Toward the application of artificial intelligence in academic content: An autonomous recommendation system

DOI: https://doi.org/10.17230/9789587207002ch1

Edwin Montoya-Múnera, Jose Aguilar, Julián Alberto Monsalve-Pulido, Camilo Salazar, Daniela Varela-Tabares, Marvin Jiménez-Narváez, Edwin Montoya-Jaramillo

The latest generation of new information and delivery channels is being produced at accelerated rates. This has made it nearly impossible to keep various courses updated (despite human intervention), which can limit the quality of virtual education and generate a systematic delay/gap in their learning processes. Thus, this project developed a system that uses big data analytics and artificial intelligence in order to gather information and knowledge from digital resources on the Internet, and recommend them to students and teachers in either blended or virtual courses. In this case, the digital resources mainly included texts and unstructured data (e.g., patents, articles, books, Wikis, etc.), audio and/or video, all of which can be difficult for processing and extracting value, information, and knowledge. It is important to note that such digital resources also support various courses by using tags, descriptions, keywords, logs, profiles, preferences, etc. Overall, the purpose of this project is to provide students and teachers with recommendations for the contents that best match their profiles and learning progress within each course.

This project, called Smart Contents (SmartCon), integrated many free sources and digital contents from the Internet in order to build a smart system that recommends the contents to different users in an educational context, based on big data analytics, autonomic computing, and artificial intelligence paradigms. This project included the following components: 1) identifying open, high-quality digital resources such as patents, articles, books, etc.; 2) harvesting these sources into a data lake; 3) processing and transforming these sources for the analytics stage; 4) building analytics models to enrich the knowledge of students and teachers as well as the contents of courses; 5) developing a search engine for all contents

and knowledge, for further retrieval by the recommendation system; 6) defining the recommendation system, the main component of SmartCon, to personalize the contents to different users (e.g., students and teachers); 7) developing SmartLMS, an open-source learning management system (LMS) that uses the recommendation system through different plug-ins and components.

This chapter also describes the different components of the SmartCon project. Particularly, it presents the design of the system's main aspects, including the general architecture of the autonomic and the intelligent recommendation system as well as the hybrid and emotional extensions. It also presents examples of some of the mining tasks developed during the project as well as the main characteristics of the developed prototype. More details regarding the methodologies, experiments, and results, etc. have been presented in various works published during this project (Aguilar, Salazar, Velasco, Monsalve-Pulido, & Montoya, 2020; Jimenez, Aguilar, Monsalve-Pulido & Montoya, 2020; Monsalve-Pulido, Aguilar, Montoya, & Salazar, 2020; Salazar, Montoya & Aguilar, n.d.; Salazar, Aguilar, Monsalve-Pulido, & Montoya, 2020; Salazar, Aguilar, Monsalve-Pulido & Montoya, n.d.; Varela, Aguilar, Monsalve-Pulido & Montoya, n.d.-a,b). An additional goal of this chapter is to present all of these works and show how they are coherently integrated.

The remainder of this chapter is organized as follows. Section 2 describes the related works with SmartCon and the domains that are covered by this project, while Section 3 presents the evolution of the architecture for SmartCon. Section 4 describes the various mining tasks conducted during the project in order to extract knowledge for the recommendation system, while Section 5 explains the proof of concept or the prototype of SmartCon as well as some actual courses that employed this prototype. Finally, Section 6 presents the main contributions, conclusion, and future recommendations.

Related Works

Recommender systems have been widely studied in the literature, due to their vast array of applications. In fact, there are many applications for the recommendation of products, movies, news, academic resources,

etc. (Adomavicius & Tuzhilin, 2005). In the literature, recommender systems have been classified into four main categories, according to how the recommendation is made: 1) content-based recommendations (CB); 2) collaborative filter recommendations (CF); 3) intelligent recommendations; and 4) hybrid recommendations (Balabanovic & Shoham, 1997). In previous research (Aguilar, Valdiviezo-Díaz, & Riofrio, 2017), the authors developed a general framework for an intelligent recommendation system integrating the processes of learning, inference, etc. This system consisted of the following components: knowledge modeling, learning methods, and reasoning mechanisms.

According to Shardanand & Maes (1995), the systems that perform content-based recommendations include some limitations. For example, they are not very efficient at recognizing the differences between two items of diverse qualities, even if the items include a large number of words in common. This makes these systems somewhat poor at evaluating item quality, since they significantly depend on users' perceptions. Additionally, Shardanand & Maes confirmed that it is not easy to find items that are apparently not of interest to users, but are actually good enough to be recommended. These types of problems are not frequent in collaborative filtering systems, since they are not based on the contents of the items, but on other users' opinions about the recommended item (Cacheda, Carneiro, Fernández, & Formoso, 2011). In this regard, a user's profile is based on the ratings given to the items. This feature also allows the system to be able to recommend items without analyzing their contents, thus making it useful to recommend any type of element. An example of this can be found in the cases of Ringo (Shardanand & Maes, 1995) and Video Recommender (Hill, Stead, Rosenstein, & Furnas, 1995), which are e-mail and web-based systems for recommending music and movies, respectively.

As for the recommendation systems based on collaborative filtering, they also present problems such as scalability, complexity of their models, sensitivity to data changes (Cacheda et al., 2011), sparsity of the rating matrix (Huang, Chen, & Zeng, 2004; Sarwar, Karypis, Konstan, & Reidl, 2001), cold start (Schein, Popescul, Ungar, & Pennock, 2002), shilling (Chirita, Nejdl, & Zamfir, 2005; Lam & Riedl, 2004), etc. Due to these issues, many recommending systems use a hybrid approach that mixes content-based and collaborative filtering methods. In particular, this can

minimize the problems that occur when they utilize a single approach. Examples of hybrid recommender systems exist in the academic world because the aim is to personalize the teaching-learning process. A general architecture for a hybrid recommendation system, which uses effort-based learning to improve quality over time, is proposed in Golovin & Rahm (2004). In the application of the recommendation algorithms, the authors included context variables such as content, user, and time. Another proposal of a hybrid architecture, which recommends open courses and educational resources, is presented in Vladoiu, Constantinescu, & Moise (2013). This architecture created a combination of two types of recommendations: one based on enhanced cases (guided by a quality model), and another based on user feedback (collaborative).

In addition, an interesting example of a recommendation system architecture in the academic arena is the one proposed in Zhu, Ip, Fok, & Cao (2008), where a recommendation methodology based on multiple hierarchical intelligent agents is presented. This methodology performed static and dynamic modeling of the users, and offered several functions, including the generation and adjustment of learning plans, personalized recommendations, and learning progress assessments in real time.

One of the most important elements in teaching-learning processes is the construction of knowledge based on collaboration. For example, in Knob, Esteves, Granville, & Tarouco (2017), a multi-agent, client-server application architecture is proposed to recommend different types of activities. More specifically, this architecture considers the set of functionalities in the application and the operations necessary to access them for the exchange of knowledge in virtual communities. Such functionalities are then used by the personal agents in Android, who execute their tasks while achieving their individual and collective objectives. This proposed architecture has been mainly used in the context of smart cities, helping them achieve the objectives of decentralization for the management of communities.

Moreover, educational systems can take advantage of the emotional state of students to enhance their learning processes, as evidenced by numerous investigations. Some studies have even built emotional-aware learning systems, and compared their performance to non-emotional-aware learning systems in order to observe any improvements in students' academic performance (Faria et al., 2017; Pekrun, 1992; Shen, Wang, & Shen, 2009).

Other authors have analyzed the correlation between emotional features and the evaluations of students, thus highlighting the relationship between emotions and learning performance (Chauhan, Agrawal, & Meena, 2019; Immordino-Yang & Damasio, 2007; Yu et al., 2018).

In a related study, Shen et al. (2009) found a 91% increase in e-learning performance by using emotional data. This increase was especially observed in user-centered learning. They also noted the lack of research in detecting emotions during the learning process in real time. Meanwhile, Pekrun, Goetz, Titz, & Perry (2002) and Yu et al. (2018) noted that positive emotions can promote self-regulation among students, whereas negative emotions can lead to dependence on external orientation. In general, these studies assume that not all emotions are relevant to learning, but only a small subset of them, with different investigations proposing several emotions related to learning. For example, Shen et al. (2009) focused on four emotions and observed a 91% increase in e-learning performance by only analyzing this subset of emotions, compared to a system with no emotion analysis. It is also important to consider that the type of education (e.g., self-learning, classroom lectures, group discussions, etc.) implies non-identical processes and requires different considerations.

Finally, several works have proposed intelligent recommender systems, some of them in the e-learning domain. For instance, Tarus, Niu, & Mustafa (2018) reviewed literature on ontology-based recommenders for e-learning. They also categorized the different recommendation techniques used in ontology-based, e-learning recommenders, according to the knowledge representation technique, ontology type, and ontology representation language in ontology-based recommender systems, in addition to the types of learning resources recommended by e-learning recommenders. Obeid, Lahoud, El Khoury, & Champin (2018) presented an approach for developing an ontology-based recommender system, with improved machine learning techniques, to orient higher education students. The main objective of their ontology-based recommender system was to identify students' requirements, vocational strengths and weaknesses, interests, preferences, and capabilities to recommend the appropriate major and university for each one. Finally, Vijayakumar, Vairavasundaram, Logesh, & Sivapathi (2019) presented a new travel

recommendation system employed on a mobile device that generates personalized travel planning, including multiple points of interest (POIs). This personalized list of recommended travel destinations was based on a heat map of previously visited and highly relevant POIs.

