipline that has been gaining increasing importance and wider audiences over the last decades, mainly as a vehicle for facilitating rationally justifiable decision making when handling incomplete and potentially inconsistent information. Argumentation provides a sound model for dialectical reasoning, which underlies discussions or opinion confrontation in social networks. In Collaborative systems, argumentation is an important aspect to help problem-solving situations, considering the cognitive processes of critical information checking, argument elaboration and the taking of multiple perspectives. Argumentation systems are increasingly being considered for applications in developing software engineering tools, constituting an important component of multi-agent systems for negotiation, problem solving, shared understanding, and the fusion of data and knowledge. Such systems implement a dialectical reasoning process by determining whether a proposition follows from certain assumptions, analyzing whether some of those assumptions can be disproved by other assumptions in our premises. In this way, an argumentation system provides valuable help to analyze which assumptions from our knowledge base are giving rise to inconsistency and which assumptions are harmless.

This chapter is structured as follows. In Sect. 9.2 we summarize the main elements which characterize computational models of argument (such as argument, counter-argument, defeat, and the notion of warranted conclusion). We will introduce the basics of Defeasible Logic Programming [1], which will be used for subsequent examples. Then, in Sect. 9.3 we focus on argument-based recommender systems, a sub-area that has received particular attention in the last years. We discuss potential applications of these recommenders for educational purposes. Section 9.4 discusses an alternative approach to argumentation based on opinion mining. We show that this particular view of argumentation processes can help enhance learning processes by identifying reasons pro and con in a very intuitive way. Section 9.5 considers the notion of shared knowledge awareness in the context of argumentation. We show how multiple knowledge bases (associated with different students) can be suitably integrated for collaborative problem solving. Finally, Sect. 9.6 presents the conclusions and discusses some avenues for future research.

9.2 Argumentation in a Nutshell

Argumentation is an important aspect of human decision making. In many situations of everyday life, when faced with new information, people need to ponder its consequences, in particular when attempting to understand problems and come to a decision. Argumentation systems [1–4] are increasingly being considered for applications in developing software engineering tools, constituting an important component of multi-agent systems for negotiation, problem solving, and the fusion of data and

knowledge. Such systems implement a dialectical reasoning process by determining whether a proposition follows from certain assumptions, analyzing whether some of those assumptions can be disproved by other assumptions in our premises. In this way, an argumentation system provides valuable help to analyze which assumptions from our knowledge base give rise to inconsistencies and which assumptions are harmless.

In defeasible argumentation, an argument is a tentative (defeasible) proof for reaching a conclusion. 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 partial order "$\preccurlyeq$"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 preferred (A $\succcurlyeq$ B and A $\preccurlyeq$ B) or that A and B are not comparable with each other. For the sake of example, let us analyze the following example about real-world knowledge on spiders. Consider the following sentences:

(1) *If something looks dead, it is usually dead;*
(2) *If something moves when touched, it is usually not dead;*
(3) *If a spider is dead, it is usually not dangerous.*
(4) *If something is a spider, it is usually dangerous.*
(5) Black widow is a spider.
(6) Black widow moves when touched.
(7) Black widow looks dead.

Sentences in italics correspond to defeasible rules (rules which are subject to possible exceptions). Statements (5), (6), and (7) correspond to facts (strict information). Note that different arguments can be constructed:

1. Argument A (based on rules 4 & fact 5): Black widow is a spider. Spiders are usually dangerous. Therefore, black widow is dangerous.
2. Argument B (based on rule 1,3 and facts 5,7): Black widow is a spider. Black widow looks dead. If something looks dead, it is usually dead. If a spider is dead, it is usually not dangerous. Therefore, Black widow is not dangerous.
3. Argument C (based on rule 2, fact 6). Black widow moves when touched. If something moves when touched, it is usually not dead. Therefore Black widow is not dead.

In this particular situation, different 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 [1]. 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

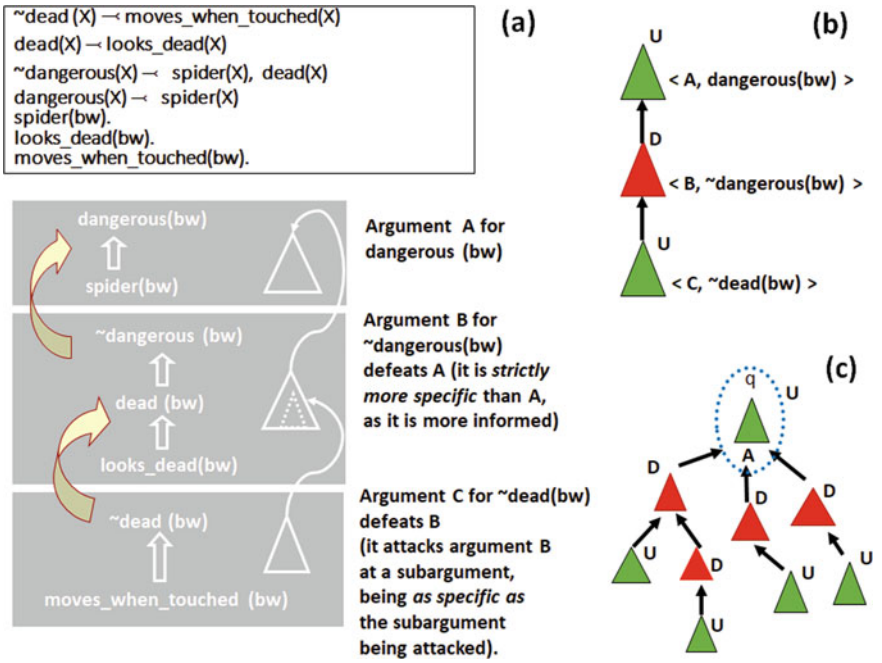

Fig. 9.1 **a** DeLP program for the spider example. **b** The original argument is deemed as "accepted". **c** The dialectical analysis might be complex, resulting in a dialectical tree with several nodes representing arguments and attack relationships among them

a second argument (a defeater), which in turn can be defeated by a third argument, reinstating the first one. As a given argument might have many defeaters, the above situation results in a tree-like structure, rooted in the first argument at issue, where every argument in a branch (except the root) defeats its parent (see Fig. 9.1c).

Defeasible Logic Programming: implementing argumentation as a programming language

Defeasible Logic Programming (DeLP) [1] is a logic-based programming language for modeling incomplete knowledge and providing argument-based inference.[1] It has been applied in different contexts, such as multi-agent reasoning [5], recommender systems [6–9], among others. As most computational argumentation systems, DeLP relies on two kinds of knowledge: *strict* and *defeasible* knowledge. Strict knowledge corresponds to the knowledge that is certain, such as *facts* about the world or mathematical truths (e.g. "all men are mortal"). The strict knowledge is *consistent*, i.e. no contradictory conclusions can be derived from it. On the other hand, defeasible knowledge corresponds to that knowledge which is *tentative*, modeled through "rules with exceptions" (*defeasible* rules) of the form **"if P then usually Q"** (e.g., "if

[1] For an in-depth treatment of DeLP and its features the reader is referred to [1].

something is a bird, it usually flies"). Such rules model our incomplete knowledge about the world, as they can have exceptions (e.g., a penguin, a dead bird, etc.). Syntactically, a special symbol ($-\!\prec$) is used to distinguish "defeasible" rules from logical implications ($\leftarrow$).

