International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 14 (2022) pp.297-312
© MIR Labs, www.mirlabs.net/ijcisim/index.html

Submitted: 15 December, 2021; Accepted: 21 March, 2021; Publish: 17 May, 2022

An Ontology-driven Architecture for Intelligent Tutoring Systems with an Application to Learning Object Recommendation

Stefano Ferilli[1], **Domenico Redavid**[1], **Davide Di Pierro**[1] **and Liza Loop**[2]

[1]University of Bari, Computer Science Department,
Via E. Orabona 4, 70125 Bari, Italy
stefano.ferilli@uniba.it, *domenico.redavid1@uniba.it*, *davide.dipierro@uniba.it*

[2]LO*OP Center, Inc.,
16511 Watson Rd, Guerneville, CA 95446, USA
lizaloop@loopcenter.org

Abstract: **The state-of-the-art in Artificial Intelligence (AI), the Internet, and the computational power reached by current technologies, allow much more advanced thinking about Intelligent Tutoring Systems than their original definition. The KEPLAIR project envisions an online platform, designed to help all players involved in educational endeavors, especially learners, to improve performance and effectiveness of their activities. Using leading edge AI solutions, KEPLAIR will act as a personalized assistant, helping its users in the entire educational experience, from goal elicitation through learning path definition, selection of materials, performance/attainment testing, analytics and report building. This paper introduces the architecture and functionalities of KEPLAIR as well as illustrating a new methodology for Learning Object (LO) suggestion based on personal profile information.**

Keywords: Adaptive Learning, Knowledge Tracing, Ontologies, Logic Programming, Educational Recommender Systems

I. Introduction

In a world evolving at an incredibly fast pace, there is an ever-growing need for continuous training and education. In traditional education, most teachers cannot give each single student in the class the time and attention needed to overcome his/her individual difficulties or to leverage his particular interests. E-learning takes this problem to the extreme: expanding the audience from the dozens to the thousands which makes it hopeless to have a single method or material fit the needs of all users. On the other hand, adaptive learning aims at addressing the specific differences between individual learners. This should make them more comfortable in the learning effort and allow them to understand and master the concepts more efficiently [5, 39].

Intelligent Tutoring Systems, or ITSs, resulting from the application of Artificial Intelligence to e-learning environments, may be the solution. ICT and AI are nowadays mature enough to allow envisioning an automated system that may act as a personalized tutor for each student, providing him/her exactly the support s/he needs to get the best from the learning experience. Such a system would foster the learner's engagement and motivation by being able to: find topics of personal interest, recognize relevant skills with respect to his/her goals, retrieve missing blocks of knowledge, and customize suggested learning journey. It can make recommendations directly to the independent learner without (or reducing) the direction of teachers and educational managers. It would report to the learners themselves the results of experience with learning materials and of performed activities. It would augment the learner's capacity while secondarily supporting teacher efforts to individualize instruction. Recommending engaging learning experiences is the primary purpose of the system.[1] In this function, we include recommendation of both learning objects/learning materials and learning paths. Learners should find its recommendations to be fun, accessible, engaging, and very relevant to their learning goals thereby fostering engagement and intrinsic motivation. When new interests emerge during a learner's experience, the system will expand recommendations in that direction or change pathways accordingly. One of the advantages of the automated system in carrying out such tasks is that it can take into account many more parameters and data than a human being while carrying out its tasks. The main success criterion for such a system would be learner satisfaction and enjoyment – personal rather than institutional [45].

In this paper, we propose a logical architecture for KEPLAIR (Knowledge-based Environment for Personalized Learning using an Artificial Intelligence Recommender) [16], an ITS

[1]We define educational experience as a combination of Learning Objects chosen according to a learner's profile presented in an accessible environment in order to help a learner achieve his/her goals.

designed to embody such a vision. With KEPLAIR, learners can live educational experiences tailored to their educational goals, prior knowledge, personal interests and preferences, physical and cognitive abilities, and social, financial and geographic contexts. The system will pervasively use symbolic AI to carry out its tasks. Distinguishing characteristics and contributions of KEPLAIR are the extensive set of features it proposes, covering all aspects of the learning process, and its emphasis on explainable behavior which we also overview in this paper. To ensure homogeneity and coordination among all of its subsystems and components, a crucial role is played in our proposal by an ontology. It will act as a schema for the data that informs all the internal representations and behavior, so as to smoothly connect and orchestrate all the various functions and ensure both internal and external interoperability. We will describe this ontology as well. Finally, we will provide a glimpse at the first functions under development: the recommendations of unseen learning objects and paths to the learner, describing the current status of implementation of these functions.

The rest of this paper is organized as follows. After presenting an example of interaction with KEPLAIR in the next section, we will provide the details of its architecture in Section III and will zoom into its knowledge base component in Section IV. Then, in Section V we will describe the initial recommendation strategy that is at the core of the system. Finally, we will discuss related works before concluding the paper.

II. User Experience & System's Functionality

Upon registration of a user, KEPLAIR will collect basic information about him/her to build an initial profile. In addition to form-based interfaces, a natural language dialogue-based interaction will be carried out so that the user may freely decide what information to provide. This data will include personal preferences for social environments, media, and teaching styles. Initial profiles will be used in any interaction to personalize the system's behavior and will be continuously expanded, refined and updated based on both explicit information provided by the users and from implicit information obtained by tracking their interactions. The profile will track their background, skills, preferences, biases or limitations, etc. [33]. Of necessity, KEPLAIR will employ up-to-date digital safeguards to protect users' privacy and data security. E.g., profiles will be associated with internal user identifiers rather than physical persons. Systems databases will include personal information. However, KEPLAIR will not reveal its users' identities, transcripts of activities, or testing outcomes unless the user explicitly instructs it to disclose to specific persons. When group or statistical data is needed, KEPLAIR will use suitable techniques to anonymize user data. In other words, all user data is assumed to be the confidential property of that user, to be released only at the discretion of that user.

When starting a new learning experience, the system will ask the user for his goal, or, if he has no goal in mind, will help him in defining one or identifying a topic of curiosity to explore. Again, dialogue-based interaction will be important for this function. The system's responses will be dynamically built during the dialogue based on the available knowledge and on any previous evolution of the dialogue, not statically determined. Given a goal, KEPLAIR will devise tailored subgoals to create learning paths. If the goal has to do with some formal learning, a personal path will be built that takes into account the institutional requirements as constraints. In any case, constraints coming from the user profile and from specific preferences or indications for the current goal will be fulfilled by the path.

After determining a path, KEPLAIR will retrieve, filter and recommend suitable Learning Objects for the various steps so as to recommend learning activities that are both engaging and productive. Recommendations will be filtered for compatibility with each learner's specific physical and digital contexts as well as their personal preferences and cognitive strengths and weaknesses.

During goal definition and LO selection, KEPLAIR might also propose random items, or items that challenge the user's preferences, offering learners opportunities to expand their understanding and interests rather than creating an educational echo chamber [24, 31]. The reaction of users to these proposals will be recorded in their profiles and used by the AI system to fine-tune and improve their profiles.

KEPLAIR will recommend LOs harvested from many different sources including online repositories of open educational resources, other KEPLAIR users, industry and government resources, internet search engine results, school and college teacher submissions, and other informational resources (e.g. Wikipedia or Open Library). Over time, the various feedback channels built into KEPLAIR will 'teach' it which LOs fit best with individual profiles, goals, and environments. Feedback will come from learners themselves, from trusted teachers and instructional designers, and from a growing base of anonymised learning analytics data.

'Learning Objects' may be digital or physical, delivered via any medium including other people such as fellow learners, learning coaches, teachers, mentors, and subject matter experts. To facilitate interpersonal interaction, KEPLAIR will provide a Social Network environment devoted to discussions and social exchange, purposely organized and designed for educational purposes. It will be able to aggregate virtual and physical communities of users based on shared profiles and to recommend interpersonal connections for those learners wishing to participate in face-to-face interactions. This will facilitate formation of self-organized pods, cohorts, and study or play groups. Information shared by users on the Social Network would be used to refine their personal profiles and the profiles of the communities in which they are involved. Rating functions will also permit crowd-sourced assessment of several aspects of LOs (complexity, correctness, etc.) and allow for microbadges to be earned by users (based on skills, credibility, etc.).

KEPLAIR will track and test learner progress for two independent purposes. The first purpose, formative evaluation, targets the learner's readiness to enjoy and benefit from a specific LO or recommended experience. The focus here is on prerequisite knowledge and skills, learner's expressed interest in a topic or subject, and the relationship between the LO and the expressed goal. Data from formative evaluation of learner performance is used to improve recommendations so that the learner is increasingly motivated to engage with LOs.

