Computers and Education: Artificial Intelligence 2 (2021) 100009

Contents lists available at ScienceDirect

Computers and Education: Artificial Intelligence

journal homepage: www.elsevier.com/locate/caeai

Deep auto encoders to adaptive E-learning recommender system

Everton Gomede, PhD [a,*], Rodolfo Miranda de Barros, PhD [b], Leonardo de Souza Mendes, PhD [a]

[a] Electrical Engineering and Computer School, State University of Campinas, Av. Albert Einstein, 400, Cidade Universitária Zeferino Vaz, Distrito Barão Geraldo, Campinas, 13083-852, SP, Brazil

[b] Computer Science Departments, State University of Londrina, Rod. Celso Garcia Cid, Km 380, s/n, Campus Universitário, Londrina, 86057-970, PR, Brazil

ARTICLE INFO

Keywords:
Adaptive recommender systems
Deep auto encoder
Artificial neural networks
E-learning
Lifelong learning

ABSTRACT

Adaptive learning, supported by Information & Communication Technology (TIC), is an important research area for educational systems which aim to improve the outcomes of students. Thus, the investigation of what should be adapted and how much to adapt constitute a foundation to Adaptive E-learning Systems (AES). In this paper, we compared three classes of Deep Auto Encoders and the popularity model to address the problem of learning and predicting the preferences of student on AES: Collaborative Denoising Auto Encoders (CDAE), Deep Auto Encoders for Collaborative Filtering (DAE-CF), and Deep Auto Encoders for Collaborative Filtering using Content Information (DAE-CI). The results point out that the DAE-CF is more effective providing significant adaptability. Furthermore, we present the concept named as signature of preference to represent a more granular class of adaptability. Therefore, this model may be used in e-learning systems to provide adaptability and help to improve the outcomes of students.

1. Introduction

Adaptive learning, supported by Information & Communication Technology (TIC), is an important research area for scholar systems to improve the outcomes of students (Bobadilla et al., 2009; Hwang et al., 2013; Krauss, 2016; Gomede, Gaffo, Briganó, Barros, & Mendes, 2018). This kind of approach allows pedagogical elements such as control flow, enrollment, attention and sequencing to be analyzed and provide suitable recommendations to students, teachers, administrators, relatives, and/or responsible (Gomede, Gaffo, Briganó, Barros, & Mendes, 2018). In addition, the learning is tracked for content adaptation and formatting to maximize the cognitive skills of each student and avoid the flood of information. Thus, to investigate what should be adapted and how much to adapt constitutes a foundation to Adaptive E-learning Systems (AES).

An AES is composed of two essential components: data and intelligent algorithms. Data provides the behavior of students when they are interacting with Learning Objects (LO) like videos, texts, tests, sounds, simulations, charts, and so forth. This behavior is produced by implicit and explicit feedback (Bobadilla et al., 2009). Implicit feedback occurs without students' awareness, for example, the click in a text, the time spent in a video, etc. Explicit feedback occurs when students rate an LO, add in a favorite list and/or comment about something, etc. Using the information on students' behavior, intelligent algorithms allow the development of models which learn behaviors based on data, being able to predict them. For instance, a behavior might represent the preference over LO that is related with simulations or charts. Another example is predicting which LO should be recommended to a student based on its preferences. Therefore, these components allow an AES to learn and predict the behavior of students.

One of the state-of-the-art algorithms to learn and predict users' (in our case, students) behaviors in a recommender system are called Deep Auto Encoders (DEA) (Sedhain et al., 2015; Strub et al., 2016; Li and She, 2017; Chae et al., 2019). These algorithms are built on Artificial Neural Networks (ANN) which uses the inputs as the outputs to train the network by semi supervised learning (Barbieri et al., 2017). Moreover, these algorithms are feedforward and use the backpropagation algorithm to update the weights of synapses with the rate provided by the stochastic gradient descent. Also, these algorithms work well with sparse matrices generated by student/object interactions, which do not require a labeled dataset and many hyperparameters, being able to deal with noise data (Barbieri et al., 2017), and allow many hidden layers, which increase network learning to avoid over and underfitting.