Argumentation systems like DeLP allow the user to define a knowledge base involving strict and defeasible knowledge. An *argument A* for a claim *c* is basically some "tentative proof" (a derivation using a non-empty set of *defeasible* information) for concluding *c* from the knowledge base (DeLP program). Arguments must additionally satisfy the requirement of *consistency* (an argument cannot include contradictory propositions) and *minimality* (by not including repeated or unnecessary information). Conflicting arguments may emerge in DeLP: an argument *A attacks* another argument *B* whenever both of them cannot be accepted at the same time, as that would lead to contradictory conclusions. Arguments are on their turn compared with each other using a modular criterion (typically specificity), so that it can be established when an argument *defeats* another.

Note that the notion of defeat among arguments may lead to complex "cascade" situations: an argument A may be defeated by an argument B, which in turn may be defeated by an argument C, and so on. Besides, every argument involved may have on its turn more than one defeater. Argumentation systems allow us to determine when a given argument is considered as *ultimately acceptable* with respect to the knowledge we have available by means of a *dialectical analysis*, which takes the form of a tree-like structure called *dialectical tree*. The root of the tree is a given argument A supporting some claim, and children nodes for the root are those defeaters B_1, B_2, … B_k for A. The process is repeated recursively on every defeater B_i, until all possible arguments have been considered. Leaves are arguments without defeaters. Some additional restrictions apply (e.g. the same argument cannot be used twice in a path, as that would be fallacious and would lead to infinite paths).

Figure 9.1a illustrates how a DeLP program for the spider example can be formulated. Note that the symbol "~" stands for strict negation (thus, ~dead(X) means "X is not dead"). In this sample DeLP code "bw" stands for "black widow". The DeLP programming language allows to make queries such as "dangerous(bw)" (standing for "is black widow dangerous?"), which prompts the computation of an *argument* supporting the query. The argument A is found (since bw is a spider, it should be considered dangerous by default). Additionally, a defeater argument B is found which attacks A (black widow is not dangerous as it looks dead), which is on its turn defeated by a third argument C (black window moves when touched, and therefore it is not dead!). All this dialectical process is carried out automatically by the DeLP inference engine (associated arguments can also be displayed using a GUI interface). The associated dialectical analysis is shown in Fig. 9.1b. Arguments with no successful attacks are deemed as ultimately accepted (e.g. argument C). An inner argument is deemed as ultimately accepted if all its attackers are not accepted; otherwise, the inner argument is defeated. Complex situations might arise (e.g. Fig. 9.1c), which are solved by the DeLP inference engine.

9.3 Argument-Based Recommendation in Learning Environments

The Internet is one of the main sources of information and resources for students to explore or learn practically any topic. However, identifying the most useful information or resources can be a difficult task for a student. One of the main difficulties is that there is an overwhelming amount of potentially useful material for learning nearly any topic. Another difficulty lies in the fact that students might not be able to pose appropriate queries to search for relevant content as they may not be entirely familiar with the topic being or to be learned.

Recommender systems can alleviate these problems by providing meaningful recommendations to students. Recommendation in learning environments can be exploited from different perspectives. One approach consists in identifying and suggesting learning objects (e.g., documents, videos, instructional games, etc.) for a specific learning objective. Learning objects are characterized by metadata such as educational resource type, interactivity type and level, content, description, language, and format. When confronted with a problem requiring procedural knowledge (i.e. a sequence of steps to be carried out to solve a task), recommender systems can play a useful role by providing suggestions and hints (e.g. by pointing out possible alternatives or by issuing a warning when a wrong decision has been made). Also, recommenders can be useful during the knowledge acquisition process itself, by engaging learners in specific activities that promote declarative knowledge construction through the exploration of both domain-specific and domain-general knowledge.

A recommender system can adopt a task modeling approach, a user modeling approach, or a combination of both. A task is a piece of work required to achieve an objective. Tasks are usually associated with the need to access information to solve problems, evaluate content, construct meaning, create knowledge and make decisions. A task-based recommender system that supports a student learning process typically monitors the student's work, analyzes its content, seeks for similar content or other students that completed similar tasks, generates recommendations, and incrementally refines the recommendations based on the student's progress on the task at hand and the student's reaction to the suggested resources. Task representations need to be continuously updated as students change their focus during learning activities. This can be captured by analyzing a variety of contextual interaction patterns resulting from clicks, dwell time, cursor movement, scrolling, etc. A learning resource that proved to be useful for a learning task is likely to be useful for a similar task. Hence being able to model tasks and determine when two tasks are similar is key to develop a task-based recommender. A learning task can be modeled by the student's log activity, documents being read or edited, web pages being visited, milestone tasks being accomplished, among other items [10]. Task representations can be stored in a repository and associated with different kinds of resources (learning objects, procedural knowledge, and domain knowledge) that proved to be useful during those particular tasks.

Different from task models, which are changeable, user models are more persistent. Students can be modeled by their declared interest, their long-term browsing history, capabilities, social network communities, and social media interactions, among other features. In [11] various aspects are considered to model the student profile, such as learning style, educational level, preferred language, preferred topic, and preferred format. By modeling students, it is possible to compute similarity scores among students, and hence to recommend items to a target student based on how useful those items proved to be in the past to students with a similar profile.

The most common variants of recommender systems are content-based [12] and collaborative filtering [13]. Content-based recommender systems rely on a representation of a user or an item to find items that match with the user's recommendation needs. For instance, a content-based recommendation for a student currently learning a specific topic in biology requires representing the specific topic or the knowledge the student currently has or seeks to have about the topic to identify material similar to these representations. On the other hand, collaborative filtering algorithms rely on past user's behavior to find other users with similar behavior. The basic idea of a collaborative filtering approach is to provide item recommendations based on those items that were useful to or were liked by similar users. For instance, by modeling a student's skill, it may be possible to identify other students with similar skills to recommend material that proved to be useful to those similar students in the past.

Most existing recommenders are based on machine learning and information retrieval algorithms. As indicated in the literature [6, 7], these approaches are unable to effectively provide informed explanations of the reasons behind a given recommendation. Also, these approaches do not naturally support the kind of analysis of actions and interactions that are crucial in any learning process.

Incorporating Argumentation to Recommend Learning Resources

Argument-based recommenders can be applied to overcome some of the limitations of traditional recommendation systems in learning environments. Content- and collaborative-based recommender systems that use task-modeling or user-modeling approaches can be enhanced by incorporating argumentation technologies, to provide reasoned recommendations and facilitate the exploration of relevant learning resources through a dialectic process. Since the students will receive both a recommendation and a reason supporting it, they will have more confidence in the presented results and they can give the system explicit feedback that can help guide the recommendation process.

The widespread availability of learning resources repositories, coming from different sources and accessed by students with mixed backgrounds, perspectives, and learning abilities offers new opportunities to create argument-based recommendation services. These services can take advantage of the diverse community of students accessing the stored learning resources to implement collaborative-based recommenders. Developing an algorithm for recommending learning resources is challenging because it requires combining many, sometimes conflicting aspects. For instance, a resource may be useful for learning a physics topic for a student with a good mathematics background but it may not be useful for someone who has not

developed a good background in mathematics yet. In light of the defeasible nature of students' information needs in learning environments, argumentation is an attractive technology to explore and revise potential recommendations, by generating suggestions of learning resources based on items that proved to be useful in the past during similar tasks and adapting them to the target student.