This data may also be employed for the second purpose, summative evaluation. Summative evaluation comprises formal testing and examination leading to microbadges, certifications, academic degrees, licenses and the like. These evidences of acquired competencies or achievements are added to the learner's profile and, at the learner's discretion, may be shared with selected individuals, schools and universities, and potential employers. No summative evaluations are required by the KEPLAIR system since it can base recommendations on formative data and self-reported preferences. However, many learners may choose to use KEPLAIR to build and enhance their portfolios of achievement as evidenced by scores on standardized tests and licensing examinations. In addition, some suppliers of LOs and courses may use results from formal testing as prerequisites for access to their materials.

During learners' use of materials, KEPLAIR may, at the learner's discretion, track their explicit (e.g., keystroke, highlighting, ...) or implicit (gesture, posture, expressions, voice, ...) behavior, to infer their emotional status. This information can increase learners' understanding of their comfort and engagement with the materials, incipient boredom, anxiety or discouragement. This information will not be used actively to control or drive the user's behavior; it contributes statistical data that may illuminate learning preferences thereby improving suggested pathways and materials. In the long term, this will improve profiles and understanding of interaction preferences in order to deliver more useful recommendations. Access to anonymised high-level inferred characteristics generated from this data can be available to teachers with permission of learners and used to improve teaching practices.

At suitable moments during the learning path, KEPLAIR might propose formative diagnostic tests to check attainment of partial or overall mastery of prerequisites skills needed to succeed with current learning goals. In addition to tests available in its own repository, the system will integrate formal external testing services to qualify learners for microbadges or other partial or full certifications.

While KEPLAIR's primary focus is on the learner, it can also support teachers and educational managers in individualizing their instruction by exposing learners' personal preferences and unique capacities. Indeed, KEPLAIR can provide educators with analytics and reports about groups of students and, with learner permission, about individuals. By highlighting the key (positive or negative) aspects and issues of performance aggregators, KEPLAIR supports improvement of programs and instructional materials.

It should be emphasized that the KEPLAIR platform is completely general and independent of the end-user interface. Its functionality could be integrated into traditional e-learning platforms offered by existing educational institutions or into new, innovative online services. A particularly interesting direction to improve engagement and interaction is gamification of the user interface, e.g. casting KEPLAIR as an ingame player in a microworld, virtual reality, or augmented reality context.

III. Architecture

The logical architecture of KEPLAIR is shown in Figure 1. Since KEPLAIR is in fact an intelligent agent, some components (SeaL, ReaL and LeaL) are inspired by the architecture proposed in [13] for agents in Ambient Intelligence applications.

A. Environment Layer

The top layer in the architecture represents the environment in which KEPLAIR 'lives' and acts. The main cloud represents the Internet, from where users can connect to KEPLAIR and where many LOs reside (shown as the LO subcloud within the Internet cloud). Additional LOs may be stored on a local, proprietary repository (labeled LO and shown to the left of the Internet cloud). Note that not all LOs need to be explicitly recognized as such. KEPLAIR can find any material on the Internet that may act as a LO for some purpose, or accept LOs submitted by its users, and include them in its metadata repository (small squares in the Internet cloud but outside of the LO sub-cloud). Users interact with KEPLAIR using the SeaL (Sensors, Effectors and Applications Layer) component, that includes all application interfaces and, in case learning takes place in a smart environment, all sensors and effectors available in the environment. This section of the architecture is generic, since KEPLAIR is independent of the specific interface. It may be embedded under existing e-learning environments, or it may develop its own application and facilities.

B. Core Layer

The main system is in the middle layer, that coordinates all processing tasks and processes involved in providing the educational services by pervasively exploiting an underlying AI level that provides all the intelligent functions, and serving the user interfaces. It consists of 3 components, each including several specific modules to carry out its function. The Harvesting Manager is in charge of identifying and collecting the LOs and their associated metadata as well as standardized learning pathways used by institutions. While the LOs may reside on the Internet and be linked by the system, some of them may be acquired and stored locally in the LO repository. For both local and remote LOs known to the system, related metadata describing both their formal appearance and more conceptual information, concerning their content and context, are stored internally in the LO Metadata repository. In addition to those found on the Internet, it may save additional metadata extracted internally from the AI or from the users' feedback. The Metadata repository determines the view that KEPLAIR has on the LOs: when reasoning on them, it can see only what is stored in this repository (still, it may run specific functions that expand the known metadata for a LO, if needed).

The Learning Manager and Social Manager components are in charge of controlling interaction with the users, and of providing them all the functionality exposed by KEPLAIR. In particular, they deal with two different kinds of tasks. The former simulates human interactions via AI, acting as an automatic tutor, counsellor, and personalised assistant to build learning paths, recommend LOs, administer tests to check

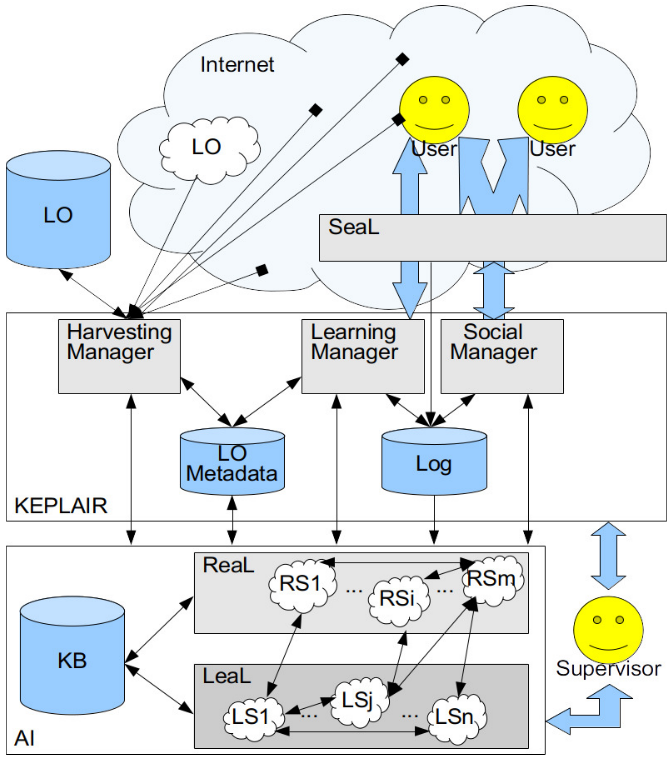

Figure. 1: KEPLAIR's logical architecture

their performance, etc. The latter handles interactions among users and communities of users in the educational context, possibly intervening to provide support. Both of them, during their operation, collect interaction data and store them in a Log repository, which will in turn provide them back to those modules when needed to drive their behavior, and to the AI layer for it to extract useful information and to build and maintain the KB.

C. AI Layer

All components and repositories in the core system communicate with the bottom layer, collecting a number of AI tools, systems and resources needed to support the advanced functions of KEPLAIR. This is where the actual intelligence of KEPLAIR resides.
All the knowledge useful for KEPLAIR's operation is stored in a Knowledge Base, including the users' profiles (built from both information explicitly provided by the users and information automatically extracted from the Log repository). According to the most recent trends in AI, it is conceived as a Knowledge Graph, in order to enable high-level reasoning functionality in addition to standard data management. To obtain both efficient data handling and effective knowledge manipulation, it is organized as prescribed by the GraphBRAIN technology [12, 17]. In that setting, data are stored in a graph DBMS, and their schema is stored in the form of a comprehensive ontology, possibly obtained by merging and aligning several domain-specific ontologies. The ontology is a fundamental reference for the entire KEPLAIR system, coordinating all of its components and modules during their interactions and information exchanges. It is fundamental to ensure that they are interoperable and assign the same meaning to the same pieces of data.
KEPLAIR will exploit state-of-the-art solutions coming from different fields of AI to support all of its functions. These include, but are not limited to:

Recommendation Systems to recommend specific LOs based on explicit and implicit information extracted by user profiles.

Natural Language Processing to manage dialogues with the users and to extract relevant information from text (e.g., from the content of LOs, the discussions on the Social Network, or the dialogues between KEPLAIR's chatbot and the users).

Pattern Recognition and Image Processing to extract relevant information from images, both in the LOs and from the environment.

Planning to build suitable learning pathways.

Social Network Analysis to enrich the users' profiles, detect trends and communities, establish materials' and people's credibility, or to harvest and rate educational resources (including other people).

Machine Learning and Data Mining to extract relevant knowledge from the materials and from user interactions so as to improve future system's behavior.

Knowledge Representation and Reasoning to build and maintain the ontology and carry out complex inferences on the data in the KB.

Information Retrieval to identify appropriate materials to satisfy the users' information needs.

User Modeling to build and maintain user profiles needed for personalization.

Sentiment Analysis to understand the state of mind of the users in performing activities.

Ambient Intelligence to get information from the learning environment and possibly interact with it using IoT technology.