Recommendation systems applied to business context aim to increase sales by recognizing the preferences of users (Sedhain et al., 2015).

* Corresponding author.
E-mail addresses: evertongomede@gmail.com, e18056@dac.unicamp.br (E. Gomede), rodolfo@uel.br (R.M. de Barros), lmendes@decom.fee.unicamp.br (L.S. Mendes).

https://doi.org/10.1016/j.caeai.2021.100009
Received 13 October 2020; Received in revised form 20 January 2021; Accepted 20 January 2021

However, an AES should include some features to be considered effective. For example, the system should consider the case in which a student A marks an object that causes this object to become available to another student B, without this student to be aware of this interaction. Thus, the cycle of students (select and rate) and system (group and filter) provides the five components, or levels, of an adaptive e-learning recommendation system. On the first level, a recommendation system works with collaborative filtering, i.e., depends on a voluntary and responsible interaction between students and objects (Hwang et al., 2013; Krauss, 2016). For example, an incorrect rating of a book might be generating a wrong recommendation, but on the other hand, a rating done by a group of specialized users (professors, tutors, reviewers, and/or researchers) might be increasing the value of the recommendations. In a second, the recommendation system presents collaborative intelligence. For example, a book recommended to student A is not related to the behavior of a student B, instead, the behavior of some collectivity is responsible for recommendations (Krauss, 2016; Protasiewicz et al., 2016). Third, a recommendation system is in user control. For instance, a recommended book, which is part of mandatory bibliography, cannot be rejected by students (although this book might be evaluated by students). As follows, the coverage of recommendations is limited. Fourth, recommendation systems offer guidance. For example, a list of recommended articles, which support students in their research, is a guide (Medio et al., 2020). This orientation represents a fundamental performance issue in learning science; excess of autonomy can be costly and generate low performance results. The key idea is achieving the equilibrium. Finally, recommendation systems provide adaptability. Different students do not take equal advantage of the same instructions; thus, adaptability is needed.

From a pragmatic point of view, an adaptive e-learning recommendation system aims to improve the outcomes of students. However, there are some secondary objectives like (a) relevance (an LO needs to be relevant, or it may become useless); (b) novelty (LO needs to present innovation or not used in the past); (c) serendipity (an LO which is not expected, but is still relevant, to the student); and (d) diversity (several types of LO improving the coverage of the subject to be learned). Moreover, there are some requirements to make suitable recommendations: (a) transparency to explain the recommendation, (b) confidence to support students to use the LO, (c) satisfaction to improve the outcomes of students, (d) persuasion to convince students to use the LO, (e) efficiency to assist students in their decisions, and (f) effectiveness to help the students to make suitable decisions (Bobadilla et al., 2009; Hwang et al., 2013; Krauss, 2016; Protasiewicz et al., 2016; Barbieri et al., 2017; Nafea et al., 2019; Sedhain et al., 2015; Strub et al., 2016; Chae et al., 2019). In this manner, the component of intelligent algorithms might be used in this context.

In this paper, we compare three classes of DEA and the popularity model to address the problem of learning and predicting students' preferences on AES: Collaborative Denoising Auto Encoders (CDAE), Deep Auto Encoders for Collaborative Filtering (DAE-CF), and Deep Auto Encoders for Collaborative Filtering using Content Information (DAE-CI). The hyperparameters, activation function, batch size, dropout, epoch, layers, loss function, and learning rate are presented and discussed. The performance metrics Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG), Personalization, Coverage, and Serendipity (SAUC) are used to compare the results. Thus, we analyze the evaluation of recommendation systems focusing on the quality of recommendations rather than on their predictive accuracy of algorithms. Finally, the split strategy of the dataset is stated and discussed.

This paper is organized as follows. This initial section presents fundamental considerations and justifications for this work and defines its core objectives. Section 2 presents an overall review of the key topics treated and the main works upon which this work is based. Section 3 presents the main concepts behind the experiments and describes the comparison. Section 4 presents the discussions upon the results obtained from the data analysis, architectures, hyperparameters, metrics, and models. In the concluding section, the conclusions and subsequent developments are presented.