During procedural knowledge acquisition, students typically create and test solutions in shared learning environments and discuss their potential solutions with teachers and other students while carrying out learning tasks. The creation and discussion could be naturally integrated with a collective dialectic process that provides a context to let learners actively explore different ideas and positions. An argument-based recommender that guides this process will foster the generation of ideas and debate.

Another natural way in which argumentation can help enhance the recommendation of general and domain-specific knowledge is by involving the student in an argumentation process, either with the recommendation system or with other students. During declarative knowledge acquisition, an argument-based recommender will advise the student on which areas to cover, to increase the effectiveness of the learning process. It can also guide students in the process of engaging in specific discourse activities, to express their viewpoints and also to react to other students' perspectives.

Figure 9.2 provides a general picture of how argumentation can be integrated into the recommendation process in a learning environment. Based on a student's current task, similar stored past tasks can be retrieved. As discussed above, stored past tasks will typically be associated with a variety of learning resources, which may include learning objects of different types (e.g., manuals, videos, instructional games, etc.), as well as with procedural and declarative knowledge that proved to be useful for the

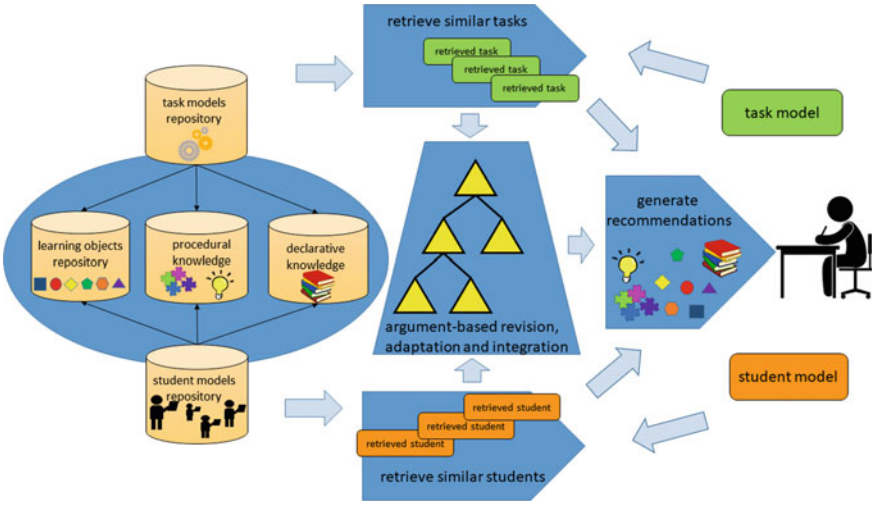

Fig. 9.2 Argument-based recommender systems in the context of educational processes from a high-level perspective

associated tasks in the past. Also, based on the student profile it is possible to retrieve the profiles of other similar students and those learning resources that proved to be useful to those students in the past. Finally, an argument-based approach can be taken to revise, adapt and integrate information coming from similar students and tasks, resulting in recommendations of potentially useful learning resources. We present next a case study illustrating how an argument-based recommendation approach based on Defeasible Logic Programming can be applied in a learning scenario.

A Case Study: Using Defeasible Logic Programming to Model Recommendations about Students' Learning Resources

The process for generating recommendations of learning resources by an argument-based recommender is different from the process adopted by most of the existing recommenders. However, they share the requirement of having access to prior knowledge about a collection of students, tasks, and learning resources, which can be codified as facts of a DeLP program, as illustrated in Fig. 9.3. Facts provide information about the students, tasks, and resources being modeled. Also, rules can be defined to determine if two students or two tasks are similar. To define such rules, similarities between students and tasks can be calculated by applying probabilistic latent semantic analysis [14] or matrix factorization [15], among other techniques. Finally, the DeLP program will contain a set of postulates that describe the conditions under which a learning resource should be recommended to a given student. For instance, a resource is typically recommended to a student if the student likes the resource type. However, even if the student likes the resource type, the resource will not be recommended if there is evidence that the resource was not useful to a similar student in the past. On the other hand, a resource will be recommended if it was useful for a task similar to the current one, albeit it was not useful to a similar student. An additional level of specificity that distinguishes between tasks for which a student finds a resource useful or not could be added if this information is available. This way, the argumentative process will deal with general facts and more specific facts that may be in conflict.

As discussed in Sect. 9.2, rules in a DeLP program are combined to support or reject a conclusion by building arguments. Figure 9.4 shows the arguments that have been computed to determine whether resource r_2 should be recommended to *Peter* while he is completing task t_1. In this example, the root argument of the dialectical tree is *recommend(peter, t1,r2)*, which turns out to be defeated and hence we have no reason to believe that *Peter* will benefit from resource r_2 while completing task t_1.

As another example, assume that the system is evaluating whether resource r_3 should be recommended to *Peter* while he is completing task t_1. Figure 9.5 presents a dialectical tree illustrating how arguments can be built in favor of such a recommendation. The root argument of the dialectical tree is *recommend(peter, t1, r3).* Although there is a second argument that attacks the root argument, the second argument is in turn defeated by a third argument, concluding that the recommendation under analysis should be made.

Facts about resources and their types, whether the resources were useful for students or tasks and whether a resource type is liked or disliked by a student

resource_type(r1, video)
resource_type(r2, manual)
resource_type(r3, instructional_game)
useful_for_task(t1, r1)
useful_for_task(t1, r2)
useful_for_task(t1, r3)
useful_for_student(sam, r1)
useful_for_student(sam, r2)
~useful_for_student(sam, r3)
likes_resource_type(peter, video)
likes_resource_type(peter, instructional_game)
dislikes_resource_type(peter, manual)

Strict rules determining whether two students or two tasks are similar

similar_student(S1, S2) ← *[Computed elsewhere]*

similar_task(T1, T2) ← *[Computed elsewhere]*

Defeasible rules (commonsense knowledge) defining the cases for which resource R should be recommended to student S during task T

recommend(S, T, R) ≺ *resource_type(R, RT), likes_resource_type(S, RT)*

~recommend(S,T,R) ≺ *resource_type(R, RT),likes_resource_type(S, RT), similar_student(S1, S2), ~useful_for_student(S2 ,R)*

recommend(S,T,R) ≺ *resource_type(R,RT), likes_resource_type(S,RT), similar_student(S,S1), ~useful_for_student(S1,R),similar_task(T,T1), useful_for_task(T1,R)*

Fig. 9.3 A sample DeLP program for modeling recommendations about resources for students

9.4 Opinion Mining and Argumentation: Contrasting Opinions and Viewpoints on the Internet

Opinion mining refers to a number of different techniques (including datamining, sentiment analysis, etc.) which are used in text analysis for automatically identifying opinion and emotion. Opinion mining is a very recent research area, and it provides a powerful resource for educational processes, as it allows students to better understand concepts and ideas which might be associated with different viewpoints.