Both sub-symbolic and symbolic techniques are exploited by the AI layer. The former are for tasks that do not require explanations. Since KEPLAIR should always be able to explain its high-level decisions and actions to its users, a strong emphasis is placed on symbolic, glass-box (interpretable and explainable) techniques, so that KEPLAIR's decisions and actions can be justified in human-level terms to its users.
The AI layer consists of the KB and of two sub-layers associated to two different kinds of AI tasks (namely, knowledge exploitation and knowledge creation). These three items are strongly interconnected, and specifically the two sub-layers are directly connected to the KB in order to draw knowledge from it and/or to modify it based on the outcomes of their reasoning efforts.
The ReaL (for Reasoning Layer) sub-layer is an inference engine that supports and drives the intelligent behavior of KEPLAIR. This component is based on original Multistrategy Reasoning approaches developed by our group [11, 14, 28]. It applies known models to the data coming from KEPLAIR and to the knowledge stored in the KB in order to infer additional useful knowledge and make decisions. ReaL may include different systems (the RSi's) to carry out several kinds of reasoning (deduction, abduction, abstraction, argumentation, etc., possibly under fuzziness and uncertainty), to handle different kinds of data, and to implement different kinds of approaches (e.g., Classification systems to support system's decisions, Process Management systems to check system's and user's behavior, Social Network Analysis tools to manipulate networked information, etc.). Some of the inference strategies provided by ReaL will be used by our recommendation technique described in V. One is associative reasoning, such as graph traversal, applied to the graph structure of the KB. Another is ontological reasoning, by applying ontological reasoners on a rendering of the schema and of the graph DB to Semantic Web languages (i.e., RDF and OWL respectively). This kind of reasoning may also exploit additional information, external to the KB, taken from ontologies available in the Web. Another is (Prolog-like) rule-based logical reasoning expoiting clauses defined on the concepts and relationships specified in the schema.
In an application domain so complex as education, the models cannot be static, but need to adapt in time so as to improve their performance and become more and more effective and tailored to the users. This requires Machine Learning and Data Mining services (the LSj's), to discover new knowledge, collected in the LeaL (for Learning Layer) component. Different kinds of tools and techniques are present and interplay, to support the different kinds of models used in ReaL (e.g., Rule Learners for the classification models, Process Mining systems for the behavioral models, etc.). Particular emphasis will be given to incremental approaches, allowing for progressive refinement and adaptation of the KB.
Not only explanations can be provided to final users. Human supervisors may also check the system's behavior at both levels (KEPLAIR and AI), and if needed intervene to modify, fix or adjust the models used by the AI systems or their outcomes, in case of wrong or biased decisions. Even more, the feedback of the supervisors (and of the users, as well) will be used by modules in this layer to expand and refine the KB.

IV. Ontology and Knowledge Base

As said, all of KEPLAIR's behavior and functionality is informed by an ontology. In this section we provide an account of the main elements of the ontology, and of the solutions we adopted to design and manage the KB using the ontology as a schema.

A. Ontology Overview

In designing the ontology for KEPLAIR we started from existing schemas and ontologies used in the Digital Libraries and education fields, trying to merge them in a coherent compound schema. Specifically, for the educational domain our starting point was the set of ontologies used in the IntelLEO European project[2], including ALOCOM [43] and SCORM[3]. We also took into account the Learning Object Metadata (LOM) standard [1]. For the digital library domain, useful to describe documents and learning materials, we considered the Functional Requirements for Bibliographic Records (FRBR) of the International Federation of Library Associations and Institutions (IFLA) [23], the Dublin Core Metadata Initiative (DCMI)[4], the METS[5] and the Open Archives Initiative Object Reuse and Exchange (OAI-ORE) standards[6] for the description of compound objects. We further expanded the resulting ontology by adding classes, attributes and relationships to account for the content and context information that is often neglected in the state-of-the-art, especially concerning the specificities of each individual student.
It includes several general classes relevant to the education domain, among which are **User**, whose identity may be anonymized, and which represents the concept of the learner that uses the system to earn new knowledge; **Person**, whose identity is known, possibly associated with a user, which represents any possible relevant person for users. **Place**, with its several subclasses for describing geographic, administrative data or locations, represents the places in which the user is immersed or places mentioned in some LOs. **Environment** describes the different kinds of environments in which users can be immersed. **Activity** represents the set of tasks to be performed to reach any kinds of goals. Other useful classes are **Organization** (public or private) and **Event**. These classes are borrowed from an ontology of general, widely used concepts, reusable in most domains.

[2]`https://cordis.europa.eu/project/id/231590`
[3]`https://scorm.com/`
[4]`https://www.dublincore.org/`
[5]`https://www.loc.gov/standards/mets/`
[6]`https://www.openarchives.org/ore/`

Then, there are the domain-specific classes for education. Since LOs may take many very different forms, we represent them through several classes. Most prominent kind of LO are documents, represented by the **Document** class and its several subclasses at different levels (the first one distinguishing Printable, Audio and Video documents). **AssessmentTool** represents tools that can be used to check if users possess given skills (e.g., tests). Other classes are **Container** (to describe classes, courses, etc.) and **AccomplishmentEvidence** collects certifications (microbadges, certificates, licenses, standardised tests, degrees, diplomas) stating that a user possesses given skills, **Artifact** (including Handicrafts, IndutsrialWorks, and Artworks, such as statues and paintings), **Device** (including simple tools, such as hammers, and more complex systems, such as computers), **Software** (again, with a taxonomy of sub-classes for the different kinds of software).

For fine-grained handling of LOs (and especially documents), KEPLAIR's ontology also allows for description of their structure and content. Class **DocumentDescription** allows for description of documents' layout (Page, Table, ...) and logical (Title, Section, ...) structure, their text and its grammatical structure (Sentence, Subject, Object, ...). Class **ContentDescription** allows for description of several kinds of document content (including, e.g., Concepts, Subjects, Languages, Keywords, etc.). Concepts and Subjects can be organized into taxonomies (expressed using class **Taxonomy**) and can be inter-related (e.g., Subjects can be connected by generalization/specialization relationships, concepts may be connected by equivalence relationships, Words may be connected to the Concepts they express). Instances of these classes are connected to the documents they describe using relationship **describes**. An attribute **degree** allows expression of the strength of a relationship. Also other classes such as **Environment**, **Person** and **Place** can be used as subjects of this relationship to specify further information, including Named Entities (persons, places, etc.) mentioned in the documents or anyway related to it.

The KB must also store user profiles, including more than just education-related concepts. Two kinds of profiles are handled in KEPLAIR. *Long-term* profiles include more 'stable' information: demographic data, educational transcript, a résumé of experiences, relevant hobbies, cognitive strengths and weaknesses and, importantly, personal preferences. *Short-term* profiles include contingent information about the context or environment in which the user is immersed (social, physical, geographical and emotional). In particular, KEPLAIR must be able to describe goals, preferences and skills of users, since they are the key elements for all educational purposes. These are very abstract concepts, hard to capture. A goal may be a certification or a skill, but the concept of goal is much broader than this, especially outside formal learning environments. A skill can be a subject or a technique, etc. Preferences may concern subjects, environments, etc. For this reason, the ontology does not provide classes 'Goal', 'Preference' or 'Skill'. These concepts are expressed through several classes, such as **ContentDescription**, **Environment**, **Place**, etc. As for LOs, the relationships linking the users to the instances of these classes that describe them is **describes**. An attribute **type** specifies the kind of relationship, with values 'Goal', 'Preference', 'Skill'. E.g., a subject (such as 'Mathematics') may be a goal, but also a word (e.g., 'astronaut') might be used for this purpose; in the latter case, KEPLAIR would use the relationships in the KB to connect the words to related subjects and skills and, ultimately, to the appropriate LOs.

KEPLAIR provides many other relationships to connect instances of these classes. Some are needed to organize documents into layout and logical components (e.g., **has** links a Document to its layout and logical components; **partOf** structures components into sub-components, etc.) LayoutComponents also have a spatial organization, expressed by relationships **leftOf** and **above**. Authors of Documents and LOs are connected to them through the **developed** relationship. Due to lack of space we will not mention all the other relationships, most of which can be easily guessed by the readers (e.g., those linking Persons to Organizations, or those describing the syntactic and semantic structure of text.

Of course, in KEPLAIR there are different roles which must be represented. When needed, they are specified by property **role** of several relationships. E.g., in relationship **wasIn** between a User and a Class (to express that a user was involved in a class) possible values of this property might be 'Student' or 'Teacher'; in relationship **developed** between a User and a Course (expressing that a user developed a course) possible values of this property might be 'Designer' or 'Creator'. **User**s may also be monitored and evaluated during their activities and for this the relationship **evaluated** becomes necessary. Any LO can be consulted by any users and this piece of information will be stored in the **interactedWith** relationship.