Architecture

This section presents the architecture of the recommendation system for virtual learning environments (VLEs) proposed by Monsalve-Pulido et al. (2020). This architecture includes four sections, beginning with a general overview of the architecture, followed by a specific description of the autonomous recommendation system. Next, the hybrid recommendation component is presented in detail, after which its extension as an affective recommendation system is described.

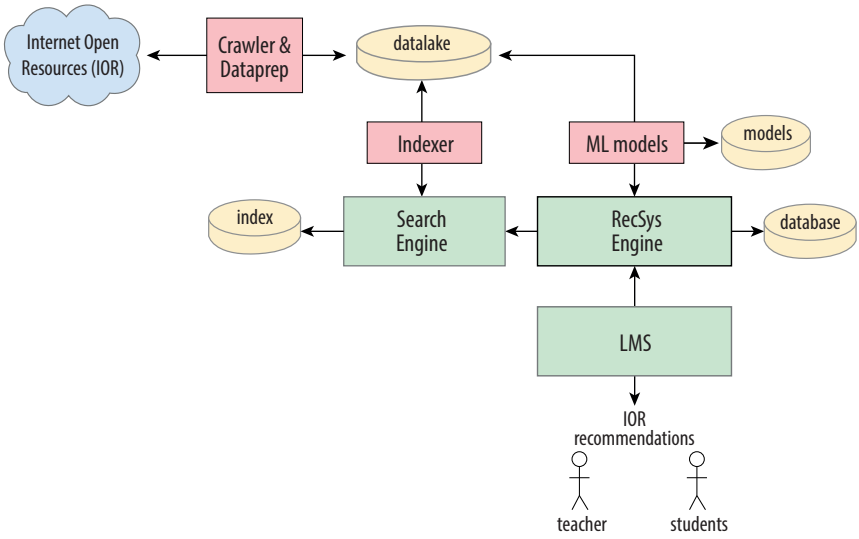
General Architecture

SmartCon is a project that harvests millions of free and open digital resources on the Internet or "Internet Open Resources" (IORs), which are stored in a data lake for further processing. In the harvesting stage, the sources, crawling, and processing of the raw data are identified in a normalized manner. Next, the data is processed through a search engine and a machine learning (ML) processor, after which it is used as an indexer and for ML techniques. In this case, the search engine is an implementation of the first version of the recommendation system, which is based on the data contents (i.e., the RecSys).

Overall, the ML processor follows three objectives: 1) to process the data and enrich the metadata indexed in the search engine; 2) to generate new data and models to improve the precision of the recommendations; and 3) to implement new features, such as clustering, top-n-related data, and collaborative filtering, incorporated into the RecSys Version 2 (i.e., smart and collaborative filtering RecSys). Meanwhile, the RecSys engine performs three tasks: 1) it pre-calculates a recommendation for each course-student pair; 2) it receives the requests from the LMS and returns the recommendations; and 3) it processes logs, favorites, students, and course profiles. Finally, the LMS is a learning environment in which

students and teachers access courses, including contents and activities, and receive recommendations from SmartCon. Figure 1 presents the general architecture of SmartCon.

Figure 1. The General Architecture of SmartCon

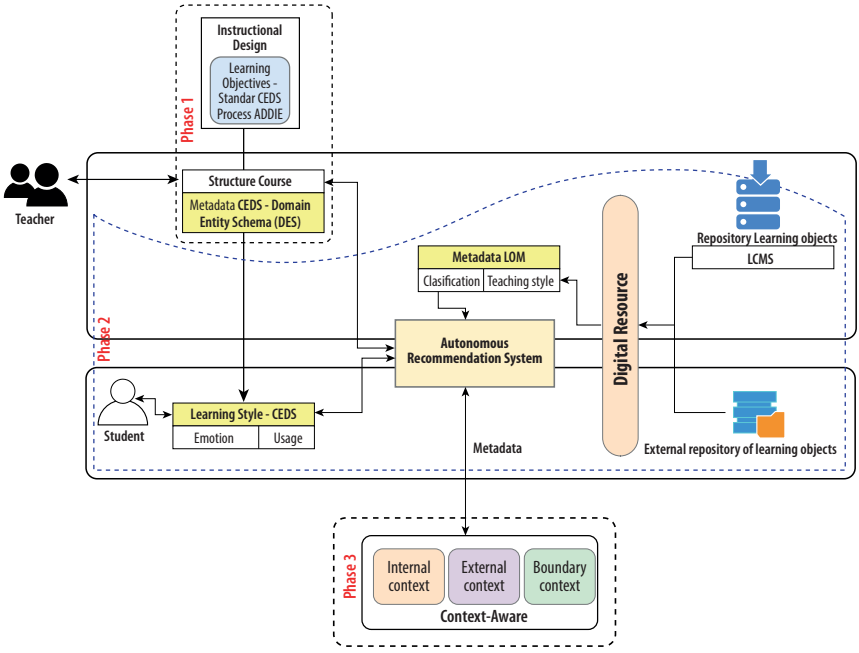


Source: Prepared by the authors

Autonomous Recommendation System Architecture

This section presents the architecture of the autonomous recommendation system (ARS) proposed by Monsalve-Pulido et al. (2020). This architecture is based on two general principles. The first is autonomous computing using a self-managed computing approach, while the second is an intelligent recommendation system (Aguilar et al., 2017). Figure 2 describes the architecture of the ARS for teaching-learning processes in VLEs. Overall, the architecture is composed of three general phases: 1) the creation of academic courses; 2) the utilization of digital resources by students; and 3) the extraction of all of the necessary context variables in order for the architecture to recommend academic contents to students and teachers.

Figure 2. Architecture of the ARS



Source: Prepared by the authors

Table 1 describes the objective of each phase of the architecture as well as the metadata used.

Table 1. Phases of Architecture

Phase	Objectives	Metadata
Phase 1	Creation of the academic course structure and instructional design.	ADxuation (Kruse, 2002) Common Education Data Standard (CEDS) (NCES, 2014)
Phase 2	Extraction of student information (e.g., academic information–learning styles) and information from academic digital resources (e.g., internal and external repositories).	Learning Object Metadata (LOM) (of the IEEE P1484.12.2/D1, 2002). Common Education Data Standard (CEDS) (Kruse, 2002)

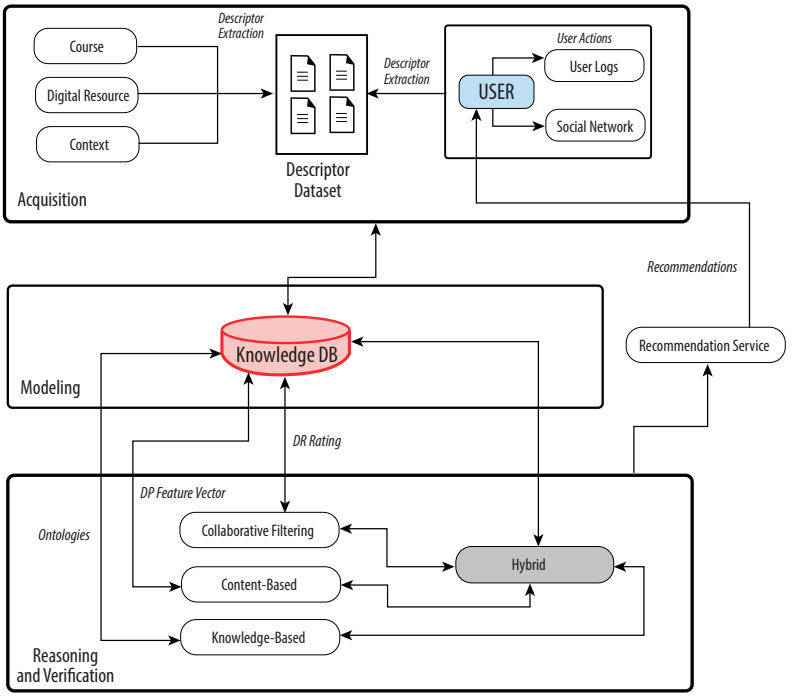
Phase 3	Extraction of context variables (e.g., internal, external and border) for the personalization of learning.	Learning Object Metadata (LOM) (of the IEEE P1484.12.2/D1, 2002). Common Education Data Standard (CEDS) (Kruse, 2002) Social networks Global Positioning System, others.

Source: Prepared by the authors

Intelligent Dimension

The ARS is defined by the intelligent dimension shown in Figure 3, which is composed of knowledge representation, learning methods, and reasoning mechanisms. Thus, the intelligent dimension of the ARS includes three main layers: 1) acquisition; 2) modeling; and 3) reasoning and verification.

Figure 3. Architecture of the Intelligent Dimension



Source: Prepared by the authors

Acquisition: In this layer, the information is extracted from two general areas. The first one is the student information generated in the teaching-learning process, which includes identifying the learning styles and all of the contextual information registered in the VLEs, social networks, connection logs, etc. The second one is the academic content, which includes extracting the information through the metadata of the educational process, including academic digital resources (e.g., books, scientific articles, learning objects, patents, etc.).