Argumentation and opinion mining can be combined into an interesting approach presented in [16] which results in **argument-based opinion mining**. In contrast with other logical approaches to argumentation, an argument A for a conclusion C is essentially a set of *statements* that provide reasons to support C. These statements can correspond to different information items available on the Internet (contents from

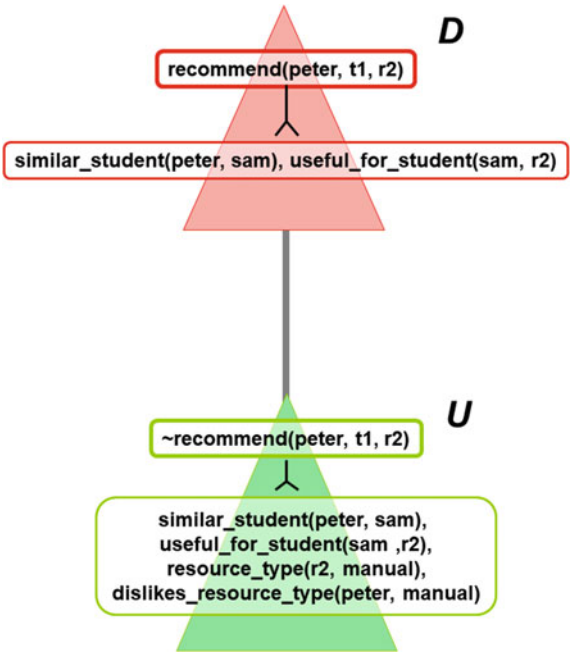

Fig. 9.4 A sample dialectical tree associated with the query "recommend(peter,t1,r2)", where the root argument is deemed as defeated

reviews, tweets in Twitter, etc.). For the sake of example, we will refer to tweets in what follows in order to present the associated framework [17]. We will take a sample topic to illustrate how argument-based opinion mining works. Consider for example the issue "abortion". Some tweets on that topic could be as follows:

Tweet_1 = "government should ban #abortion, it means killing babies"

Tweet_2 = "#abortion is debatable, not all cases are to be equally considered"

Tweet_3 = "#abortion is a right every woman has. Defend it"

Tweet_4 = …

We will refer to the set of topics or issues at hand as the **query Q** to be associated with a given argument (e.g. Q = "abortion" or Q = "abortion, Argentina"). In addition to the notion of query, we will introduce the idea of **context** or **criterion C.** This concept is intended to identify particular properties or features that we would like to consider associated with the query Q. We will aggregate these two elements when defining arguments, and hence will write **(Q,C).** Thus, for example, C1 could be a criterion that indicates that only tweets posted between timestamp T1 and T2 are to be selected. Then (Q,C1) will select only those tweets that contain all the terms of query Q and have been posted in the time period [T1,T2]. Other examples of criteria that can be naturally applied are, for instance, requiring that those tweets were retweeted more than n times, requiring that every user that posted tweets T has at least m followers, etc.

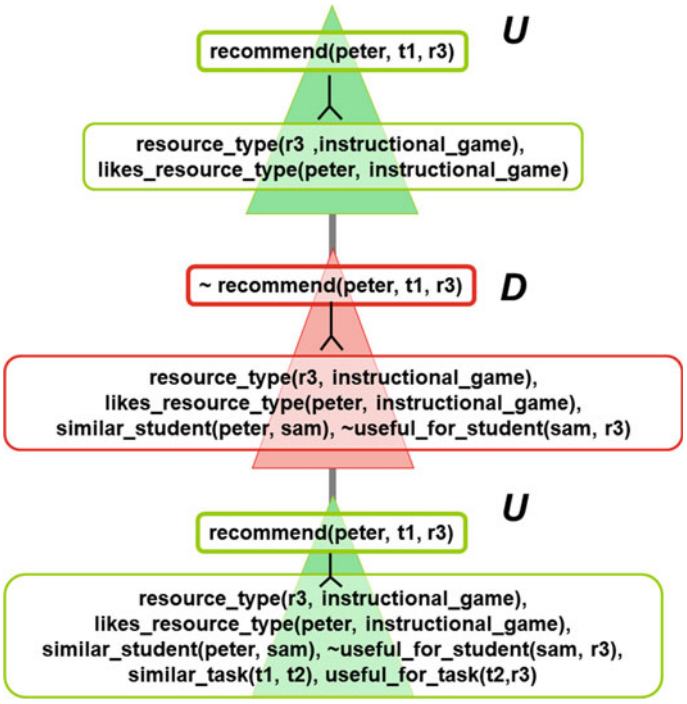

Fig. 9.5 A sample dialectical tree associated with the query "recommend(peter,t1,r3)", where the root argument is deemed as undefeated

Finally, we will also assume a set S of possible **sentiments.** A possible range for S could be *positive*, *negative* and *neutral*.[2] For the sake of example, Tweet_1 could be considered as a negative tweet towards abortion, whereas Tweet_3 corresponds to a positive tweet on that topic. We will generalize the notion of sentiment associated with a single tweet to the notion of *prevailing sentiment* in a bunch of tweets (i.e., the sentiment that prevails, according to some criterion, e.g. percentage). In the same way, we will assume that sentiments might convey conflicting feelings or emotions (e.g. anger vs. happiness; boredom vs. excitement, positive vs. negative, etc.). We will abstract away which is the prevailing sentiment as well as existing conflicts through mapping functions *Sent* and *Conflict*, respectively. Thus, Sent(T) will determine which is the sentiment value associated with a tweet T (as a singleton). As stated before, we will extend the intended meaning of Sent to an arbitrary set of tweets T = {t1,t2….tk}, where every ti denotes a tweet, so that Sent(T) denotes the prevailing sentiment associated with T (e.g. most tweets in T are positive, and hence we deem T as "positive").

[2] This approach is used in some commercial platforms for assessing tweets in terms of a positive, negative or neutral value and the percentage of tweets corresponding to each value (e.g. sentiment 140.com).

Two sentiments Sent1 and Sent2 in *Sent* will be "in conflict" whenever Sent1 differs from Sent2. (e.g. positive will be in conflict with negative; neutral will be in conflict with negative). According to this, we can say that a set of tweets T1 is in conflict with a set of tweets T2 whenever Sent(T1) differs from Sent(T2). We further assume that all possible conflicts are "equally preferred" in the sense that a conflict between positive and negative is as strong as a conflict between positive and neutral; the underlying idea is to identify the situation that the prevailing sentiments in both sets of tweets are not the same.

Characterizing an Argument as a Set of Tweets. Arguments in Conflict and Opinion Trees

For the sake of example, let us assume that we have a set T of 20,000 tweets associated with the query "abortion", and the context is given by "Argentina" and "years 2018–2020" (e.g. we consider only tweets originated from Argentinean accounts posted in the period 2018–2020). Note that in many cases we can easily identify a query because it was used as a hashtag (e.g. #abortion) within a thread of tweets.

In our approach, an **argument** A based on opinion mining for a query Q under a criterion C is a set of tweets associated with (Q,C) with a prevailing sentiment Sent. Thus, following the previous example, for a query Q = "abortion" and a criterion C corresponding to "all tweets in the period 2018–2020", and assuming that the possible sentiments S = {pos, neg, neutral}, then the argument A for Q under C would be the subset of all tweets related to "abortion" restricted to the period 2018–2020. Assuming that e.g. 80% of the tweets have a negative connotation, then the prevailing sentiment Sent = neg.