B. Knowledge Base

As mentioned in Section III, the knowledge base that enables all intelligent functions in KEPLAIR is organized as a knowledge graph. More specifically, it is implemented as a graph database. There are several reasons for choosing the graph DB technology over traditional relational databases (or other NoSQL models). Graph DBs provide optimized and efficient solutions for instance-based data navigation, as opposed to batch data processing of entire tables using join operations in relational DBs. Also, graph DBs are a kind of NoSQL DB for which no data schema is needed. While we still require schemes (in the form of ontologies) to inform the DB and drive data handling, this allows us great flexibility in imposing different schemes on the same data, depending on the specific needs at hand. It is especially important since KEPLAIR aims at collecting and combining data of many different kinds, coming from many different sources. As explained in [17], graph representation is also perfect to support many AI tasks, including automated reasoning and social network analysis, which are the core of KEPLAIR's innovative features.

In particular, in our implementation we adopted the GraphBRAIN technology [17], that is based on Neo4j, currently the most prominent graph DB available [37]. It is based the Labeled Property Graph (LPG) model. In a nutshell, it allows associating both labels and properties, expressed as key-value pairs, to both nodes and arcs in the graph. Whilst the LPG model is incompatible with the RDF triple-based model traditionally used in the ontology community, we preferred the LPG perspective because KEPLAIR is meant to be an operational system, and thus flexibility of representation and efficiency of data handling are prioritary in our approach. GraphBRAIN uses ontologies as graph DB schemas, but keeps the ontology/schema separate from the data, so as to allow applying different ontologies to the same data in order to reflect different perspectives on them. Ontologies in GraphBRAIN are described using a specific formalism (GBS) written in XML according to a specific DTD. Such a formalism directly reflects the LPG organization, but there exists a mapping that allows exporting GraphBRAIN KGs to standard ontological formats (namely, OWL) and vice-versa [12]. This ensures smooth import and export of both the ontologies and the data from/to external repositories.

C. Basic AI Functions on the Graph

As said, part of the AI functions embedded in KEPLAIR rely on network analysis and graph mining solutions. A set of general tools of this kind is provided by GraphBRAIN [17]. Here we will describe those that are reused by KEPLAIR and that play a role in the recommendation functionality we will describe in the next section.

Subgraph Extraction One function allows KEPLAIR to extract from the KG subgraphs that may be relevant for specific purposes. This is useful because automated reasoning techniques usually do not scale to very large KGs. So, being able to select only the portion of KG that contains relevant information for one's purposes is crucial. Starting from a set of nodes that represent items for which we need information, KEPLAIR includes several options to extract a portion of the graph that should include all relevant information about each of them in isolation and about them as a whole. Two solutions are based on the relevance value associated with nodes directly or indirectly connected to the starting nodes, and return the subgraph of all nodes having a relevance value that exceeds a given threshold according to the following algorithms:

Spreading Activation [42] Each of the starting nodes takes an initial relevance value; each node that is adjacent to a node with a relevance value recursively obtains a relevance value that is a fixed portion of that node.

PageRank [35] Using the well-known algorithm designed to assign a relevance value to interconnected Web pages based on the number and relevance of the pages having incoming or outgoing links of the pages.

These subgraphs may include nodes that are reachable from any of the starting nodes but are not necessarily connected to any of the other starting nodes. Another solution returns only nodes that are on the path between at least two of the starting nodes; it is based on:

Shortest Paths The set of all shortest paths connecting any pair of nodes in the starting set is returned.

The subgraphs returned by these algorithms can be used separately or merged together.

Centrality assessment Knowing the relevance of a node in a (sub-)graph may be useful to understand its importance in the context described by the subgraph. Note that this kind of relevance is different than that considered in the subgraph extraction algorithm: in the latter, it is relative to a set of starting nodes; here, it is referred to the subgraph itself, independently on how it was generated. Many centrality assessment strategies have been proposed in the literature, and KEPLAIR may exploit several thereof:

- Closeness Centrality [34];
- Betweenness Centrality [44];
- Harmonic Centrality [9];
- Katz Centrality [47];
- Page Rank Centrality [35].

Other functions Finally, among the other available functions we mention:

- **Link Prediction** [48]: used to discover unknown relationships between graph nodes. This information can be considered a form of abduction, and allows us to support associative and logical reasoning with information that is missing in the KB.
- **Clustering** [40]: used to partition the nodes in the graph (or in part of it) into groups (defining subgraphs) based on a given similarity function. This may allow finding strictly related groups of items on which further reasoning can be carried out.

Note that these techniques may be combined to obtain more complex behavior and non-trivial insights in the KG content. E.g., the centrality of nodes may be used to extract a subgraph of the whole KG representing the 'most relevant core' of information, by selecting the nodes with highest centrality in the whole KG and applying on them the subgraph extraction tool.

V. Course and Learning Materials Recommendation

We will now describe a prototype of the first high-level function we implemented in KEPLAIR, namely the recommendation of courses and learning materials to students. With reference to KEPLAIR's architecture, the recommendation engine is located in ReaL, and draws from the KB the knowledge necessary to carry out its task.

A. Recommendation Strategy

The recommendation function currently embedded in KEPLAIR leverages the information expressed in the form of descriptors (to which we will refer as *tags* in the following) associated to the items (users, courses, materials) in the KB. The tags may have been obtained by the creator of the course or material, by feedback from KEPLAIR users, using the metadata found on the Internet, or by inference carried out by the system. They may concern the subject matter content (computer science, mathematics, etc.) and/or any other kind

of information expressible through descriptors: e.g., the way in which it is presented (text, video, etc.). The recommendation strategy is based on a mix of different kinds of computation, as proposed in [15]: graph mining and network analytics are exploited to obtain the relevant information from the KB; reasoning is used to drive the graph navigation and exploit high-level knowledge that is not explicitly expressed by the instances, and finally numeric computation is used to compute relevance scores needed to filter and rank recommendations.
The current prototype provides for two kinds of recommendations: courses and learning materials. Thus, a first step in our strategy consists in selecting courses or learning materials that can be candidates for recommendation. For this, we use associative reasoning (namely, the subgraph extraction algorithms described in Section IV-C), using as starting nodes the user and:

- the courses the user has already attended, in the case of course recommendation;
- the materials belonging to the current course, plus the materials that the user has already exploited, in the case of learning material recommendation.

From the extracted subgraph, only nodes corresponding to courses (resp., learning materials) are extracted as candidates, and the courses (resp., learning materials) already exploited by the user are removed. Logical reasoning is used to remove the courses (resp., learning materials) that the user has already exploited or that are incompatible with his profile.

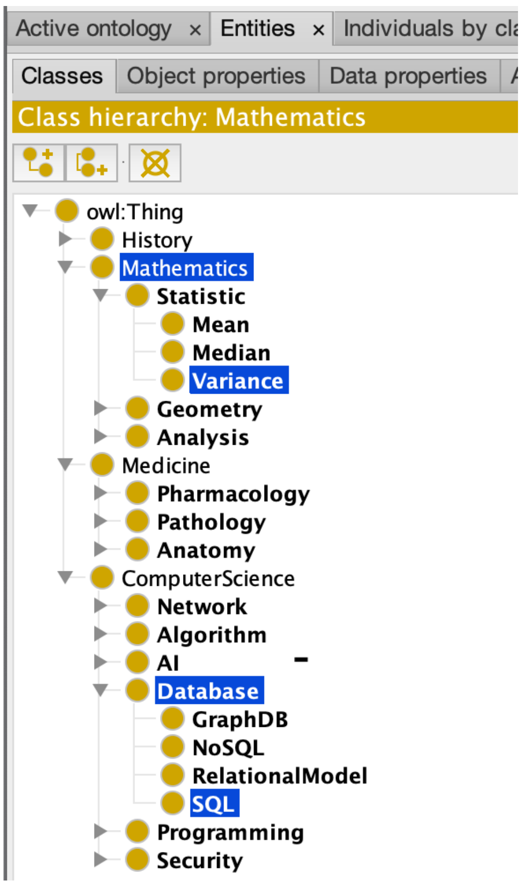

Figure. 2: Fragment of Subjects' Ontology

1) Descriptors selection

Now, for the user node, and for each of the courses (resp., learning materials) nodes, the corresponding descriptors must be extracted. All such items (learning materials, courses and even users) may be directly linked to descriptors describing them in the KB. E.g., it may be expected that much information is associated directly to learning materials to describe them in detail. Less information might be expected for courses (perhaps just the topics they concern), and even less for users (perhaps just a few topics of interest explicitly mentioned by them). Still, additional relevant descriptors may be associated to them using associative reasoning. E.g., information about courses may be extended by considering the information of the associated learning materials, and information about users may be extended by considering the information attached to the courses and materials they used. The descriptors can be filtered using logical reasoning to remove those that are incompatible with the user profile, or that are obtained through forbidden paths. Also, ontological reasoning can be used to remove items from a set of descriptors including many items from a taxonomy (e.g., many topics in a classification system, such as the one in Figure 2), those that are generalizations of other ones in the same set (e.g., 'Medicine' is removed if 'Anatomy' is also present).
Now, any item is described by a set of tags, independently of its type (material, course, user). This allows us to compare any two of them, of the same or different type (e.g., materials to materials, materials to users, etc.). For this, we need a similarity assessment strategy.