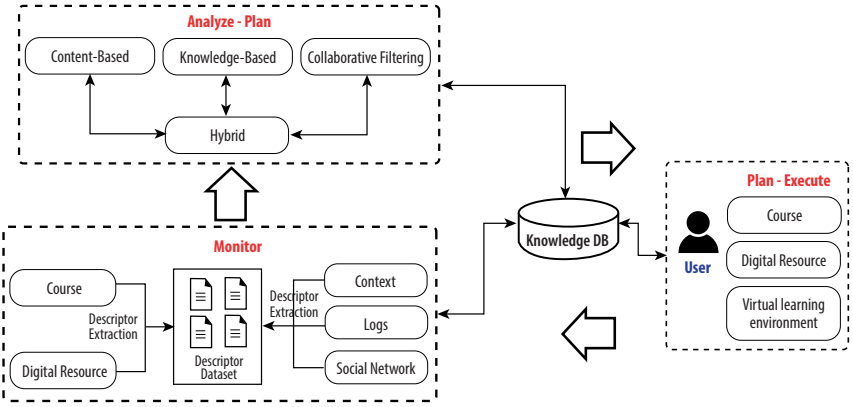
Modeling: The modeling layer stores the information from the results of the acquisition and reasoning and verification layers in a knowledge database. It begins with structured storage based on the metadata of digital resources, academic courses, students, and contexts that interact in the acquisition layer. Moreover, the recommendation results are stored in this layer by means of classification vectors, while the ontology results are from the reasoning and verification layer.

Reasoning and verification: The main objective of this layer is to effectively recommend digital academic content to teachers and students. Different reasoning mechanisms can be used in this layer (e.g., deductive, inductive or abductive). This recommendation process also uses hybrid recommendation techniques by combining the collaborative filtering, content- and knowledge-based filters.

Autonomous Dimension

The autonomic architecture of the recommendation system aims to guarantee self-management and adaptability in any context, without human intervention. In order to meet the autonomic objective, the Monitor-Analyze-Plan-Execute-Knowledge (MAPE-K) model (Vizcarrondo, Aguilar, Exposito, & Subias, 2017) was used, as described in Figure 4.

Figure 4. Architecture of the Autonomous Dimension



Source: Prepared by the authors

In Table 2, the components of the autonomous dimension are described by means of the iterative processes of the MAPE-K model.

Table 2. Components of the Autonomic Dimension

MAPE-K	Objective	Knowledge DB
Monitor	Extract the properties of the digital resources, academic courses, and context information.	Store-consult
Analysis–Planning	Process the information from the monitor stage. Run the hybrid recommendation filter.	Store and query results to estimate future recommendations
Planning–Execution	Deploy recommendations on the virtual learning platform, according to the needs of teachers and students.	Store-consult

Source: Prepared by the authors

Generic Hybrid Adaptive Architecture

Various techniques have been proposed for recommending items to users in different contexts and domains. They are mainly classified into four

approaches: 1) content-based (CB); 2) collaborative filtering (CF); 3) knowledge-based; and 4) hybrid. However, many others have emerged with the vast amount and variety of available data.

CB algorithms recommend items that match the user's preferences or profile, which is mainly defined by the items that the user has previously chosen. They also use keywords, tags or weights to characterize the objects. For example, the TF-IDF representation is a frequently used tool to obtain certain features (Aguilar et al., 2017; Burke, 2007).

CF approaches are based on the user's behavior and rating patterns in relation to other users. In general, collaborative techniques can be either memory- or model-based. The first one uses the concept of neighborhood to find similar users (or items), while the second type is derived from historical data to make predictions (Burke, 2002, 2007).

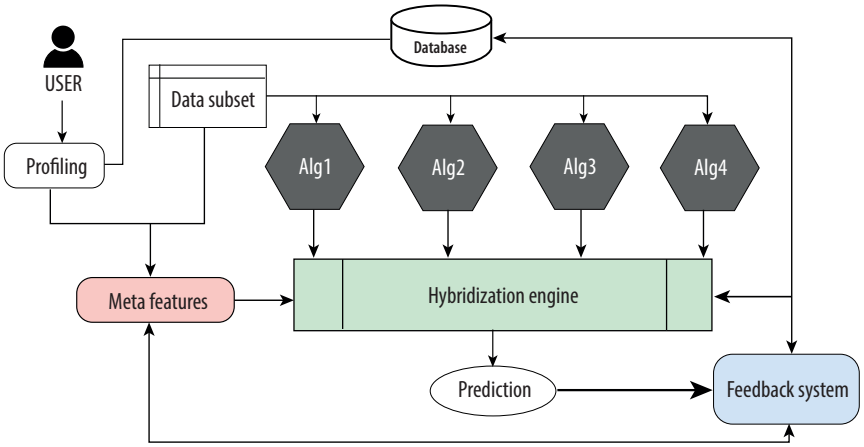
Given the strengths and weaknesses of each technique, such as the cold start problem when there is a lack of data, hybrid approaches have been widely used to improve performance by combining different types of recommendation algorithms, also known as "hybridization methods." There are many ways in which they can be combined, most of which are presented as follows (Burke, 2002, 2007).

- Weighted: The scores are numerically combined.
- Switching: The components are turned off and on, according to certain criteria.
- Mixed: The output from different recommenders are presented together.
- Feature combination: The features from different sources are combined into a single algorithm.
- Cascade: The output from one technique is used as an input feature for another.
- Meta-level: The model learned by one algorithm is used as the input for another.

Despite the improved performance over single algorithms, hybrids also present certain challenges, especially when they are implemented in actual scenarios. It is well known that data varies over time and that there is no static configuration that will optimally work for all recommendation requests. Thus, the adaptability of the system has been garnering interest, particularly as more data are emerging and more digital environments are requiring this type of service.

In Figure 5, a generic adaptive hybrid architecture is proposed, which follows the dynamic behavior of the environment through the use of metrics (i.e., meta-characteristics), from which the hybrid configuration for the recommendation is determined.

Figure 5. Generic Hybrid Adaptive Architecture



Source: Prepared by the authors

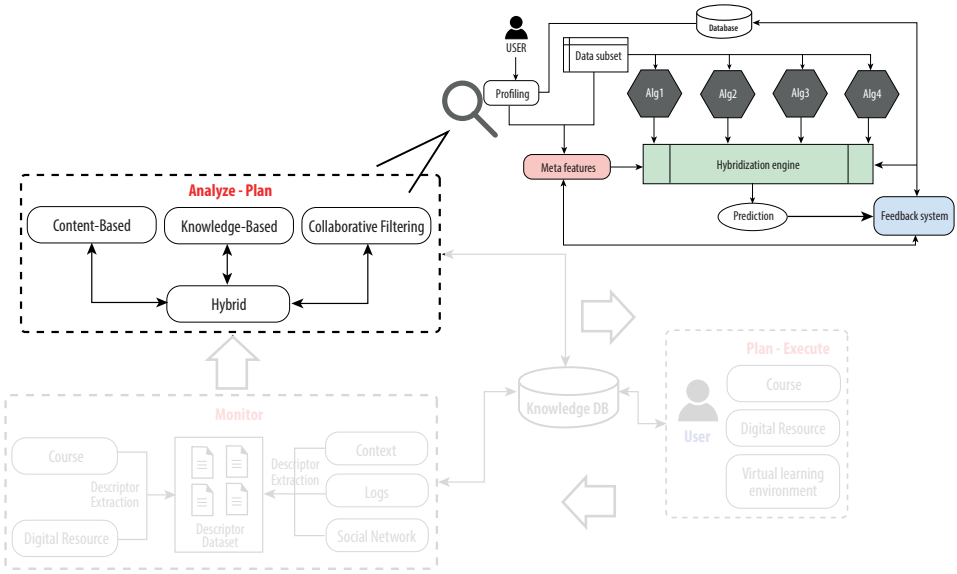
As shown in this figure, the architecture can adapt to different domains. In our case, the users would be students in a VLE. When a student enters the system for the first time, he/she completes a survey with their preferences, after which the system reduces the universe of interest for the student, since computing the subsequent processes with a large number of documents would be inefficient.

In general, the algorithms that are combined in hybrid systems should be of different types in order to take advantage of their characteristics and compensate for their weaknesses. The red block, identified as "meta-features", is a key component in this system that represents a set of numerical indicators used to describe the users (or items) at a specific moment. The idea is that the high or low values of these dynamic variables reflect multiple characteristics of the context, which can be used to configure the hybrid blend in a more optimal manner.

Meanwhile, the feedback engine is responsible for capturing the logs and other variables that allow the system to perform different tasks such as checking the usefulness of the recommendations (and potentially optimizing them), recalculating the meta-features to update the context status, and enriching the data inputs for the algorithms (mainly the collaborative-based ones).

The generic hybrid adaptive architecture is incorporated into the autonomous recommendation architecture, as shown in Figure 6.

Figure 6. Incorporation of the Generic Hybrid Adaptive Architecture into the ARS



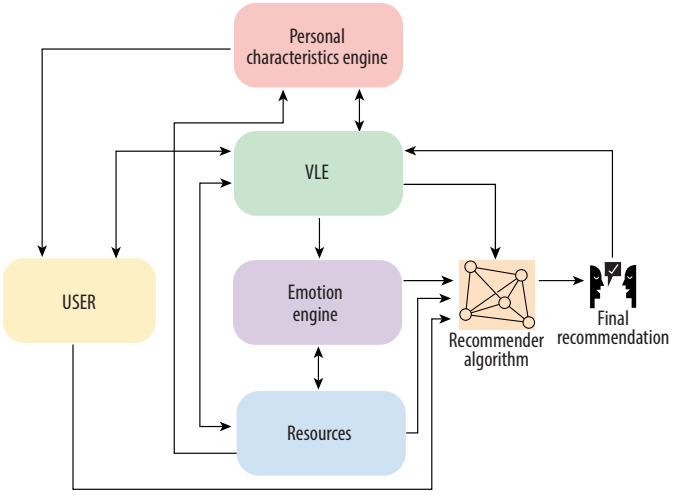
Source: Prepared by the authors

The incorporation of this component into the ARS is achieved through the MAPE-K model. More specifically, it occurs in the "Analysis–Planning" phase, in which it executes the hybrid recommendation filter (i.e., collaborative filter, based on content and knowledge), thus defining the correct combination for guaranteeing high-quality content recommendations (see Varela et al., (2020a,b) for more details).