We have shown how to express arguments for particular queries under a certain criterion, associated with a given prevailing sentiment. Such arguments might be *attacked* by other arguments, which on their turn might be attacked, too. In argumentation theory [3], this leads to the notion of *dialectical analysis*, which can be associated with a tree-like structure in which arguments, counter-arguments, counter-counter-arguments, and so on, are taken into account. The central idea underlying the exploration of possible attacks for a given argument is given by the notion of *specificity*.

Suppose that an argument supporting the query Q = "abortion" is obtained, with a prevailing negative sentiment. If the original query Q is extended in some way into a new query Q' that is more specific than Q (i.e. Q' = Q ∪ {w}, for some particular word w), it could be the case that the argument supporting Q' would have a different (possibly conflicting) prevailing sentiment. For example, more specific opinions about abortion are related to other topics, like for example ethics, social problems or programs, religious issues, etc. To explore all possible relationships associated with arguments returned for a specified query Q and criteria C, we can define a high-level algorithm to construct an **opinion tree** recursively as follows[3]:

[3] The full-fledged description of the algorithm can be found in [16].

Algorithm BuildOpinionTree

Input: query Q, criterion C

Output: Opinion Tree OT(Q) rooted in an argument A for Q under criterion C with prevailing sentiment Sent.

1. We start with an argument A obtained from the original query Q under a criterion C with a prevailing sentiment S, which will be the root of the tree.
2. Next, we analyze within the tweets in A all relevant words that might be used to "extend" Q, by adding a new element (w) to the query, obtaining Q' = Q ∪ {w}.
3. Then, a new argument for Q' under criterion C with prevailing sentiment S' is obtained, which will be associated with a subtree rooted in the original argument A (i.e., the tree resulting from BuildOpinionTree(Q',C)).

It is also easy to see that for any query Q, the algorithm BuildOpinionTree finishes in finite time: given that a tweet may not contain more than 280 characters, the number of contained descriptors is finite, and therefore the algorithm will eventually stop, providing an opinion tree as an output (Fig. 9.6).

A Case Study: The Abortion Issue

As a case study to illustrate our approach, we consider the abortion issue based on information from Twitter in December 2012, when the Michigan legislature was debating several regulations on abortion practices. Consider the query Q = "abortion", and a criterion C = {tweets posted less than 48 h ago}. A root argument is computed for Q and C, obtaining an associated prevailing sentiment (negative). It

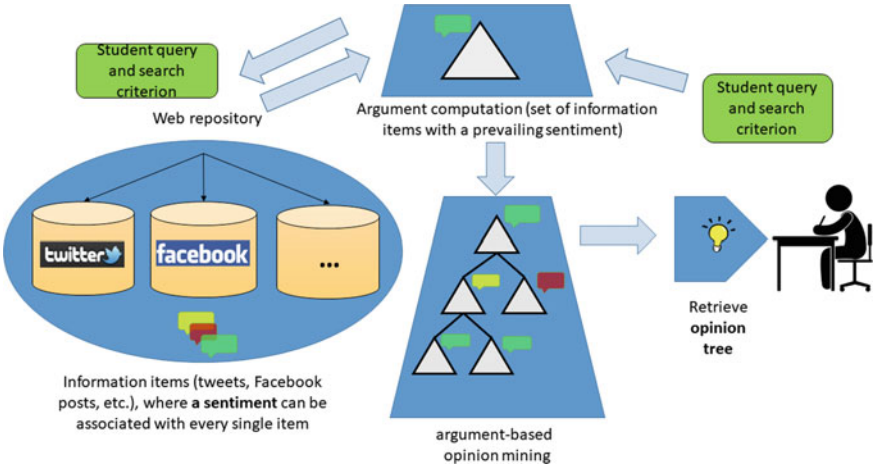

Fig. 9.6 Schematic overview of argument-based opinion mining. Based on a student query and a given context, an argument is computed along with possible conflicting arguments. The whole opinion tree is retrieved as an answer

should be remarked that the algorithm for building opinion trees avoids the repetition of any new descriptor used to extend the query associated with a node. The construction is performed depth-first, so that new descriptors are gradually introduced using a technique specifically designed to guide term selection (outside the scope of this paper, for a detailed description see [16].

Figure 9.7 illustrates how the construction of an opinion tree for the query Q = "abortion" looks like. Distinguished symbols (+, −, =) are used to denote positive, negative and neutral sentiments, respectively. Note that the original query Q has cardinality 1, and further levels in the opinion tree refer to incrementally augmented queries (e.g. {"abortion", "michigan"}, or {"abortion", "murder"}). Leaves correspond to arguments associated with a query Q' which cannot be further expanded, as the associated number of tweets is too small for any possible query Q' U {w}, for some w. Furthermore, we can identify some subtrees in the Opinion Tree rooted in "abortion" which consist of nodes having all the same sentiment. In other words, further expanding a query into more complex queries does not change the prevailing sentiment associated with the root node. In other cases, expanding some queries results in a sentiment change (e.g. from "abortion" into {"abortion", "option"} or {"abortion", "wish"}).

Integrating opinion trees into the learning process allows students to analyze public debate in a more systematic way while at the same time encourages social awareness and an interest in current affairs. Opinion trees help students synthesize complex information and analyze a specific topic from different perspectives. This approach helps improve logical and critical thinking.

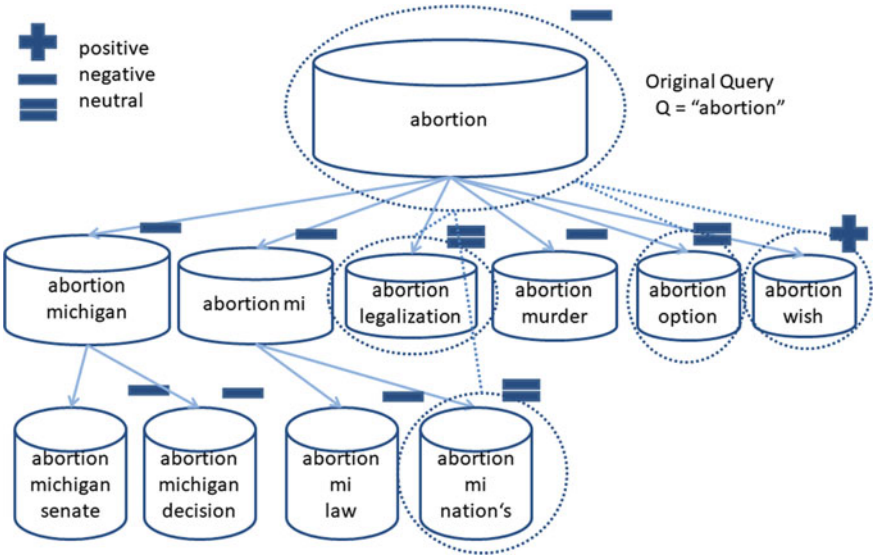

Fig. 9.7 An Opinion Tree for The Abortion Issue (computed from Twitter, 2012). Adapted from [18]

9.5 Shared Knowledge Awareness and Argumentation

Shared Knowledge (SK) concerns the common knowledge constructed by a student group when carrying out a collaborative learning activity in a CSCL environment.[4] In this setting, Shared Knowledge Awareness (SKA) has been defined as the consciousness on the SK that this student group has when performing a specific collaborative task in a restricted moment of time [19, 20]. Indeed, the construction of SK is strongly related to the acquisition of an appropriate level of SKA, as being aware of any knowledge (in particular SK) implies learning something about it.