2) Tag Distance

We base our similarity assessment on the distance between nodes (tags) in the graph, expressed as the number of arcs in a path connecting them (specifically, we consider the shortest paths returned by the function described in Section IV-C). The distance of a node from itself is obviously 0; it is undefined when there is no path connecting those nodes in the graph. For instance, considering the topic subgraph in Section V-C, the distance between tags 'database' and 'sql' tags is 1; that between 'tissues' and 'bacteria' is 4; finally, the distance between 'geometry' and 'pharmacology' is undefined, sinte there exists no path among them in the graph.

3) Score Function

Now, given two items to be compared, they are described by two sets of tags. We base the similarity assessment between sets of tags on a modified version of the Jaccard index, a simple but widely used [19, 22] formula for assessing the similarity between two sets.
In our case, however, the set members are not independent in general. They may be directly or indirectly related in the knowledge base. For this reason, instead of set operations, we leverage the notion of distance introduced above, and compute the ratio between the sum of the (defined) distances

between pairs of related tags and the number of pairs having a defined distance. The resulting value is used as a parameter in the following function:

$$s(x) = \frac{1}{a^x}$$

for some $a > 0$. Independently of a, we have that:

$$x = 0 \Rightarrow f(x) = 1$$

$$\lim_{x \to \infty} f(x) = 0$$

(a only determines the slope of the curve). So far, non-related pairs (having undefined distance) have not played any role. To take them into account, we further compute a penalty to be used to smooth the score:

$$P = 1 - r^b,$$

where r is the ratio of non-related pairs of tags in the two sets over the total number of tags, while b is a parameter that affects the weight of the penalty. When all pairs of tags are related, $r = 0 \Rightarrow P = 1$, so no penalty will be applied to the score.

Let us show an example score computation, based again on the taxonomy of subjects shown in Figure 2. After some experiments, we decided to use parameter values $a = 3$ and $b = 0.5$ (i.e., the square root). Consider the sets of tags A = [database, sql, mathematics, variance] and B = [sql, pharmacology, statistics].

1. Compute reduced tag lists:
   - rA = [sql, variance]
   - rB = [sql, pharmacology, statistics]
2. Compute distances between pairs of tags in $rA \times rB$:
   - D = [$-$, $-$, $-$, 1, $-$, $-$, 1, 0]
3. Compute the distance:
   - DJ = (1 + 1 + 0) / 3 $\approx$ 0.666
4. Compute the score function:
   - Y = 1 / $3^{0.666}$ = 0.4807
5. Compute penalty:
   - R = 5 / 8 = 0.625
   - P = 1 - $\sqrt{R}$ = 1 - 0.791 = 0.209
6. Apply penalty:
   - S = 0.4807 * 0.209 = 0.100

Summing up, the score computed between [database, sql, variance] and [sql, pharmacology and statistics] using our sample data in our setting is 0.100.

B. Implementation

Albeit existing e-learning platforms are inspired by, and developed for, supporting traditional courses and learning institutions in their activities, we opted for selecting an existing platform in which to embed KEPLAIR's functionality at this early stage of development. After all, a learning path created by KEPLAIR may be seen as a course, and the system may store existing (off-the-shelf or created by itself) courses for reuse and adaptation. At this stage, we focused on the functions of proposing lists of courses to the users, showing the list of lessons for each course, and listing learning materials for such lessons.

1) Platform

Among the existing Open Source e-learning platforms, we opted for OpenOLAT[7], since it is one of the few LMSs written in Java language, that provides us a very large set of libraries, full programming functionality compared to scripting languages, and many possibilities to interact with other techonlogies, including modules written in different languages. Our intervention in the inner workings of OpenOLAT concerned the alignment of its database to KEPLAIR's knowledge base and the embedding of our recommending engine based on Prolog.

Concerning the interface of OpenOLAT, we modified two pages: the main page where all courses are listed, and the specific pages of each lesson, by adding a bar on the top that shows the list of recommendations. The former shows the courses that may be of interest to the learner based on his past activities. The latter shows the Learning Objects related to the subject of the lesson, recommended without taking into account the user's preferences.

This behavior was inspired by recommendations in YouTube, that appear both in the user's homepage and in each specific video page next to the video itself.

2) Knowledge Base

Figure 3 shows a portion of the data model used in OpenOLAT that has some overlapping with the ontology used as a data schema by KEPLAIR. A few sample instances are also shown. In particular, the entities that are most relevant to our recommendation function are 'Student' (identified by a username), 'Course', characterised by an id and a name, and 'Learning Object', characterised by an id, a name and a list of tags. These three classes are represented in our ontology in this way: **User** represents the concept of "Student", "Course" is mapped with our **Container** class and the generic "Learning Object" expressed in OpenOLAT can be mapped as the union of three classes: **Document**, **AssessmentTool** and **Container**. As an example referring to Figure 3, 'Marco' becomes an instance of **User**, 'Intelligenza Artificiale' becomes an instance of **Container** and 'Slide Concettualizzazione' an instance of 'Document'. The relevant relationships express the set of Learning Objects read by a Student, and the Course to which the Learning Objects are associated. The portion of the schema relevant to the recommendation functions had to be aligned and synchronized with

[7]See `https://github.com/OpenOLAT/OpenOLAT` for code and documentation.

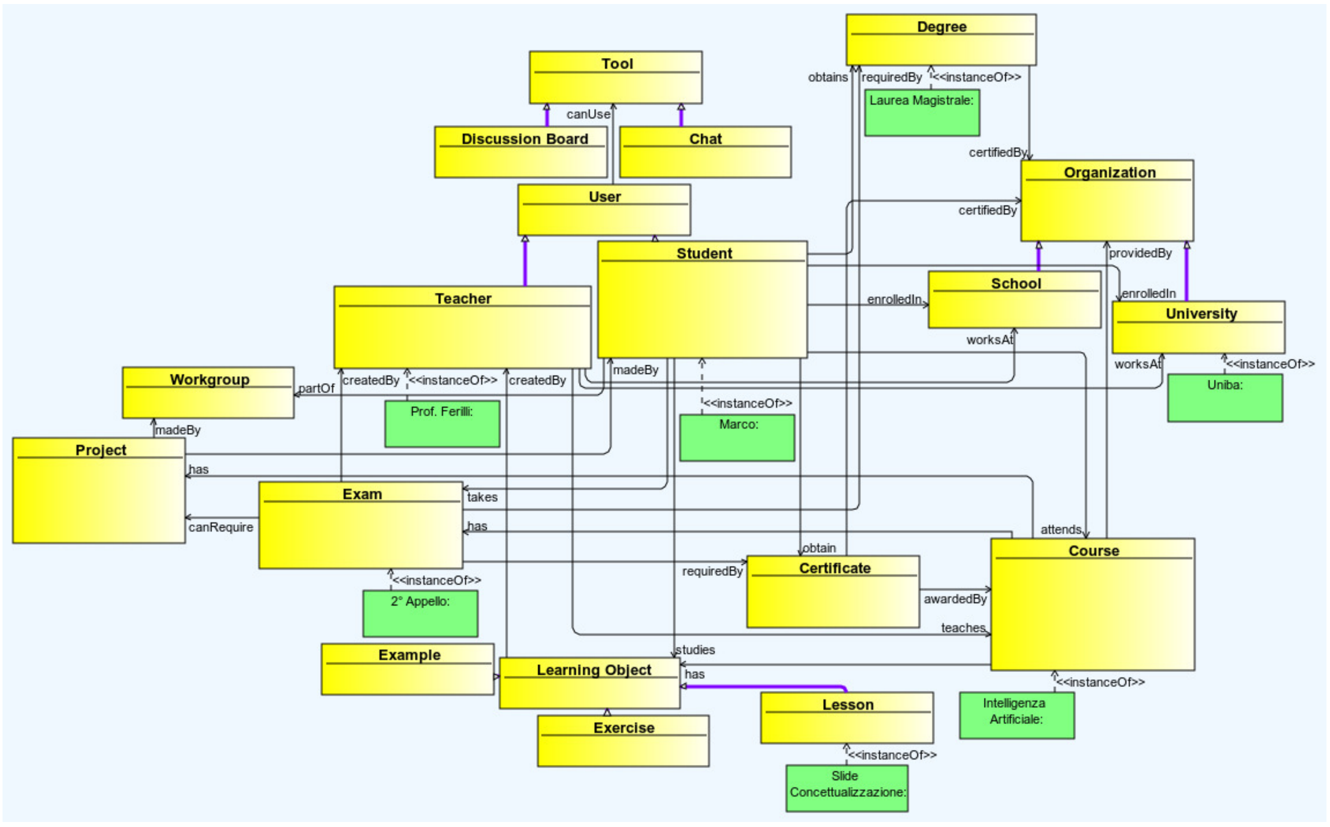

Figure. 3: Conceptualization of OpenOLAT components with example instances

KEPLAIR's ontology, in order to handle the same concepts and synchronize their instances. Of course, the KG contains more information (especially contextual) about those instances. Since OpenOLAT does not provide a specific representation of tags to describe LOs, we inserted the tags in the 'Description' field, using a '#' prefix to recognize them (so, tags cannot contain blank spaces).