Generic Architecture of an Affective Recommender System for e-learning Environments

The proposed architecture for an affective recommender system is presented in Figure 7. It consists of five major components: 1) user; 2) personal characteristic engine; 3) VLE; 4) emotion engine; and 5) resources. Briefly, the user component stores all of the information regarding the user's profile, while the personal characteristic engine extracts personal characteristics from the user such as personality traits and learning style. Moreover, the VLE component is the e-learning environment of the user, while the emotion engine captures (but does not store) the emotional information of the user and the course contents (i.e., the learning resources). Moreover, the resources component stores the metadata of the learning resources and the emotional logs of the user when interacting with the contents (see Salazar, Montoya & Aguilar (n.d.) for more details).

Figure 7. Generic Architecture of an Affective Recommender System



Source: Prepared by the authors

The proposed flow is as follows. A student enters the VLE and registers him/herself. During the registration process, personal information is captured, and questionnaires regarding personal traits and learning styles are completed. Additionally, the expertise level of the student can be obtained by using quizzes or questionnaires during registration. In this case, such processes are executed by the personal characteristics engine. All of this information is then stored in the user's profile, except for the expertise level, which is stored in the VLE database.

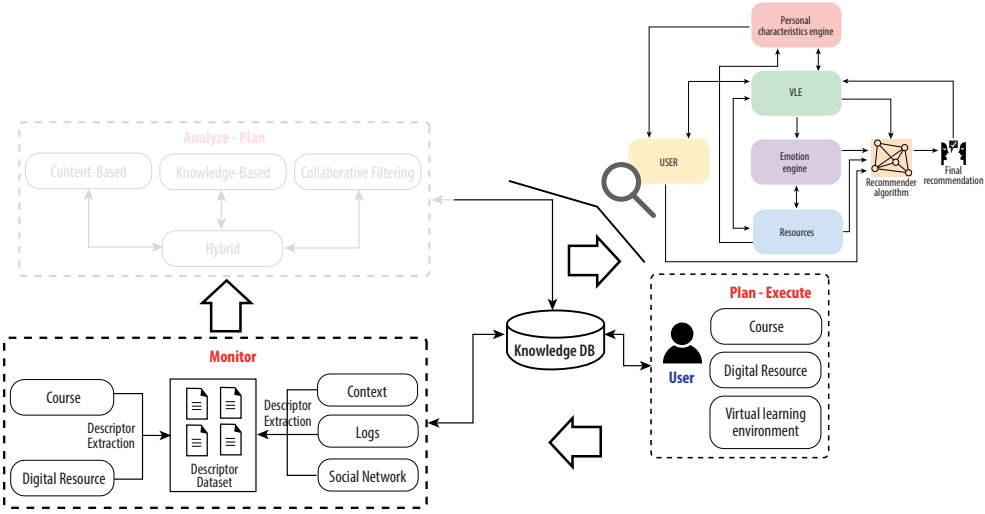
When a student is registered, he/she can log in on the platform and interact with the different contents. While the student is using the contents, several logs are captured by the VLE logger and stored in the VLE database. Meanwhile, the emotion engine captures the students' emotional information before, during, and after using the contents through multiple sources such as a camera, microphone, questionnaires, etc. Such sources are low-invasive and unobtrusive when obtaining such information during the learning process. The collected emotional information is then stored in the Resources module, in a special database. This information not only includes the emotions felt by the student in a specific course, but also some metadata of his/her interactions (e.g., timestamps).

The emotion engine is also in charge of extracting emotional information from the contents, and assigning such aspects, emotional tags/values that can be used for the recommender algorithm. Moreover, the Resources module is in control of storing all of the resource information, including the aforementioned emotional tags/values and the emotional logs from the student using the resources. Since learning styles, expertise levels, and (in some cases) personality traits are dynamic, the personal characteristic engine periodically implicitly assesses these characteristics through logs or explicitly administers questionnaires to the student in the VLE.

Finally, the recommender algorithm collects the information from the user's profile, VLE logs, emotional logs when interacting with the resources, his/her current emotional state, and the metadata of the resources in order to generate personalized content recommendations. The recommendation logs are then stored in the VLE for analyzing the performance of the recommendations, and boosting the recommender algorithm.

The generic hybrid adaptive architecture is incorporated into the autonomous recommendation architecture, as shown in Figure 8.

Figure 8. Visualization of the Emotional Architecture in the ARS



Source: Prepared by the authors

The incorporation of this component into the ARS is carried out through the MAPE-K model, specifically in the "Monitor" and "Plan–Execute" phases. In this case, the "Monitor" phase recognizes the emotions of the student, while the "Plan–Execute" phase executes the emotional recommendation filter to exploit this source of information, and to guarantee high-quality content recommendations.

Mining Tasks

Content Analysis

Content Feature Extraction

The classification of the contents was performed according to topics and keywords, using the Latent Dirichlet Allocation (LDA) technique. This topic generative model is widely used in natural language processing (NLP) for topic modeling and other tasks. It also generates K topics (K, as a tunable parameter) from the documents in a corpus (i.e., a collection of documents), and estimates two distributions: 1) the distribution of

topics in a document; and 2) the distribution of terms in a topic. In addition, this technique not only treats documents in regard to their relevance to the topic, but it also extracts two types of metadata from the contents: 1) keywords; and 2) descriptors of textual data. Moreover, the LDA technique gives a percentage regarding the membership of a document to a topic, i.e., the document is assigned to the topic with a higher percentage.

On the other hand, keywords were assigned as follows. They were extracted from the topic to which each document belonged (to a greater extent), taking the top p as more relevant in terms of the topic. In other words, for each document, the LDA technique assigned a set of K membership percentages of the specific document to each topic, after which the topic to which the document belonged the most was selected. This technique also gave a set of m (m = number of different terms in the corpus) relevance of each term to each topic. The top p relevant terms in the previously selected topic were then chosen as the keywords of the specific document. In our experimentation, p was set to 5.

Finally, two types of metadata were used for enhancing the generated recommendations. The keywords were then added to the information retrieval system. The intention was to generate keywords for all of the documents in order to generate better recommendations (see Aguilar et al. (2020) for more details).

Content Grouping

Content grouping was performed utilizing the textual descriptors (extracted with the LDA technique) to calculate similarities and generate recommendations. For extracting the textual descriptors, the set of memberships of each document to each topic was selected, thus obtaining a K-dimensional vector for each document and indicating the percentage of the membership of the document to each K topic. These vectors were primarily composed of low membership percentages, with only a few topics, presenting a considerable percentage for each document.

Moreover, the vectors representing the textual data of the documents were used to calculate similarities between them. This was helpful for providing pre-calculated recommendations for the students. For example, when a student rates the content as relevant, he/she may be interested in similar contents for learning. Thus, with the vector of a document, the

cosine similarity was used for calculating the top p most similar contents and recommending them to the student that rated the document as relevant (see Aguilar et al. (2020) for more details).

Audio Feature Extraction

In general, many academic resources found on the Internet are multimodal objects (e.g., texts, audio, images, video). Thus, it was necessary to have efficient methodologies for extracting the descriptors that allow the resources to be characterized and recommended to the students and teachers in an appropriate manner.

Traditionally, the extraction of audio descriptors consists of extracting the text content from the audio, and then performing text mining. However, there is a significant amount of audio (i.e., non-text) information contained in many learning resources (e.g., video tutorials) that is not exploited, including frequency (in Hz), loudness or sound intensity (in decibels), reverb (in sec), etc. In some cases, the audio data does not include speech. As a result, no extraction of text content is possible.

In this project, an automatic feature engineering methodology was proposed for the audio data, which can automatically extract, analyze, and select the best features for such data (Jimenez et al., 2020). In this case, different types of characteristics in the audio data were considered such as sound engineering, basic statistics, and the time-series domain. In regard to the latter, each audio sample was considered as a time-series set, since the set of variables was measured at different times, and each variable (i.e., the time-series) was characterized by a set of time-series descriptors. The proposed approach can also be developed in different ways, since various methods of exploring combinations of characteristics (e.g., genetic algorithms) and different types of evaluation functions can be used to select the characteristics, with some based on grouping or classification metrics, and others based on information theory (see Jimenez et al. (2020) for more details).

Emotion Recognition

For recognizing the emotions of the students, three sources of information (i.e., modalities), which were non-invasive or obtrusive, were used: 1) the audio from the student's speech was captured by a microphone;

2) the facial expressions were captured by a camera; and 3) the text was obtained from student's interactions such as the reviews of the contents and related chats. As for the emotion recognition process, it was performed in two phases: unimodal and multi-modal. In the first phase, the data was analyzed separately from each modality in order to obtain the recognition by each modality, while the second phase consisted of fusing the decisions from the modalities to obtain a more robust final recognition. These steps are described in the following sub-sections (see Salazar et al. (2020, n.d.) for more details).

Unimodal Phase

In this project, each modality was separately processed in the following order: 1) feature extraction; 2) feature selection; and 3) recognition. For the audio modality, 6,373 features (INTERSPEECH 2013 COMPARE feature set (Schuller et al., 2013)) were extracted from each audio sample. For the video modality, 68 facial landmarks were extracted, after which the distances between each landmark (normalized by the height of the face detected) was used as the feature for each image. Meanwhile, in order to summarize the frames in a video, the average of the frames' features in the video was used, resulting in a total of 2,278 features per video.