Students' acquisition of SKA in CSCL scenarios is not a simple task, and a number of questions that should be considered to reach it have been proposed [19]. However, it is difficult to ascertain how to provide mechanisms to model the construction of SKA in a real CSCL system. Indeed, this problem is related to different features, in particular with characterizing the students' dialectical reasoning underlying negotiation processes when looking for an agreement or consensus about a given claim.

In this section, we will illustrate how computational argumentation can contribute when modeling educational processes where different *knowledge sources* (associated with capabilities or domain knowledge corresponding to different students) can be integrated following an argumentative approach. We will consider DeLP as the underlying programming language to provide a support tool for dialectical discussions in a CSCL framework. Indeed, our framework will allow modeling the dialectical analysis carried out by participants in CSCL scenarios, helping them to identify the emerging SK and the explicit specification of its associated SKA. As a starting point, we will consider the individual knowledge constructed by different students when performing a collaborative task (probably expressed in natural language and stored in a generic CSCL platform).

We depart from the assumption that the knowledge required for solving the collaborative task is complex, so that students should be able to integrate different perspectives and conflicting opinions about the task to be solved. Our goal is that participating students can make use of the reasoning and visualization capabilities provided by the argumentation system in order to support part of their SK construction as well as making explicit its associated SKA. Figure 9.8 illustrates the process of acquiring shared knowledge awareness through argumentation. As a result of this process, students will be able to identify what we will call *Argument-Based Shared Knowledge* (ArgSK): students are aware of how different conflicting pieces of knowledge are related to each other, why some of such pieces should be deemed as warranted (and some others should not), and how their own individual knowledge may be in conflict with other participants' knowledge.

A Case Study: Solving a Printer Configuration Problem Collaboratively[5]

Consider the following case study: Computer Science students from three different universities U_1, U_2 and U_3 (located in different cities) have to solve an activity

[4] CSCL stands for "Computer Supported Collaborative Learning".

[5] This example was originally presented in [20].

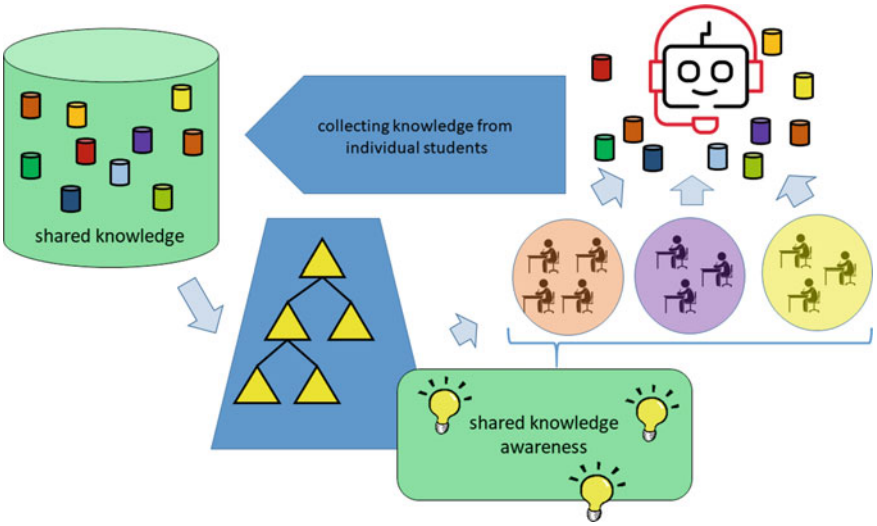

Fig. 9.8 Schematic overview of argument-based shared knowledge awareness. Based on knowledge from individual students an intelligent agent builds shared knowledge, which is combined with a dialectic process to acquire shared knowledge awareness

collaboratively in a CSCL scenario. The activity is structured using the *JIGSAW* technique [21] and includes the task T of *detecting good and bad features in different configurations of a personal computer model called "pcu" (acronym for "PC for universities")*, which is the computer model available in the computer labs of the three universities (e.g. the three labs have pcus with the same configuration, devices, etc.). The students are divided into small groups of three people, each of them belonging to a different university. Following the JIGSAW technique, each member of the group will be responsible for analyzing a different piece of knowledge when constructing his/her individual knowledge. Let us focus on one jigsaw group G formed by three students, namely S_1, S_2 and S_3. As stated before, we will assume that S_1, S_2 and S_3 are using a particular CSCL system to solve T, as they are located in different cities. For the sake of example, the students must learn about different topics related to pcus as follows: (1) S_1 is assigned the topic "input/output devices"; (2) $S_2$is assigned the topic "memory devices"; and (3) S_3 is assigned the topic "processors".

Let us assume that students have already constructed their individual knowledge and they are coming back to the group G to solve T. At this moment, S_1, S_2 and S_3 have to present a well-organized report to other members of G about the topic each of them has studied. The immediate goal is to construct SK and SKA in order to solve T. As a part of their SK and SKA, students are offered to re-cast their knowledge using an argumentative formalism (such as DeLP), where arguments and their acceptance statuses can be computed automatically.

Following our proposal, an automated argumentation platform is integrated with the CSCL scenario. It includes a knowledge base K (empty at the beginning), an

inference engine for computing arguments and a suitable front-end for posing queries and visualizing results. First, each student S_i exchanges (separately) messages with an intelligent agent about what he/she knows, and the intelligent agent writes down this in terms of rules and facts. Following our example, suppose that student S_1 has acquired knowledge about printers (as they are I/O devices). He/she has learned the following: *hp1020 and hp1018 are models of laser printers. Laser printers work ok if the computer has a good RAM memory. Inkjet printers usually work ok with any kind of computer*. Besides, S_1 has checked the computer model "pcu" (the object of study) and has seen that there was a printer connected, namely the hp1020. In the same way, S_2 has studied memory devices. He/she has learned that *a RAM memory of 256* Mb *or more is usually good enough for a computer, unless you want to use it with a laser printer, since in such a case a RAM of 256* KB *has slow access, which is usually not a good feature.* In addition, S_2 has checked the computer model "pcu" and has seen that the computer had 256 Mb of RAM memory (note that S_2does not know anything about processors or printers, he just knows that they appeared as related concepts when learning about memory devices). Concerning S_3, he/she has individual knowledge about processors. He/she has learned that *if a computer has a processor double-core, then the processor is usually fast. Pentium processors result in slow access time for RAM memory. An exception are Pentiums with the special swap technology, which do not have this problem.* He/she has checked the computer model "pcu" and has seen that it has a Pentium processor with "swap technology".[6]

At the end of all the dialogues between S_1, S_2 and S_3 with the intelligent agent, the knowledge base K stores the sum of the three students' individual knowledge, which could have been written down by the intelligent agent as follows:

Facts about the computer in the lab

```
printer(pcu, hp1020)        % fact from student S1
has_ram (pcu,256)           % fact from student S2
processor (pcu,pentium)     % fact from student S3
```

Defeasible rules (Commonsense knowledge)—C stands for an arbitrary computer

Knowledge about printers coming from S_1

[6] Names and values used here are fictitious. They are just considered for the sake of the example and not necessarily according to a real-world situation.