3) Recommendation Engine

Our recommendation strategy was implemented in Prolog, a general-purpose programming language specifically suited for AI applications that require reasoning rather than algorithmic or numeric computations. Indeed, Symbolic (and especially First-Order Logic) approaches in AI are particularly relevant when the system's behavior has to be explained in human-level terms, including all the reasoning steps that led to its conclusions (something that is not allowed by subsymbolic methods, based on mathematical/statistical aproaches). In the education domain, specifically, we believe this is fundamental to make the users aware of what happened, and let them provide feedback and correction if needed.
When a recommendation is needed, the relevant portion of the KG is selected and exported as a set of Prolog facts, based on different predicates. Some are omitted (learningObjectTagList, member, ...) and we report the most relevant ones:

student(X) : X is the username of a registered user

course(X,Y) : Y is the name of the course having id X

learningObject(X,Y) : Y is the name of the LO having id X

learningObjectTag(X,Y) : LO X is described by tag Y

hasLearningObject(X,Y) : course X includes LO Y

hasRead(X,Y) : user X used LO Y

The set of rules that act on these facts to carry out our recommendation strategy is organized by function into several groups:

- predicates extracting the relevant tags for an item:

  userHasReadTag/2 : returns the tags of an object seen by the user.

  taglist/2 : returns all tags describing an item.

- predicates extracting nodes and paths in the graph based on the navigation strategies used by our approach;

- predicates computing distances in the graph between items or lists of items, e.g.:

  distance/3 : returns the distance between two nodes, if it exists, or *undefined* if they are not connected in the graph.

  distanceTaglistTaglist/3 : returns the list of all pairwise distances between the tags in two lists.

- predicates computing single and aggregate scores, e.g.:

scoreFunction/2 : computes the score function seen by the user and all unseen objects.

orderedScoreUserObject(X,O,S) : returns all objects to be recommended to the user, ranked by score.

C. Use Case

To carry out a preliminary test of our recommendation approaches, we populated KEPLAIR's KG with sample data. As to the topics, we considered the domains of computer science, mathematics, humanities and medicine. A fragment of the topic taxonomies rooted in such disciplines and uploaded in KEPLAIR's KG is shown in the table 1. This taxonomy has been represented from the ontology in Figure 2.
We also added information about a few test learners. E.g., a portion of the information associated to user 'davide', expressed as Prolog facts used by the recommendation engine, is shown in Figure 4.
Let us show a use case in this setting, concerning user 'davide'. Applying the methodology described above to recommend new lectures to 'davide', based on the lectures he has already attended. Specifically, 'davide' attended lectures about: Database, Programming, AI, Programming Languages, History, Computer Networks, Data Mining.
Using parameter values $a = 3$ and $b = 0.5$ for the score computation, here is the set of lectures recommended to 'davide', ranked by decreasing score. With reference to the Prolog engine, they were obtained running query:

```
?- orderedScoreUserObject('davide', X,
_).
X = 'Lecture 3 - Computer Networks' ;
X = 'Lecture 1 - Cybersecurity' ;
X = 'Lecture 2 - Cybersecurity' ;
X = 'Lecture 4 - IA' ;
X = 'Lecture 1 - Algorithms and Data Structures' ;
X = 'Lecture 3 - IA' ;
X = 'Lecture 3 - Programming Languages' ;
X = 'Lecture 3 - Algorithms and Data Structures' ;
X = 'Lecture 5 - Programming' ;
X = 'Lecture 5 - IA' ;
X = 'Lecture 4 - Programming' ;
X = 'Lecture 4 - Database' ;
X = 'Lecture 3 - Programming' ;
X = 'Lecture 3 - Database' ;
X = 'Lecture 2 - Programming' ;
X = 'Lecture 2 - Database' ;
X = 'Lecture 2 - Computer Networks' ;
X = 'Lecture 3 - Data Mining' ;
X = 'Lecture 3 - Cybersecurity' ;
X = 'Lecture 3 - History' ;
X = 'Lecture 2 - History' ;
X = 'Lecture 2 - Programming Languages' ;
X = 'Lecture 2 - Algorithms and Data Structures' ;
X = 'Lecture 2 - IA' ;
X = 'Lecture 2 - Data Mining' ;
X = 'Lecture 4 - Statistics' ;
...
X = 'Lecture 3 - Geometry' ;
```

Even if we do not go into the numerical values dictated by the calculation of similarities, we can guess the reasons behind why some LOs are suggested before others. The first is "Lecture 3 - Computer Networks", which, as we can see, presents the tags "computer_science", "computer_networks" and "algorithm" which are all present among the tags of the LOs that the user has already visited. The next two LOs, on the other hand, also contain the tag "cybersecurity", which is not present among the tags of the LOs seen. The order, however, is not indifferent. One of the read LOs has the tags "computer_science", "computer_networks" and "protocols". The second and third suggested LOs have the same tags except for the last one, as "computer_networks" appears in the second and "protocols" in the third. Since the tags are placed in order of specificity, more importance is given to "computer_networks" since it appears as the second tag, and therefore the second LO takes precedence over the third. Finally, proceeding rapidly downwards, we find more and more LOs with fewer (relevant) tags in common with those present in the LOs visited. The last suggested LO does not contain any tags present among the tags of the visited LOs, but contains "statistics" which is related to the LO of the IA lesson.
After running a few experiments and asking the sample users to rate the recommendations, we may preliminarily conclude that the set of suggested Learning Objects were generally considered satisfactory and useful. Since the recommended lectures are ordered by relevance, the users confirmed that the very first LOs are more closely related to what they had already studied, and represent good paths to expand their knowledge in those directions.

VI. Related Works

Today, open educational resources such as MIT's Open Courseware [2] provide free curated content [30] and location-aware search engines may offer course details in one's subject area. However, the realisation of truly personalised learning through technology is still to come. Many branches of AI are sufficiently mature to take significant steps towards this goal, but have only partially used for supporting educational purposes, as we quickly review in the following. In this work, much emphasis should be placed on feedback as a way of learning or fine-tuning the recommendation strategy. As they are also expressed in natural language, it may be necessary to extract significant parts from the text and insert new information into the graph automatically. [27] proposes a method for automatic extraction and insertion of triples in an RDF triplestore. In our context, we do not have tweets but feedback and we do not want to insert triples but enrich the database. Much research investigated LO recommendation [32], but usually following the same approach as commercial applications, and thus typically ignoring the learners' unique features and context, such as their background, history and preferences [41]. Some works tried to learn learners profiles specifically (e.g., [29]), and will be the basis for developing our profile-based recommendation approaches. Also, recommendations have been unable to guide the learners along the path necessary to attain their personally chosen goals as compared to goals specified by a teacher or school curriculum. Among the many proposals, only a few use ontologies to describe LOs, user profiles, and context information together.
Several works considered the use of ontologies in the ed-

Table 1: Fragment of subjects' taxonomy

Computer Science	Database	SQL
		Relational_model
		NoSql
		Graph
	Programming	C
		Java
		Python
	AI	Machine_learning
		Neural_networks
		Expert_systems
	Algorithm	Complexity
	Network	Protocol
	Security	Cryptography
Medicine	Anatomy	Tissue
		Articulation
		Blood
	Pathology	Virus
		Bacteria
		Toxins
	Pharmacology	Antibiotics
		Antihistamines
		Anti-inflammatories
Mathematics	Statistics	Mean
		Variance
		Median
	Analysis	Line
		Parabola
		Hyperbola
	Geometry	Triangle
		Square
		Cube
History	Roman_history	Roman_empire
	Modern_history	French_revolution
	Contemporary_history	World_War_II

ucational context, but mostly focusing on LOs only. [21] uses domain ontologies to annotate LO content, and content structure ontologies to enable direct access to LOs' components. [43] investigates the interoperation of learning content defined according to different specifications. More recently, [25] defines the basis for cross-repository semantics and proposes semantically enhanced characterization of LOs within a digital repository, so as to increase discoverability of its resources. Other initiatives, such as [26], rely on existing metadata schemes, e.g. Learning Object Metadata and Dublin Core. Some projects tried to expand the focus to other issues (e.g., the social aspects of education in IntelLEO[8]). While taking into account all these efforts, KEPLAIR further expands the ontology in several directions, especially concerning the context and subjective aspects that are neglected by other approaches. In the management of knowledge bases, it is assumed that they will be modified during use both in instances but sometimes also in schemas. For this reason, it may be useful to manage the versions of the knowledge base. [3] proposes a mechanism for managing, storing and querying different versions of the same knowledge base. Among less explored directions, social networking specifically oriented to educational purposes has been proposed in [4]. Concerning interaction between learners and the system, conversational agents have been proposed to answers questions asked by the learners to improve their learning path [38], and to provide suggestions on educational material [18]. Switching from the virtual to the physical context, [10] describes a multiagent context-aware system using IoT technology to allow understanding and interaction in a smart environment.