As for the texts, two knowledge bases were used for extracting the affective information from each word: Senticnet 5 (Cambria, Poria, Hazarika, & Kwok, 2018) and AffectiveSpace (Cambria, Fu, Bisio, & Poria, 2015). From Senticnet 5, seven features were extracted from each word, i.e., five continuous and two categorical. From AffectiveSpace, 100 continuous features were extracted for each word. Moreover, for the text, 105 continuous and two categorical features were extracted. In order to summarize the features of words in a text, the percentiles 0, 25, 50, 75, and 100 were used for continuous variables, while the summation of categories was used for each categorical feature, resulting in 441 features (i.e., 105 continuous * five percentiles + two categorical * eight classes).

When all of these features were extracted, the relevant ones were selected in the following three ways (i.e., filters): 1) removing the features with a variance lower than $1 * 10^{-4}$; 2) avoiding multicollinearity by only keeping the features with a variance inflation factor (VIF) lower than 10; and 3) conserving the characteristics with a relevance greater than

$1 * 10^{-3}$. This relevance was calculated by using a random forest model and utilizing the information it provided about the importance of the variables. At the end of the feature selection process, 131 aural, 21 facial, and 224 textual features were selected. In addition, unimodal models were constructed for each modality, with the ML techniques including support vector machines (SVM), random forest, and partial least squares (PLS) regression. Overall, PLS regression provided the best results for each modality in terms of R2 and relative error of standard deviation.

Multi-modal phase

This phase fused the features or recognitions obtained from each modality in the previous phase, and generated final robust recognition results by using the information from the three modalities. In virtual education, students are, in general, not writing or talking all of the time. Thus, audio and textual data are only available at certain moments. For this reason, the multi-modal fusion model must deal with the missing data (i.e., the missing modalities).

Overall, three approaches were proposed and compared, with two based on decision-level fusion and one based on feature-level fusion. The first approach concatenated the recognition from each modality and filled in the missing modalities with zeros, obtaining a six-dimensional vector as the input for the model. The second approach used recurrent neural networks, which varied the input length and dealt with the missing modalities. Finally, the third approach was very similar to the first, but it concatenated the extracted features and filled in the missing modalities with zeros. Moreover, the models used for the three approaches were different light architectures of neural networks, due to the possible scalability issues a VLE could face.

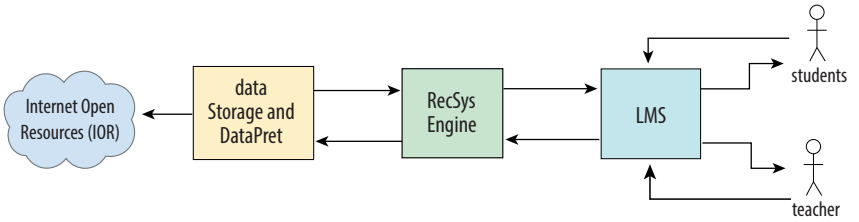
Pilot Testing

This section introduces the technological development process of SmartCon. It also describes the use of SmartCon in the hybrid and virtual courses at EAFIT University.

Design, Development, and Implementation

SmartCon is a system composed of several reference architectures that integrate big data, data analytics, artificial intelligence, and software development technologies into one product. The general architecture of SmartCon is defined by three main modules, as shown in Figure 9.

Figure 9. General Architecture of SmartCon



Source: Prepared by the authors

- *IORs:* The Internet includes many digital resources that can be used to help various learning activities. However, the main issue is how teachers or students can find appropriate content among the millions of resources. Thus, the first step is to identify the main resources on the Internet.
- *Data Storage and Data Prep:* All of the resources are collected into a data lake using different types of robots and crawlers. The documents and resources are then pre-processed to standardize and facilitate their further utilization in the data mining tasks.
- *RecSys:* This module is based on two components: 1) a search engine; and 2) an application that implements a recommendation system founded on the content-based, collaborative filtering, and hybrid methods. The RecSys represents the core of SmartCon because it integrates the different models of analytics, ML, and artificial intelligence. In addition, it not only offers several web services to the LMS in order to send recommendations to the

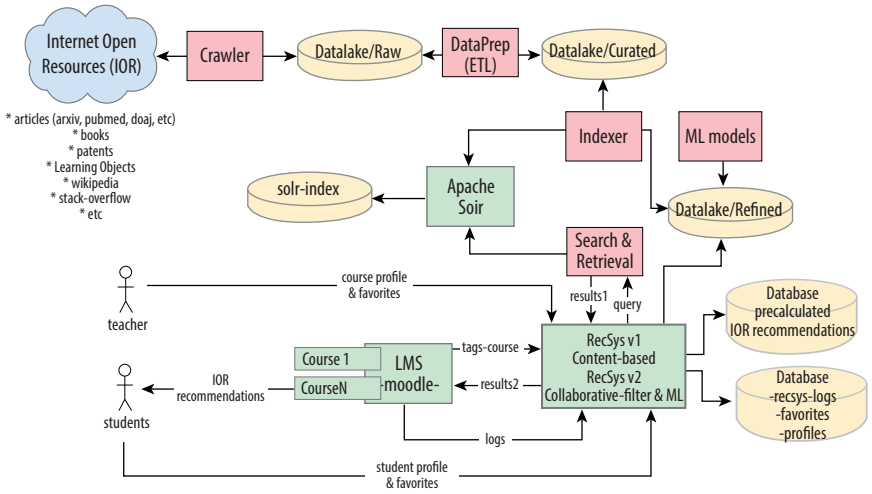
courses, teachers, and students, but it also exposes web services to manage other features, such as profiles, favorites, caches, logs, etc., which will improve future recommendations.

- *LMS:* This is an application in which courses are managed, teachers perform instructional design, contents are loaded, and students interact with such aspects. In order to integrate this LMS in SmartCon, some plug-ins and adapters were developed. Moreover, the LMS received the recommendations from the RecSys module, and it managed several features that improved the recommendations through the learning process.

SmartCon Detailed Architecture

Based on the architecture described in Figure 9, SmartCon defines the components presented in Figure 10:

Figure 10. A Detailed Architecture of SmartCon



Source: Prepared by the authors

- *IORs:* SmartCon identifies different categories and sources, which are described in Table 3. These digital resources are mainly texts in different formats (e.g., PDF, HTML, TXT, etc.), characterized

as unstructured or semi-structured. Nevertheless, SmartCon also considers the resources from audio, video or other formats.

Table 3. Categories of Digital Resources

Category	Source (MD: Metadata & FT: full-text)
Articles	* Arxiv (MD & FT) * Pubmed (MD) * DOAJ (MD) * OpenAire (MD)
Patents	* WIPO (MD) * USPTA (MD)
Books	* BOAJ (MD)
Wikis	* Wikipedia-English (FT) * Wikipedia-Spanish (FT)
Communities	* Stack-overflow (FT)
Learning Objects	* Merlot (MD)

Source: Prepared by the authors

- *Harvester:* SmartCon collects near 30 million digital resources among the full-texts and metadata. All of the data is stored in a data lake (i.e., the raw zone), after which it is processed by the Data Prep-ETL module. After all of the data is curated, it is stored back in a data lake (i.e., the curated zone), and then it is ready to be indexed and analyzed by ML techniques. The main purpose of the Data Prep module is to filter some fields by source and place all of the documents in the same format in order to facilitate further processing (i.e., indexing and data mining).
- *Indexer:* The Indexer places the normalized IORs into a search engine. In this project, we used Apache Solr. This search engine supported the first version of the recommendation system, which was content-based using an information retrieval system.
- *ML Models:* The core of SmartCon is that the models are implemented using ML techniques. In the ML stage, we designed and implemented several ML models (supervised and unsupervised). These models focused on improving the metadata to be indexed into the search engine, and generated new data to support the RecSys.

- *Search Engine:* SmartCon uses an information retrieval system based on Apache Solr, which is a popular open-source software to index, search, and retrieve multiple types of files. Apache Solr is based on Apache Lucene. Thus, it supports full-text data (PDF, HTML, JSON, CSV, DOC, etc.), which includes the data sources in SmartCon. The standard file format indexed by Apache Solr is CSV.
- *Recommendation System (RecSys) Engine:* SmartCon uses two types of RecSys: content-based and collaborative filtering. Content-based RecSys (based on Apache Solr) is a search engine that retrieves relevant documents, according to certain keywords or tags specified by the teacher in the course's sections and by the students in their personal preferences in the LMS. In this case, the RecSys functions as a traditional search engine, but it is smarter, since the students are unaware of how the documents are selected and organized according to their contexts, profiles, and behaviors. As for collaborative filtering, it is based on ML. More specifically, this RecSys merges several approaches such as hybrid RecSys, collaborative filtering through the use of the RecSys, and the logs from the LMS (Moodle). The main drawback of recommendation systems based on collaborative filtering is the cold start. In order to solve this problem, the RecSys collects various data from the LMS, including logs related to the contents accessed by the students (either from the course or from SmartCon), interactions among the students, etc.
- *LMS:* Finally, the most important module of the architecture is the LMS, in which teachers and students interact and receive recommendations from SmartCon. Within the LMS, teachers can define the course profiles (e.g., tags per course and sections, categories and sources of interest, language, etc.), and manage favorite contents suggested by SmartCon (e.g., contents found on SmartCon or external links), while students can define their own profiles (e.g., categories, sources, language, favorites, etc.). This module is also implemented by using an open-source platform called "Moodle", which is a state-of-the-art LMS from the free-software environment. Additionally, Moodle interacts with SmartCon in

four ways: 1) through plug-ins and blocks developed within SmartCon; 2) by activating the modules of the course and student profiles; 3) by accessing the graphical user interface (GUI) of SmartCon to search for a specific course or student; and 4) by exposing some web services to send logs and information toward RecSys. Moreover, SmartCon can support any legacy or new Moodle.