printer(C, laser) $\prec$ *printer(C, hp1020)*

printer(C, laser) $\prec$ *printer(C, hp1018)*

printer_ok(C)$\prec$ *ram_memory(C,good), printer(C, laser)*

printer_ok(C)$\prec$ *printer(C, inkjet)*

Knowledge about RAM memories coming from S_2

ram_memory (C, good)$\prec$ *has_ram (C,X), X* $>=$ *256*

ram_slow_access(C) $\prec$ *has_ram (C,X), X* $=$ *256, printer(X,laser)*

~ram_memory (C,good) $\prec$ *ram_slow_access (C)*

Knowledge about processors coming from S_3

processor(C, fast) $\prec$ *processor(C,double_core)*

ram_slow_access (C)$\prec$ *processor(C,pentium)*

~ram_slow_access (C) $\prec$ *processor(C,pentium), has_processor(pentium,swap_tech)*

Now, consider that as part of task T to solve (detecting good and bad features in different configurations of a "pcu"), S_1, S_2 and S_3 are discussing about the piece of knowledge "*printer_ok(pc)*", which stands for the claim *"is it ok to have a printer connected to the computer pcu?"*. By analyzing the individual knowledge provided by each S_i separately, the members of G cannot infer anything (except from the facts provided). However, if they *jointly* consider all the information stored in K (which accounts for part of their SK) they can rely on DeLP to automatically compute a dialectical tree rooted in the above claim, which will include all possible combinations of arguments and defeaters related to the claim. This way, they can guarantee that those pieces of knowledge subject to dialectical discussions will be part of the SK only if they are warranted on the basis of the joint knowledge of the group, thus avoiding a dialectical discussion based on incomplete and biased perceptions of reality. Hence, if the claim results to be supported by a warranted argument, then the above piece of knowledge can be part of the argument-based shared knowledge (ArgSK) for the group on the basis of rational and justified information.

In this particular example, S_1, S_2 and S_3 will obtain a warranted argument supporting the claim "*printer_ok(pcu)*" (as the warranted argument *Arg1* supports it), and they will add the claim to their ArgSK. Note that the claim is deemed as warranted by the underlying argumentation system, based on the dialectical tree shown in Fig. 9.3. Besides, S_1, S_2 and S_3 will visualize the dialectical tree shown in Fig. 9.3 (left), which explicitates a rational justification of the obtained results. Indeed, it can be seen that there exists an argument *Arg1* supporting *"printer_ok(pcu)"*, which can be obtained by combining knowledge from $S_1$and S_2. The argument is based on knowing that "pcu" has enough RAM memory to support hp1020, the laser printer connected to it. However, *Arg1* is defeated by *Arg2*, which supports the claim

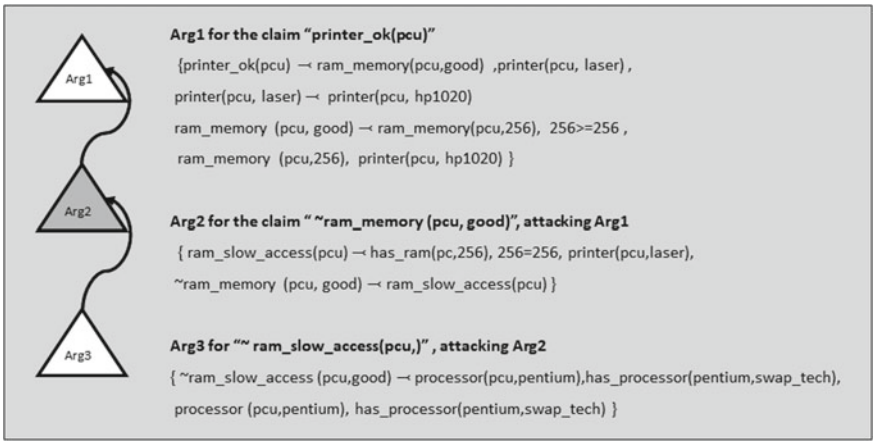

Fig. 9.9 Outline of the dialectical analysis obtained for the claim "is it ok to have a printer connected to the computer pcu?". Left: Arg1 is warranted and ultimately prevails, as it is defeated by Arg2, which is on its turn defeated by Arg3. Right: the argument contents provided by the argumentation engine in DeLP

"~ram_memory (pcu, good)" (the student who studied memory devices provided a defeasible rule which states that 256 Mb usually do not suffice for a laser printer to run ok). But this argument *Arg2* is on its turn defeated by *Arg3* standing for "*~ram_slow_access(pcu)*" (the student who studied processors provided a defeasible rule which states that computers with Pentium processors with swap technology, as it is the case here, do not have problems with RAM of 256 Mb). This way, the visualization of the tree will be linked to the ArgSKA associated with the claim under consideration, helping S_1, S_2 and S_3 to be aware of their own SK. Later on, S_1, S_2 and S_3 will be able to use the piece of warranted knowledge (the fact that the printer connected will work ok) when going further on the resolution of T (Fig. 9.9).

9.6 Conclusions and Related Work

In this chapter we have analyzed the role of computational argumentation as a metaphor for handling incomplete and potentially contradictory information. The task of contrasting alternative arguments and determining which ones are to be ultimately accepted is core to many educational processes in which critical thinking is involved. Even though argumentation has been central to education for many centuries in Western civilization, it has not been until recently that more evolved computational models for argumentation have been developed. We contend that these models can provide effective alternatives for new conceptualizations that improve and empower analytical thinking for both students and teachers.

In the last years, argumentation has had considerable growth and consolidation, establishing itself as a discipline in its own right within the research community in Artificial Intelligence. Argument-based recommender systems (one of the applications discussed in this chapter) have received particular attention when context-based information is taken into account [22–24]. Another important aspect that needs to be considered in the learning environment is the notion of trust. Students or instructors may trust certain learning resources because other users trusted by them recommend those resources, or simply because they trust the resources' sources. Trust is subjective, not always symmetric or transitive, context dependent, dynamic, and defeasible. Hence, as discussed in [25], trust can be naturally modeled using an argumentative framework, playing a useful role at the moment of integrating the notion of trust to support any learning process.

Recent research [26, 27] has been focused on integrating persuasion and computational argumentation in a unified system, leading towards a so-called Automated Persuasion System (APS). Persuasion is an activity that involves one party trying to induce another party to believe something or to do something. It is an important and multifaceted human facility. As the authors point out, persuasion is present in many human activities (such as a doctor persuading a patient to drink less alcohol, a road safety expert persuading drivers to not text while driving, or an online safety expert persuading users of social media sites to not reveal too much personal information online). An automated persuasion system (APS) is a system that can engage in a dialogue with a user (the persuadee) in order to persuade the persuadee to do (or not do) some action or to believe (or not believe) something. To do this, an APS aims to use convincing arguments in order to persuade the persuadee. Computational persuasion is the study of formal models of dialogues involving arguments and counterarguments, user models, and strategies, for APSs. The authors claim that a promising application area for computational persuasion is in behavior change (particularly in the context of healthcare organizations, where there is much interest in changing behavior of particular groups of people away from actions that are harmful to themselves and/or to others around them). In our opinion, education is also an area where APS could play an important role (as students need typically to be persuaded of carrying out different goals as part of educational processes—e.g., carrying out some particular kind of exercise, mastering some skill, etc.).