2. Related works

Many papers about recommendation systems are dedicated to computational issues like performance metrics, mathematical models, algorithms, and data pre-processing (Strub et al., 2016; Li and She, 2017; Chae et al., 2019; Ordway-West et al., 2018). However, there is an increased awareness related to pedagogical issues to improve the outcomes of students (Bobadilla et al., 2009; Hwang et al., 2013; Krauss, 2016; Protasiewicz et al., 2016; Barbieri et al., 2017; Nafea et al., 2019; Sedhain et al., 2015; Strub et al., 2016; Chae et al., 2019). Like so, some researchers have published approaches to e-learning recommendation systems including educational elements (Bobadilla et al., 2009; Hwang et al., 2013; Krauss, 2016; Protasiewicz et al., 2016; Barbieri et al., 2017; Nafea et al., 2019).

Related to computational issues, one of the firsts papers presenting the auto encoder applied to a recommendation system was proposed by Wang et al. (2015). The auto encoder architecture is a class of feedforward neural networks which use semi supervised learning and, therefore, does not need a labeled dataset (Strub et al., 2016; Li and She, 2017; Chae et al., 2019; Ordway-West et al., 2018). It works with collaborative filtering based on sparse matrices, composed of implicit and explicit feedback, and provides better results compared to other algorithms of collaborative filtering. The deep hidden layers, responsible for coding and decoding, constitute the latent space which represents the factorization of the sparse matrix. The number of hidden layers varies based on the problem. Also, these architectures use many components present in general artificial neural networks such as loss function (for example, Root Mean Square Error (RMSE)), activation function, backpropagation algorithm, stochastic gradient descent, learning rate, bias, and splitting dataset. The standard metrics used by evaluating the predicted outputs is the Recall@K, where K represents the number of objects to be recommended. However, the models built by training of auto encoders might be difficult to explain due to the stochastic process of updating the weights. Another limitation is the dataset available to train and test. If the number of observations is limited, the model may present an overfitting or underfitting (Wang et al., 2015). Moreover, the splitting of the dataset needs to be considered developing suitable models. The most known split techniques are the holdout, k-fold cross-validation, leave-one-out, and

Table 1
The structure and some example of the raw data.

student_id	object_id	learning_object[a]	ip	hours	timestamp
1	1	Learning Object 1	172.168.1.1	1.10	1517446861
2	2	Learning Object 2	172.168.1.1	0.30	1517446161
…	…	…	…	…	…
n	n	Learning Object n	200.175.1.1	1.00	1512441861

[a] The name of learning objects was suppressed to avoid issues related to privacy (GDPR[1]).