Big Data Architecture

SmartCon includes the characteristics related to the "5Vs" of big data: 1) Volume: the ability to store high volumes of data (gigabytes to petabytes, due to the large amount of resources available on the Internet); 2) Variety: the main data source is either unstructured or semi-structured; 3) Velocity: the models that are trained and tested require high-performance computing, with bounded processing times; 4) Value: the ability to extract information and knowledge from the raw data; and 5) Veracity: the ability to perform quality processes (filtering and Data Prep).

In general, big data technologies are used as storage (i.e., configured as a data lake and as SQL/NoSQL databases) and as Apache Spark clusters that allow the processing of large volumes of unstructured and semi-structured data.

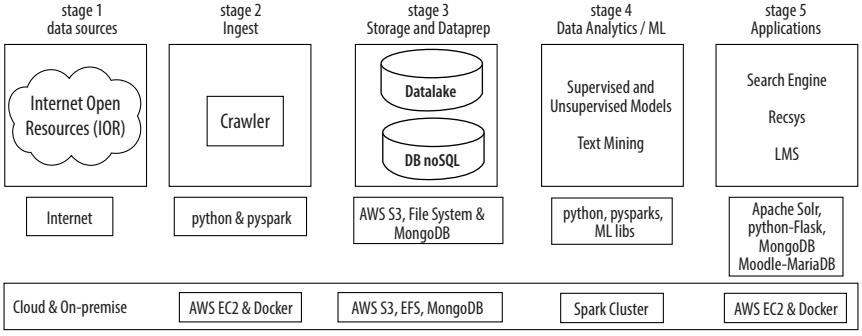
In this project, we used the following five-stage reference architecture:

- Stage 1. Data sources: These include the same sources identified in SmartCon's detailed architecture, mainly based on unstructured and semi-structured data such as text documents, audio or video.
- Stage 2. Ingest: Software robots that collect data from the sources on the Internet.
- Stage 3. Data storage and preparation: The main storage is performed in a data lake, which is designed with four zones: raw, stage, trusted, and refined. More specifically, the data is first stored in the raw zone or in the stage zone if the data requires some pre-processing such as data decompression or file format transformation. Then, a series of extraction, transformation, and loading (ETL) processes are performed, which allow the data to be normalized for later stages of data analysis and the search engine.

- Stage 4. Data analysis: This is considered the main component of the project, since it is the stage of information and knowledge generation toward the intelligent recommendation module. From the perspective of big data, it is the component that allows the execution of ML models over a Spark cluster. The data and output models are stored back in the data lake (i.e., the refined zone) or in NoSQL databases, for later use in the RecSys. This stage primarily runs in a supercomputer environment, with big data clusters based on Apache Spark and Hadoop. In addition, it is mainly deployed in the Academic Data Center at EAFIT University and (to a lesser extent) in the AWS cloud.
- Stage 5. Application: This stage implements the search and retrieval engine modules (based on Apache Solr), and uses the RecSys module for both content and collaborative filtering. It also implements a module for managing preferences, logs, favorites, etc. Finally, it adapts an open-source LMS, such as Moodle, to utilize all plug-ins and adapters. These applications were deployed in the test and the production environment of the pilot test. Moreover, the main deployment was in the AWS cloud, while the testing was conducted at the EAFIT Academic Data Center.

Figure 11 presents SmartCon's big data reference architecture, including the different technologies used and the execution environment:

Figure 11. SmartCon Big Data Architecture



Source: Prepared by the authors

Development and Deployment of SmartCon

SmartCon was implemented through several software components developed within the project. It also used different open-source projects to implement certain components, and employed the research infrastructure at EAFIT University as well as cloud services for project deployment. At the software level, the following modules were developed:

- Crawler: A software module developed in Python and PySpark that runs on a data server and collects all of the data sources on the Internet. The collected data is then stored in a data lake (i.e., the raw zone).
- Data Prep: A software module that runs on a data server and transforms the original data from the Internet into a standardized form in order to facilitate the mining processes and conduct normalization for the indexing module.
- Indexer: A software module that runs on a data server and indexes all of the standardized contents in the search and retrieval engine.
- ML Models: A software module that implements all of the data mining models. It is also a software component that primarily utilizes the big data and data processing infrastructure. It also runs on a Spark cluster, since some models can take hours or even days to run. More specifically, these models mainly run in an on-premise cluster in the Academic Data Center at EAFIT. In this case, the cluster is built using three servers that total 512 GB RAM, 4 TB of SSD storage, 72 cores, and two Nvidia K80 GPUs.
- RecSys: A software module that implements the main core of SmartCon. It integrates the results of the ML models and search engine and exposes a series of web services toward the LMS.

Overall, SmartCon uses the following open-source software:

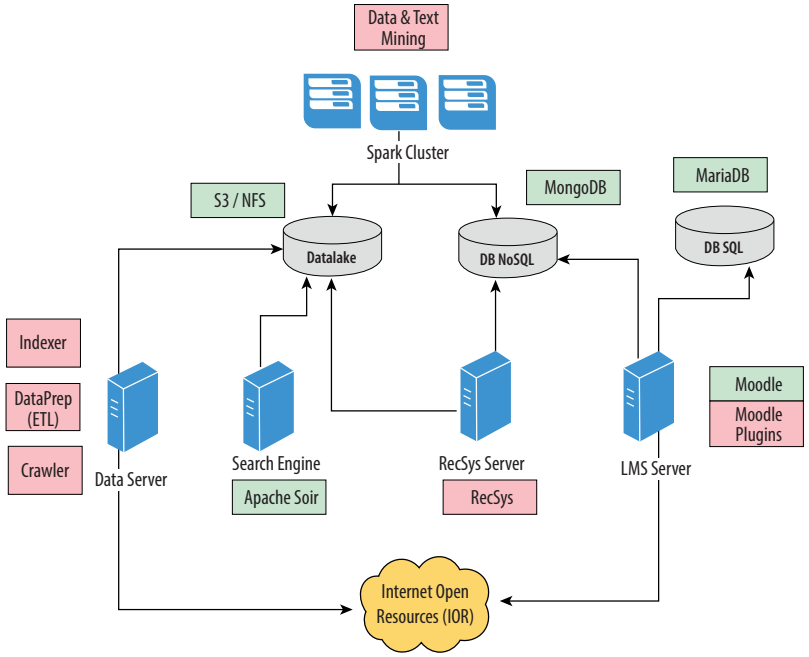
- Apache Solr [1] Version 8.6.2 to implement the search engine.
- NoSQL MongoDB [2] database Version 4.0, in which the RecSys stores the data, logs, student and course profiles, favorites, caches, etc.
- MariaDB[3] SQL database Version 10.3, used by Moodle.

[1] https://lucene.apache.org/solr/
[2] https://www.mongodb.com/
[3] https://mariadb.org/

- LMS Moodle [4] 3.9.2, in which the courses, users (i.e., the students and teachers), contents, etc. are managed. It also creates users for students and teachers, stores course contents, etc.

The deployment and information technology (IT) infrastructure for SmartCon is shown in Figure 12.

Figure 12. Deployment and IT Infrastructure



Source: Prepared by the authors

AWS Cloud Services: At the cloud level, services are primarily used online and in production for the following:

- AWS EC2 virtual machines, which run both native or docker versions of different SmartCon modules. They also run data server, search engine, RecSys, LMS server, DB SQL server, and DB NoSQL server.

[4] https://moodle.org/

- Object storage in AWS S3 to deploy the data lake.
- Load balancers with AWS ELB.
- Name service in AWS Route 53 to manage the domain: contenidosint.org

On-premises servers and Spark cluster: Academic Data Center IT infrastructure, in which the online testing version of SmartCon runs and where they run the following ML models:

- Virtual machines for the testing environment.
- Virtual machines for the software development environment.
- Apache Spark cluster for training and testing of the ML models. This component also runs in batches.

Testing SmartCon

SmartCon has been tested in various courses in the undergraduate computer science program at EAFIT University, including Computational Thinking, Programming Fundamentals, and Special Topics in Telematics.

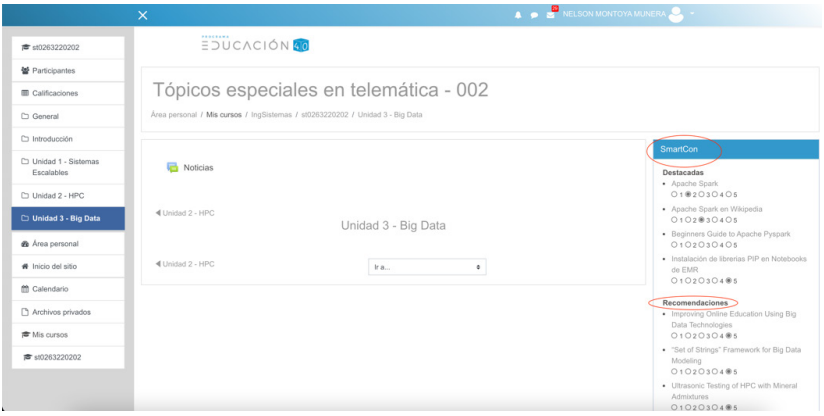
Each of these courses used the following modules, which were activated in Moodle:

- Recommendation module: The block in which SmartCon recommendations are received.
- Favorite module: The contents that are selected by the teacher, either from SmartCon or external websites. They are also contents from SmartCon that, due to their popularity, are promoted to this category.
- Search module: SmartCon provides teachers and students with a search interface, which can be accessed from Moodle as well as from an external application.
- Profile module: Allows students and teachers to select categories, sources, and language preferences. The objective is to personalize the recommendations generated by SmartCon.
- Scoring module: This module allows the scoring of the contents, both implicitly and explicitly. More specifically, implicitly, it is through a wrapper that intercepts all of the intentions of opening the recommended content, whereas explicitly, it is through ratings such as likes/dislikes or promotions to favorites. All of this data allows SmartCon to "learn" from the interactions with the recommended contents, and to improve its learning algorithms.