Logic Programming integrates easily with databases: Datalog [7] represents the meeting point between relational database manipulation languages and logic programming languages. Over the years, database technologies have increased, but the need to transfer data in the form of Knowledge Bases is ever greater. There are a number of works, such as [6], that have tried to translate graphs into a formalism based on first-order logic, in order to obtain inference of a different nature.

[20] proposed a task similar to ours, using LP for e-learning. However, as similar as the methodology is, their aim was to suggest LOs to users who could confirm or dismiss the recommender system proposal. In our case, we want to propose a broader system that takes into account many more factors and allows for the creation of fully personalised learning paths.

On the other hand, the potential of (First-Order Logic) reasoning in building recommender systems has been explored in application fields different than education. E.g., [8] uses a framework for argumentation (one of the possible types of reasoning) for the resolution of queries on search engines.

Concerning emotion recognition, whilst much research exists on standard emotions, such as joy, fear, etc., investigation on emotional statuses more relevant to education, such as engagement, stress, boredom, etc., is still an open issue. Although in a completely different domain, [46] proposes a method to unify ontologies and SWRL rules for risk management in the field of Project Management. Unlike our ap-

[8]http://www.intelleo.eu/ontologies/activities/spec/

```
student('davide').
course(c1, 'Database').
course(c2, 'Programming').
course(c3, 'Artificial Intelligence').
course(c4, 'Programming Languages').
course(c5, 'History').
course(c6, 'Computer networks').
course(c7, 'Data Mining').
learningObject(lo1, 'Lecture 1 - Database').
learningObject(lo2, 'Lecture 1 - Programming').
learningObject(lo3, 'Lecture 1 - IA').
learningObject(lo4, 'Lecture 1 - Programming Languages').
learningObject(lo5, 'Lecture 1 - History').
learningObject(lo6, 'Lecture 1 - Computer Networks').
learningObject(lo7, 'Lecture 1 - Data Mining').
learningObjectTagList(lo1, [computer_science, database, sql]).
learningObjectTagList(lo2, [computer_science, programming, algorithm]).
learningObjectTagList(lo3, [computer_science, ai, statistics]).
learningObjectTagList(lo4, [computer_science, algorithm, complexity]).
learningObjectTagList(lo5, [history, roman_history, julius_caesar]).
learningObjectTagList(lo6, [computer_science, computer_networks, protocols]).
learningObjectTagList(lo7, [computer_science, machine_learning, database]).
hasLearningObject(c1, lo1).
hasLearningObject(c2, lo2).
hasLearningObject(c3, lo3).
hasLearningObject(c4, lo4).
hasLearningObject(c5, lo5).
hasLearningObject(c6, lo6).
hasLearningObject(c7, lo7).
hasRead('davide', lo1).
hasRead('davide', lo2).
hasRead('davide', lo3).
hasRead('davide', lo4).
hasRead('davide', lo5).
hasRead('davide', lo6).
hasRead('davide', lo7).
```

Figure. 4: Fragment of facts used by KEPLAIR's recommendation engine

proach, it does not start from the perspective of databases and is therefore useful in a context where it is possible to express as many rules as one considers necessary.

VII. Conclusions

In closing, KEPLAIR is a learning-centric ITS designed to act as a personalised tutor that helps learners find the tools, resources, experiences, and connections they need to learn exactly what they have chosen to learn. It pervasively uses symbolic AI to carry out its tasks. In this paper, we proposed and illustrated the logical architecture and functions of the system, with a specific focus on its AI engine and on the ontology that acts as a data schema that coordinates and drives all of the system's functions, activities and modules. We also described the first component actually implemented in KEPLAIR: a recommender module for learning objects (courses and materials). It leverages First-Order Logic reasoning, based on graph-based algorithms and numeric similarity computation, to deliver contents tailored to the learners' educational goals, prior knowledge, personal interests and preferences, physical and cognitive abilities, and social, financial and geographic contexts. Albeit in its first prototype implementation, the recommender provided relevant and valuable suggestions to sample users in an initial controlled experiment. The overall implementation of KEPLAIR is underway, and will require collaboration from many different people and groups, contributing with different skills and perspectives. Concerning the recommendation functionality described in this paper, further experiments are being carried out, to obtain a quantitative evaluation. Also, the current recommendations do not include any serendipitous component, which we consider to be fundamental in learning, to avoid the user getting stuck in a 'filter bubble' [36]. We are currently working in this direction, so that future recommendations may also include LOs which are not linked to the extracted features of a user.

References

[1] IEEE Standard for Learning Object Metadata. *IEEE Std 1484.12.1-2002*, pages 1–40, 2002.

[2] H. Abelson. The Creation of OpenCourseWare at MIT. *Journal of Science Education and Technology*, 17:164–174, 2008.

[3] L. Bayoudhi, N. Sassi, and W. Jaziri. Towards a semantic querying approach for a multi-version owl 2 dl

ontology. *International Journal of Computer Information Systems and Industrial Management Applications*, 13:080–090, 2021.

[4] M. Biasini, V. Carmignani, N. Ferro, P. Filianos, M. Maistro, and G. M. Di Nunzio. FullBrain: a Social E-learning Platform. In *17th Italian Research Conference on Digital Libraries (IRCDL)*, volume 2016 of *CEUR Workshop Proceedings*, pages 25–41. CEUR-WS.org, 2021.

[5] J. R. Carbonell. AI in CAI: An artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems*, 11:190–202, 1970.

[6] R. Catherine and W. Cohen. Personalized Recommendations Using Knowledge Graphs: A Probabilistic Logic Programming Approach. In *Proceedings of the10th ACM Conference on Recommender Systems*, RecSys '16, page 325–332. Association for Computing Machinery, 2016.

[7] S. Ceri, G. Gottlob, and L. Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166, 1989.

[8] C. I. Chesnevar, A.G. Maguitman, and G. R. Simari. A first approach to argument-based recommender systems based on defeasible logic programming. In James P. Delgrande and Torsten Schaub, editors, *10th International Workshop on Non-Monotonic Reasoning (NMR 2004), Whistler, Canada, June 6-8, 2004, Proceedings*, pages 109–117, 2004.

[9] M. Deverashetti and S. K. Pradhan. Identification of topologies by using harmonic centrality in huge social networks. In *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, pages 443–448. IEEE, 2018.

[10] F.R. dos Santos and P. Notargiacomo. Intelligent Educational Assistant based on Multiagent System and Context-Aware Computing. *International Journal of Advanced Computer Science and Applications*, 9, 2018.

[11] F. Esposito, N. Fanizzi, S. Ferilli, T.M.A. Basile, and N. Di Mauro. Multistrategy Operators for Relational Learning and Their Cooperation. *Fundam. Inf.*, 69(4):389–409, dec 2006.

[12] S. Ferilli. Integration Strategy and Tool between Formal Ontology and Graph Database Technology. *Electronics*, 10(2616), 2021.

[13] S. Ferilli, B. De Carolis, and D. Redavid. An Intelligent Agent Architecture for Smart Environments. In *Foundations of Intelligent Systems - 22nd International Symposium, ISMIS 2015, Lyon, France, October 21-23, 2015, Proceedings*, volume 9384 of *Lecture Notes in Computer Science*, pages 324–330. Springer, 2015.

[14] S. Ferilli and F. Esposito. A Logic Framework for Incremental Learning of Process Models. *Fundamenta Informaticae*, 128:413–443, 2013.

[15] S. Ferilli and L. Loop. Toward Reasoning-based Recommendation of Library Items – A Case Study on the e-Learning Domain. In *Proceedings of the 18th Italian Research Conference on Digital Libraries (IRCDL 2022)*, Central Europe (CEUR) Workshop Proceedings, 2022.