Along this chapter, we have carried out an analysis of the impact and possibilities of computational argumentation from different perspectives, based on some common elements of argumentation provided in Sect. 9.2 (including a brief account of Defeasible Logic Programming). In Sect. 9.3 we showed how argumentation and traditional recommender systems can be unified into argument-based recommender systems, in which recommendations associated with the outcome of a particular query on a certain domain can be backed up by arguments that have emerged as ultimately accepted after performing a dialectical analysis. Then, in Sect. 9.4 we moved into mining opinions from user content knowledge (particularly information in Twitter). We showed that argumentation can provide the backbone for an enhanced model in which arguments are given by sets of information units (e.g. tweets) that

have a prevailing sentiment. Such arguments could also be contrasted using a dialectical analysis, identifying so-called "opinion trees" (an alternative form for representing dialectical analysis). Finally, in Sect. 9.5 we analyzed the concept of shared knowledge awareness in a group of students. We showed that for solving a particular problem (e.g. making a printer work), students might need to combine pieces of information from their own knowledge, and that potential conflicts and inconsistencies might arise. Once again, argumentation comes out as a solution for such situations, allowing students to be aware of their "shared knowledge" that contributes to finding a solution for a given problem.

In summary, we have shown that computational argumentation can indeed provide a powerful model for recasting and enhancing traditional educational processes (particularly those in which incomplete and potentially inconsistent information is at hand, and different, alternative viewpoints have to be assessed and contrasted). We think that the three alternatives explored in this chapter (argument-based recommendation, argument-based opinion mining and argument-based shared knowledge awareness) illustrate the power of argumentation as a backbone for developing new, different intelligent techniques and approaches for educational purposes. Even though many advances have been achieved, the most promising results in this direction seem still to be seen in the future.

Acknowledgements This research was supported by Projects PICT 2014-0624 and PGI 24/N051(Universidad Nacional del Sur and ANPCyT, Argentina).

References

1. A.J. García, G.R. Simari, Defeasible logic programming: an argumentative approach. Theory Pract. Logic Programm. 4.1+ **2,** 95–138 (2004)
2. P. Besnard, A. Hunter, *The Elements of Argumentation* (The MIT Press, 2008)
3. I. Rahwan, G. Simari (Eds.), *Argumentation in Artificial Intelligence.* Springer (2009)
4. S. Modgil, F. Toni, F. Bex, I. Bratko, C. Chesñevar, W. Dvorak, M. Falappa, X. Fan, S. Gaggl, A. Garcıa, M. González, T. Gordon, J. Leite, M. Mozina, C. Reed, G. Simari, S. Szeider, P. Torroni, S. Woltran, The added value of argumentation, in S. Ossowsky (Ed) The Law, Governance and Technology Series (LGTS), "Agreement Technology Handbook" vol.8, 357–404 (2012)
5. M. Nicolas, S. Parsons, I. Rahwan, Argumentation in multi-agent systems: context and recent developments. *International Workshop on Argumentation in Multi-Agent Systems.* Springer, Berlin, Heidelberg (2006)
6. C. Chesñevar, A. Maguitman, G. Simari, Argument-based critics and recommenders: a qualitative perspective on user support systems. Data Knowl. Eng. **59**(2), 293–319 (2006)
7. C. Chesñevar, A. Maguitman, G. Simari, Recommender systems based on argumentation, in "*Emerging Artificial Intelligence Applications in Computer Engineering*". Maglogiannis et al (eds). *Frontiers in Artificial Intelligence and Applications*, IOS Press, vol. 160, pp. 53–70 (2007)
8. C. Briguez, M. Budán, C. Deagustini, A. Maguitman, M. Capobianco, G. Simari, Towards an argument-based music recommender system. COMMA **2012**, 83–90 (2012)
9. Briguez, E. Cristian et al., Argument-based mixed recommenders and their application to movie suggestion. Expert Syst. Appl. **41**(14), 6467–6482 (2014)

10. N.K. Tselios, N.M. Avouris, M. Kordaki, Student task modeling in design and evaluation of open problem-solving environments. Educ. Inf. Technol. **7**(1), 17–40 (2002)
11. P. Rodríguez, S. Heras, J. Palanca, J.M. Poveda, N. Duque, V. Julián, An educational recommender system based on argumentation theory. AI Commun. **30**(1), 19–36 (2017)
12. M.J. Pazzani, D. Billsus, Content-based recommendation systems, in *The Adaptive Web*, pp. 325–341. Springer, Berlin, Heidelberg (2007)
13. J.B. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative filtering recommender systems, in *The Adaptive Web*, pp. 291–324. Springer, Berlin, Heidelberg (2007)
14. T. Hofmann, Probabilistic latent semantic indexing, in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 50–57 (1999)
15. Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
16. K. Grosse, M.P. González, C. Chesñevar, A. Maguitman, Integrating argumentation and sentiment analysis for mining opinions from Twitter. AI Commun. **28**(3), 387–401 (2015)
17. C. Chesñevar, A. Maguitman, E. Estévez, R. Brena, Integrating argumentation technologies and context-based search for intelligent processing of citizens' opinion in social media, in *Proceedings of 6th International Conference on Theory and Practice of Electronic Governance*, ICEGOV '12, pp. 171–174. ACM Press (2012)
18. Chesñevar, C. Iván, A. Gabriela Maguitman, M.P. González, Empowering citizens through opinion mining from twitter-based arguments. *Proceedings of the 8th International Conference on Theory and Practice of Electronic Governance* (2014)
19. C. Collazos, L. Guerrero, J. Pino, S. Ochoa, Introducing knowledge-shared awareness. *Proceedings of IASTED'02*, USA, pp.13–18 (2002)
20. M. González, C. Chesñevar, C. Collazos, G. Simari, Modelling shared knowledge and shared knowledge awareness in CSCL scenarios through automated argumentation systems. CRIWG **2007**, 207–222 (2007)
21. E. Aronson, N. Blaney, C. Stephin, J. Sikes, M. Snapp, *The Jigsaw classroom* (Sage Publishing Company, Beverly Hills, CA, 1978)
22. J. Teze, S. Gottifredi, A. García, G. Simari, An approach to generalizing the handling of preferences in argumentation-based decision-making systems. Knowl. Based Syst., 189 (in press) (2020)
23. J. Teze, L. Godo, G. Simari, An argumentative recommendation approach based on contextual aspects. SUM **2018**, 405–412 (2018)
24. M. Leiva, M. Budán, G. Simari, Guidelines for the analysis and design of argumentation-based recommendation systems. IEEE Intell. Syst. **35**(5), 28–37 (2020)
25. C.E. Briguez, M. Capobianco, A.G. Maguitman, A theoretical framework for trust-based news recommender systems and its implementation using defeasible argumentation. Int. J. Artif. Intell. Tools **22**(04), 1350021 (2013)
26. A. Hunter, Towards a framework for computational persuasion with applications in behaviour change. Argument Comput. **9**(1), 15–40 (2018)
27. L. Chalaguine, A. Hunter, A persuasive chatbot using a crowd-sourced argument graph and concerns. COMMA **2020**, 9–20 (2020)