- Log module: Allows extracting logs from the Moodle platform to improve the recommendation algorithms, especially the collaborative filtering and hybrid algorithms.

In the GUI Moodle, the following modules are shown. First, the RecSys module is shown in Figure 13.

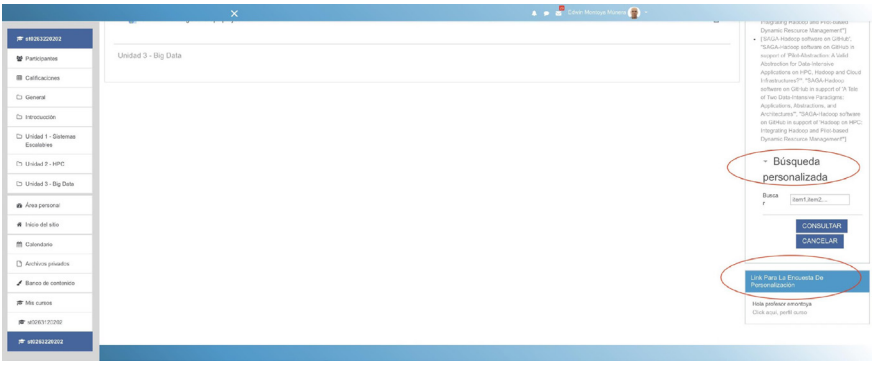
Figure 13. The RecSys Module



Source: Prepared by the authors

Second, the Search module is shown in Figure 14.

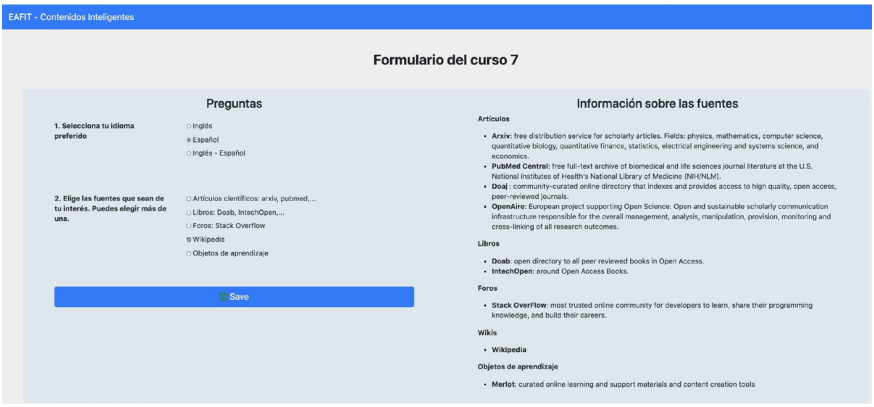
Figure 14. The Search Module



Source: Prepared by the authors

Third, the Profile module is shown in Figure 15.

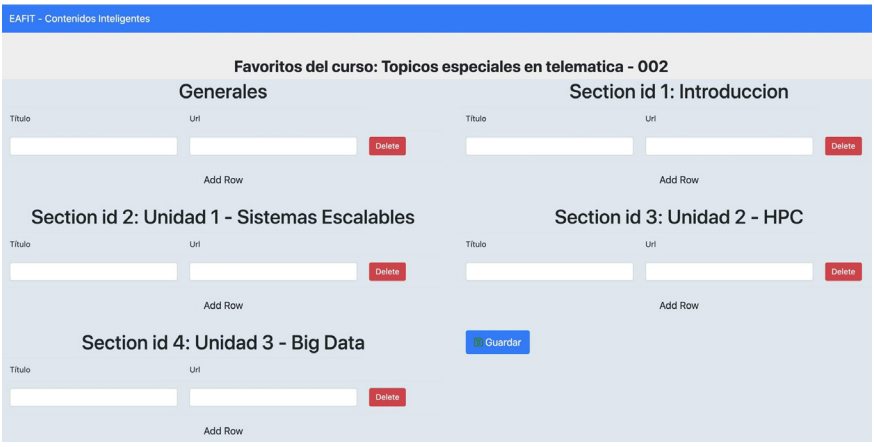
Figure 15. The Profile Module



Source: Prepared by the authors

Finally, the Favorites module is shown in Figure 16.

Figure 16. The Favorites Module



Source: Prepared by the authors

Proposed Testing Methodology

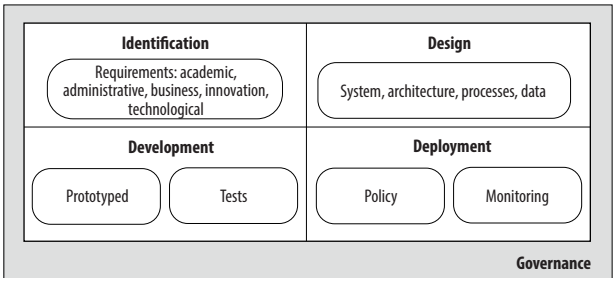
For the recommendation system, the following testing methodology applied in the three aforementioned courses. The methodology consisted of the following three phases:

- *Surveys:* In this phase, three initial surveys were administered to the students. The first was a socio-demographic survey, the second examined learning styles, and the third focused on personalities. As a result of this phase, the data obtained in the Profile module for each student was used to recommend a list of contents for each topic.
- *Content rating:* After recommending such contents, the students rated the list from 1 to 5, according to the level of importance. In this phase, the students mentioned the relevance of the contents, according to the related topic.
- *Evaluation:* Finally, the relevance and precision of the recommended contents were evaluated in order to verify the effectiveness of the recommendation system.

Adapting the ARS Toward a Service-Oriented Architecture (SOA)

The implementation of a SOA architecture will guarantee that the ARS easily adapts to any educational institution's IT. For the implementation of our ARS for VLEs, a SOA methodology, based on Suhardi, Doss, & Yustianto (2015), was proposed, which describes four general phases that guide the construction and management of a service-oriented architecture. The first phase describes the general identification of academic, administrative, business, innovation, technological requirements, etc., while the second phase designs the process, architecture, system, data, and services that can be a part of the architecture. In the third phase, the development and test of each of the previously designed services are carried out, while in the final phase, the architecture is deployed through monitoring, versioning, and the discovery of new services (see Figure 17). Each of these phases can be articulated with SOA governance, in which the processes and activities must be aligned with institutional IT policies.

Figure 17. General Methodology for Adapting the ARS Ioward an SOA



Source: Prepared by the authors

Conclusion

This chapter presented the main contributions of the SmartCon project. The main goal was to integrate different free sources and digital contents from the Internet in order to build an ARS that recommends contents to different users in an educational context, based on big data analytics, autonomic computing, and artificial intelligence paradigms.

Overall, the SmartCon project included five major components: 1) a data lake to store open digital resources; 2) processing tasks to prepare the sources of information; 3) data analytics tasks to enrich the knowledge of the contents, courses, students, and teachers; 4) a search engine system; and 5) a recommendation system to personalize the contents to different users (e.g., students and teachers). In addition, the project developed a prototype called SmartLMS, which is an open-source LMS that uses our recommendation system.

Particularly, the project defined the concept of an ARS (with intelligent and autonomic capabilities) and its extensions to consider hybrid recommendation algorithms and emotion recognition. These aspects are important characteristics that can be easily added to an ARS (according to its MAPE-K model), thus supporting the robustness of the system recommendation process.

Meanwhile, the hybrid recommendation process was adapted to the data at the moment of execution, after which the hybridization was dynamically configured for each user, depending on the advantages/

disadvantages of the various recommendation approaches. In this case, the recommendation approaches were determined in real time (see Salazar et al. (2020, n.d), Salazar, Montoya & Aguilar (n.d.) and Valera et al. (n.d, -a,b) for more details). More specifically, the proposed fuzzy system for managing the integration of the recommendation approaches (using the defined metrics) included the ability to solve existing problems, such as cold start, in the individual recommendation algorithms.

Moreover, the SmartCon project developed different mining tasks for the various tasks required by the ARS. For example, it proposed different content extraction approaches and featured an engineering process for audio datasets. It also analyzed different recognition approaches and meta-features that could be used to guide the hybrid recommendation process.

Finally, the project developed a prototype in which it clearly defined the platform required by our ARS. Particularly, the prototype was composed of a data lake, a ML module, a search engine system, a recommendation system, and a LMS.

As for future recommendations, research must exploit the different sources of knowledge incorporated by our proposal in a smart classroom in order to improve the learning process. In this regard, various concepts, such as autonomous learning analytics cycles (Aguilar, Cordero, & Buendía, 2018) that allow the natural integration of context information in a dynamic process of continuous improvement, should be used. Furthermore, future works should analyze the results of the learning process using appropriate metrics that can measure the overall impact of our recommendation system on students.

Acknowledgments

This work was supported by Project No. 64366 "Contenidos de aprendizaje inteligentes a través del uso de herramientas de Big Data, Analítica Avanzada e IA" –The Ministry of Science –The Government of Antioquia –The Republic of Colombia.

References

Adomavicius, G., & Tuzhilin, A. (2005, June). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734-749. https://doi.org/10.1109/tkde.2005.99.

Aguilar, J., Cordero, J., & Buendía, O. (2018). Specification of the autonomic cycles of learning analytic tasks for a smart classroom. *Journal of Educational Computing Research*, *56*(6), 866-891.