[16] S. Ferilli, L. Loop, W. Rankin, and P. Trafford. KE-PLAIR – A Platform for Independent Learners. In *13th International Conference on Education and New Learning Technologies (EDULEARN)*, pages 9638–9647. IATED, 2021.

[17] S. Ferilli and D. Redavid. The GraphBRAIN System for Knowledge Graph Management and Advanced Fruition. In *Foundations of Intelligent Systems - 25th International Symposium, ISMIS 2020, Graz, Austria, September 23-25, 2020, Proceedings*, volume 12117 of *Lecture Notes in Computer Science*, pages 308–317. Springer, 2020.

[18] V. Fernoagă, G. Stelea, C. Gavrilă, and F. Sandu. Intelligent Education Assistant Powered by Chatbots. In *14th International Scientific Conference eLearning and Software for Education*, volume 2, pages 376–383, 2018.

[19] A. Gardner, J. Kanno, C. A. Duncan, and R. Selmic. Measuring distance between unordered sets of different sizes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 137–143, 2014.

[20] A. Garro, N. Leone, and F. Ricca. Logic Based Agents for E-Learning. In *Proceedings of the Workshop on Knowledge Representation and Automated Reasoning for E-Learning Systems (18th International Joint Conference on Artificial Intelligence (IJCAI 2003))*, pages 36–45, 2003.

[21] D. Gasevic, J. Jovanovic, and V. Devedzic. Ontology-based annotation of learning object content. *Interact. Learn. Environ.*, 15(1):1–26, 2007.

[22] K. J. Horadam and M. A. Nyblom. Distances between sets based on set commonality. *Discrete Applied Mathematics*, 167:310–314, 2014.

[23] IFLA Study Group on the FRBR. Functional Requirements for Bibliographic Records – Final Report. Technical report, International Federation of Library Associations and Institutions, 2009.

[24] M. Kardas and E. O'Brien. Easier Seen Than Done: Merely Watching Others Perform Can Foster an Illusion of Skill Acquisition. *Psychological Science*, 29:521–536, 2018.

[25] D.A. Koutsomitropoulos, A.D. Andriopoulos, and S.D. Likothanassis. Semantic Classification and Indexing of Open Educational Resources with Word Embeddings and Ontologies. *Cybern. Inf. Technol.*, 20(5):95–116, December 2020.

[26] D.A. Koutsomitropoulos and G.D. Solomou. A learning object ontology repository to support annotation and discovery of educational resources using semantic thesauri. *IFLA journal*, 44:4–22, 2018.

[27] N.S. Kumar and M. Dinakaran. Semantic filtering and event extraction of twitter streams through rdf and sparql. *International Journal of Computer Information Systems and Industrial Management Applications*, 10:208 – 218, 2018.

[28] F. Leuzzi and S. Ferilli. A Multi-Strategy Approach to Structural Analogy Making. *J. Intell. Inf. Syst.*, 50(1):1–28, feb 2018.

[29] O. Licchelli, T. M. A. Basile, N. Di Mauro, F. Esposito, G. Semeraro, and S. Ferilli. Machine Learning Approaches for Inducing Student Models. In *Innovations in Applied Artificial Intelligence*, pages 935–944, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[30] T. Malloy and G. Hanley. MERLOT: A faculty-focused Web site of educational resources. *Behavior Research Methods, Instruments, & Computers*, 33:274–276, 2001.

[31] D. Muller. *Designing Effective Multimedia for Physics Education*. PhD thesis, University of Sydney, June 2008.

[32] A.H. Nabizadeh, J.P. Leal, H.N. Rafsanjani, and R.R. Shah. Learning path personalization and recommendation methods: A survey of the state-of-the-art. *Expert Systems with Applications*, 159:113596, 2020.

[33] P. Ocheja, B. Flanagan, H. Ueda, and H. Ogata. Managing lifelong learning records through blockchain. *Research and Practice in Technology Enhanced Learning*, 14:1–19, 2019.

[34] K. Okamoto, W. Chen, and X. Li. Ranking of closeness centrality for large-scale social networks. In *International workshop on frontiers in algorithmics*, pages 186–195. Springer, 2008.

[35] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, Brisbane, Australia, 1998.

[36] E. Pariser. *The filter bubble: How the new personalized web is changing what we read and how we think*. Penguin, 2011.

[37] I. Robinson, J. Webber, and E. Eifrem. *Graph Databases*. O'Reilly Media, 2nd edition, 2015.

[38] S. Rouhani. Development and Evaluation of Intelligent Agent-Based Teaching Assistant in e-Learning Portals. *International Journal of Web-Based Learning and Teaching Technologies*, 10:11, 2015.

[39] R.M. Ryan and E.L. Deci. Intrinsic and extrinsic motivation from a self-determination theory perspective: Definitions, theory, practices, and future directions. *Contemporary Educational Psychology*, 61:101860, 2020.

[40] S. E. Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

[41] J.K Tarus, Z. Niu, and G. Mustafa. Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. *Artificial intelligence review*, 50(1):21–48, 2018.

[42] K. Thiel and M. R. Berthold. Node similarities from spreading activation. In *2010 IEEE international conference on data mining*, pages 1085–1090. IEEE, 2010.

[43] K. Verbert and E. Duval. ALOCOM: a generic content model for learning objects. *Int. J. Digit. Libr.*, 9(1):41–63, 2008.

[44] D. R. White and S. P. Borgatti. Betweenness centrality measures for directed graphs. *Social networks*, 16(4):335–346, 1994.

[45] T.T. York, C.W. Gibson, and S. Rankin. Defining and measuring academic success. *Practical Assessment, Research and Evaluation*, 20:1–20, 2015.

[46] W. Zaouga and L.B.A. Rabai. A Decision Support System for Project Risk Management based on Ontology Learning. *International Journal of Computer Information Systems and Industrial Management Applications*, 13:113 – 123, 2021.

[47] J. Zhan, S. Gurung, and S. P. K. Parsa. Identification of top-k nodes in large networks using katz centrality. *Journal of Big Data*, 4(1):1–19, 2017.

[48] P. Zhao, C. Aggarwal, and G. He. Link prediction in graph streams. In *2016 IEEE 32nd international conference on data engineering (ICDE)*, pages 553–564. IEEE, 2016.

Author Biographies

Stefano Ferilli, PhD, born 1972, Laurea degree in Computer Science (1996), Ph.D. in Computer Science (2001), Specialistic Laurea degree in Computer Science (2003), qualifying examination as an Engineer in 2005. Since 2002 Assistant Professor, and since 2006 Associate Professor in Computer Science at the University of Bari, where he serves as the Director of the Inter-departmental Center for research on Logic and its Applications. His scientific interests are centered on Expert Systems and on Knowledge Representation and Acquisition, with particular reference to Incremental and Multistrategy Learning using First-Order Logic formalisms. He is the author of a book on Document Processing published by Springer, 2 textbooks and more than 320 scientific papers published on National and International journals, books and conferences/workshops proceedings. He served in the Program Committee of major National and International events, and as an Editorial Board member of international journals.

Domenico Redavid, PhD, born 1972, Laurea degree in Computer Science (2002), Ph.D. in Computer Science

(2008). Assistant professor in Computer Science at the University of Bari from 2019. His scientific interests include Semantic Web Services and Semantic Management of Business Processes with the application of Description Logic Reasoning. Since from 2003 he has participated in several research projects in the areas of Semantic Web and Machine Learning applied in different application domains. He has been active in technology transfer as CEO of Artificial Brain S.r.l. for over 10 years. He is author of more than 50 scientific papers and reviewer to international journals, conferences and workshops.

Davide Di Pierro, PhD student, born 1997, Bachelor's Degree in Computer Science and Software Production Technologies (2019), Master's Degree in Computer Science curriculum Knowledge Engineering and Machine Intelligence (2021). His current scientific interests are centered on Artificial Intelligence, with particular reference to Knowledge Representation and Reasoning, using First-Order Logic formalisms.

Liza Loop, born 1945 in Boston, MA, USA. B.A. in Philosophy, Sonoma State University (1969), M.A. in Design and Evaluation of Educational Programs, Stanford Graduate School of Education (1989), and various certificates in educational teaching, testing and diagnosis. In 1975, Loop founded LO*OP Center, Inc., a California nonprofit corporation that provides educational products, services, consulting, research and technical resources relating to child and adult learning, history and future of computing in learning and education, and intercultural communications. Loop has served continuously as Executive Director of this organization and has personally conducted research on educational applications for numerous companies including Atari, Apple, Commodore, Personal Software (VisiCorp), People's Computer Company, Sun Micro Systems and institutions such as Stanford University and the Community College Consortium for Open Educational Resources. She is co-author of the book, ComputerTown, USA, has written user's manuals for many personal computer software applications, and presented at conferences on education, computing, and intercultural communications.