Aguilar, J., Salazar, C., Velasco, H., Monsalve-Pulido, J., & Montoya, E. (2020). Comparison and evaluation of different methods for the feature extraction from educational contents. *Computation*, *8*(2), 30.

Aguilar, J., Valdiviezo-Díaz, P., & Riofrio, G. (2017). A general framework for intelligent recommender systems. *Applied Computing and Informatics*, *13*(2), 147-160. https://doi.org/10.1016/j.aci.2016.08.002.

Balabanovic, M., & Shoham, Y. (1997, March). Fab: content-based, collaborative recommendation. *Communications of the ACM*, *40*(3), 66–72. https://doi.org/10.1145/245108.245124.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, *12*(4), 331-370.

Burke, R. (2007). Hybrid web recommender systems. In P. Brusilovsky, A. Kobsa & W. Nejdl (Eds.), *The Adaptive Web* (pp. 377-408). Springer.

Cacheda, F., Carneiro, V., Fernández, D., & Formoso, V. (2011, February). Comparison of collaborative filtering algorithms. *ACM Transactions on the Web*, *5*(1), 1-33. https://doi.org/10.1145/1921591.1921593.

Cambria, E., Fu, J., Bisio, F., & Poria, S. (2015, January). Affectivespace 2: Enabling affective intuition for concept-level sentiment analysis [Conference paper]. In *Proceedings of the Twenty AAAI Conference on Artificial Intelligence*, New York (pp. 508-514). AAAI Press.

Cambria, E., Poria, S., Hazarika, D., & Kwok, K. (2018, February). Senticnet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings [Conference paper]. In *Proceedings of the Thirty Second AAAI Conference on Artificial Intelligence*, New Orleans (pp. 1795-1802). AAAI Press.

Chauhan, G. S., Agrawal, P., & Meena, Y. K. (2019, April). Aspect-based sentiment analysis of students' feedback to improve teaching-learning process [Conference paper]. In *Proceedings of Information and communication technology for intelligent systems*, Ahmedabad (pp. 259-266). Springer. https://doi.org/10.1007/978-981-13-1747-7_25.

Chirita, P.-A., Nejdl, W., & Zamfir, C. (2005, November). Preventing shilling attacks in online recommender systems [Conference Paper]. In *Proceedings of the seventh ACM international workshop on web information and data management WIDM '05*, Bremen (pp. 65-74). ACM Press. https://doi.org/10.1145/1097047.1097061.

Faria, A. R., Almeida, A., Martins, C., Gonçalves, R., Martins, J., & Branco, F. (2017). A global perspective on an emotional learning model proposal. *Telematics and Informatics*, *34*(6), 824-837.

Golovin, N., & Rahm, E. (2004, April). Reinforcement learning architecture for web recommendations [Conference Paper]. In *Proceedings of the International Conference on Information Technology: Coding and Computing, 2004. Proceedings*, Las Vegas (pp. 398-402). IEEE. https://doi.org/10.1109/ITCC.2004.1286487.

Hill, W., Stead, L., Rosenstein, M., & Furnas, G. (1995, May). Recommending and evaluating choices in a virtual community of use [Conference Paper]. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems CHI '95*, Denver (pp. 194-201). ACM Press. https://doi.org/10.1145/223904.223929.

Huang, Z., Chen, H., & Zeng, D. (2004, January). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 116–142. https://doi.org/10.1145/963770.963775.

Immordino-Yang, M. H., & Damasio, A. (2007). We feel, therefore we learn: The relevance of affective and social neuroscience to education. *Mind, Brain, and Education*, *1*(1), 3-10.

Jimenez M., Aguilar J., Monsalve-Pulido J. & Montoya, E. (2020). An automatic approach of audio feature engineering for the extraction, analysis and selection of descriptors (Accepted for publication, *International Journal of Multimedia Information Retrieval*).

Knob, L. A. D., Esteves, R. P., Granville, L. Z., & Tarouco, L. M. R. (2017, July). Mitigating elephant flows in sdn-based ixp networks [Conference Paper]. In *Proceedings of the 2017 IEEE Symposium on Computers and Communications (ISCC)*, Heraklion (pp. 1352-1359). IEEE. https://doi.org/10.1109/ISCC.2017.8024712.

Kruse, K. (2002). *Introduction to instructional design and the addie model*. Retrieved January, 26, 2005.

Lam, S. K., & Riedl, J. (2004, May). Shilling recommender systems for fun and profit [Conference Paper]. In *Proceedings of the 13th international conference on World Wide Web*, New York (pp. 393-402). ACM Press. https://doi.org/10.1145/988672.988726.

Monsalve-Pulido, J., Aguilar, J., Montoya, E., & Salazar, C. (2020). Autonomous recommender system architecture for virtual learning environments. *Applied Computing and Informatics*.

NCES. (2014). Common Education Data Standards (Vol. 66). Retrieved from http://ceds.ed.gov.

Obeid, C., Lahoud, I., El Khoury, H., & Champin, P.-A. (2018, April). Ontology-based recommender system in higher education [Conference Paper]. In *Proceedings of the The Web Conference*, Lyon (pp. 1031-1034). ACM Press. https://doi.org/10.1109/IEEESTD.2002.94128.

Pekrun, R. (1992). The impact of emotions on learning and achievement: Towards a theory of cognitive/motivational mediators. *Applied Psychology*, *41*(4), 359-376.

Pekrun, R., Goetz, T., Titz, W., & Perry, R. P. (2002). Academic emotions in students' self-regulated learning and achievement: A program of qualitative and quantitative research. *Educational psychologist*, *37*(2), 91-105.

Salazar, C. Montoya, E. & Aguilar, J. (n.d.). *Analysis of different affective state multimodal recognition approaches with missing data oriented to virtual learning environments* (submitted to publication).

Salazar C, Aguilar J., Monsalve-Pulido J., & Montoya, E. (2020). Análisis de sentimientos/polaridad en diferentes tipos de documentos. *Revista Ibérica de Sistemas y Tecnologías de la Información*, *E38*, 171-184.

Salazar C., Aguilar J., Monsalve-Pulido J., & Montoya, E. (n.d). *A generic architecture of an affective recommender system for e-learning environments* (submitted to publication).

Sarwar, B., Karypis, G., Konstan, J., & Reidl, J. (2001, April). Item-based collaborative filtering recommendation algorithms [Conference Paper]. In *Proceedings of the 10th international conference on World Wide Web*, Hong Kong (pp. 285-295). ACM Press. https://doi.org/10.1145/371920.372071.

Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002, August). Methods and metrics for cold-start recommendations [Conference Paper]. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, Tampere (pp. 253-260). ACM Press. https://doi.org/10.1145/564376.564421.

Schuller, B., Steidl, S., Batliner, A., Vinciarelli, A., Scherer, K., Ringeval, F… others (2013, August). The interspeech 2013 computational paralinguistics challenge: Social signals, conflict, emotion, autism [Conference Paper]. In *Proceedings of the INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association*, Lyon (pp. 148-152). ISCA Archive.

Shardanand, U., & Maes, P. (1995, May). Social information filtering [Conference Paper]. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Denver (pp. 210-217). ACM Press. https://doi.org/10.1145/223904.223931.

Shen, L., Wang, M., & Shen, R. (2009). Affective e-learning: Using "emotional" data to improve learning in pervasive learning environment. *Journal of Educational Technology & Society*, *12*(2), 176-189.

Suhardi, S., Doss, R., & Yustianto, P. (2015). Service engineering based on service-oriented architecture methodology. *Telecommunication Computing Electronics and Control*, *13*(4), 1466-1477.

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

Varela D., Aguilar J, Monsalve-Pulido J., & Montoya, E. (n.d.-a). *Analysis of meta-features in the context of adaptive hybrid recommendation systems* (submitted to publication).

Varela D., Aguilar J., Monsalve-Pulido J. & Montoya, E. (n.d.-b). Propuesta arquitectónica de un sistema de recomendación híbrido adaptativo (Accepted for publication, *Revista Ibérica de Sistemas y Tecnologías de la Información*).

Vijayakumar, V., Vairavasundaram, S., Logesh, R., & Sivapathi, A. (2019). Effective knowledge based recommender system for tailored multiple point of interest recommendation. *International Journal of Web Portals*, *11*(1), 1-18.

Vizcarrondo, J., Aguilar, J., Exposito, E., & Subias, A. (2017). Mape-k as a service-oriented architecture. *IEEE Latin America Transactions*, *15*(6), 1163-1175.

Vladoiu, M., Constantinescu, Z., & Moise, G. (2013, September). QORECT –a case-based framework for quality-based recommending open courseware and open educational resources [Conference Paper]. In *Proceedings of the 5th International Conference on Computational Collective Intelligence, ICCCI*, Craiova (pp. 681-690). Springer.

Yu, L. C., Lee, C. W., Pan, H. I., Chou, C. Y., Chao, P. Y., Chen, Z. H., ... & Lai, K. R. (2018). Improving early prediction of academic failure using sentiment analysis on self-evaluated comments. *Journal of Computer Assisted Learning*, *34*(4), 358-365.

Zhu, F., Ip, H. H. S., Fok, A. W. P., & Cao, J. (2008). PeRES: A personalized recommendation education system based on multi-agents and SCORM. In H. Leung, F. Li, R. Lau, Q. Li (Eds.), *Advances in Web Based Learning – ICWL 2007. ICWL 2007. Lecture Notes in Computer Science* (vol 4823, pp. 31-42). Springer. https://doi.org/10.1007/978-3-540-78139-4_4