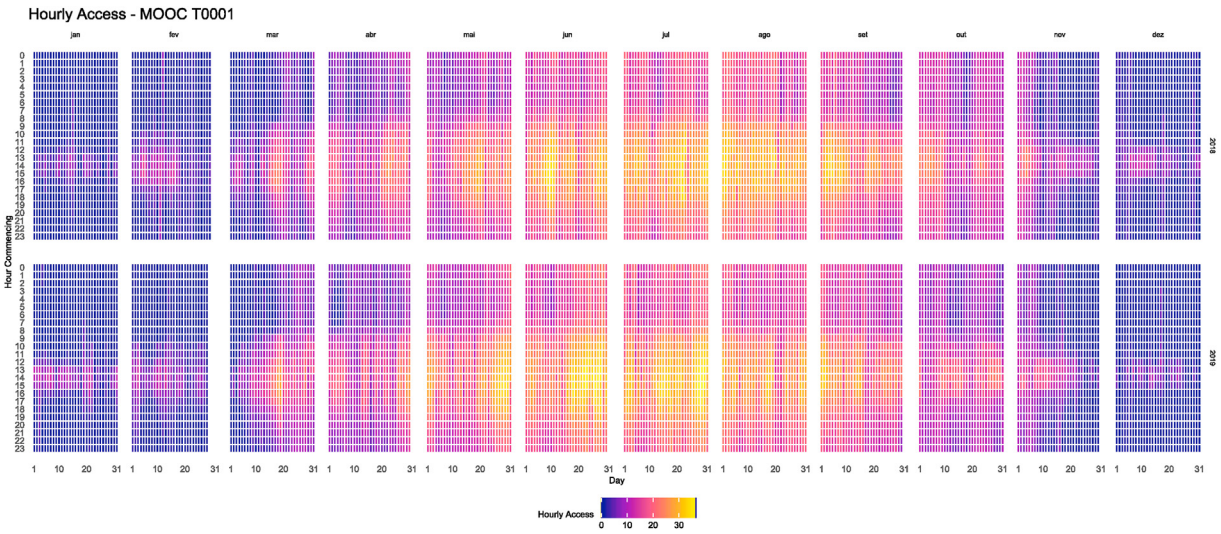


Fig. 1. Temporal preference of students to access learning objects. It can be noted that such accesses were made during the school period and, most of them, between the months of March until October and between the hours of 9 a.m. until 6 p.m. The dataset was collected between January of 2018 and December of 2019.

bootstrap (Wang et al., 2015). Broadly speaking, an auto encoder is composed of common elements of neural networks, which build a latent space by coding and decoding sparse matrices, being suitable to fill out the gaps of interaction between student/object, learning and predicting the behavior of students.

With respect to pedagogical issues, one of the firsts papers on the subject was proposed by Bobadilla et al. (2009). The authors present many types of learning objects to be recommended like books, texts, tests, subjects, and teachers, providing a broad vision of e-learning recommendation systems. Also, the learning objects can be standardized by use of Shareable Content Object Reference Model (SCORM) and provided by Learning Objects Repository (LOR). The use of the SCORM and LOR improves the quality of recommendations and helps teachers, tutors, and administrators to build valuable learning objects. Likewise, these objects are manipulated from pre-school to post-graduate degree and reutilized by many students, teachers, and tutors. Also, other papers present the results of application from recommendation systems to increase the engagement of students; this is achieved by use of the concept of flow (Bobadilla et al., 2009; Hwang et al., 2013; Krauss, 2016; Shih, Chuang, & Hwang, 2010). These results were collected from real environments. Moreover, other ways of adaptability were proposed by the classification of learning styles. These approaches are based on Felder-Silverman Learning Style Model (FSLSM) and identify the tendency of students to acquire, process, comprehend, and store information (Nafea et al., 2019). Shih et al. (2010) proposed an adaptability of the learning path of students, selecting the suitable courses, modules, and content according to preferences for them. To sum up, the authors proposed approaches to provide adaptability of many aspects (learning objects, paths, contents, and so forth) to assist students to improve their outcomes.

According to Dockterman (2018), there are five strategies of adaptability, summarized from the simplest to the most complex; (a) based on the name (b) auto described, (c) segmented, (d) based on cognition, and (e) student centered. The first one boils down to identify the characteristics of students, like name, and to present that information on the content. The second one, besides the characteristics, collects the preferences of students, like the tendency of determined subjects. The third one classifies students by similarity and provides adaptability to the group. The fourth one identifies the cognitive preferences, for example, the tendency to hear an audio rather than examine a video. Finally, the last one collects performance of students (in tests, for example) and adjusts the difficulty level of the content. This strategy requires real-time data (Bobadilla et al., 2009; Hwang et al., 2013; Krauss, 2016). Our approach can be considered based on cognition, but instead of the set the preferences of students in a class previously defined, we build a signature of preferences to consume learning objects. This approach allows building models suitable to each student and predicting the behavior from the use of a learning object.

Table 2
The sparse matrix built with the transposed raw data.

	object_id				
student_id	**1**	**1**	**2**	**…**	**n**
	2	1.10	?	…	?
	…	?	0.30	…	?
	N	?	?	…	1.00

*The "?" symbol represents the pair student/object without interaction.

3. Materials and methods

The essential idea behind the operation of the auto encoder is to fill out the unknown values of the input sparse matrix (Sedhain et al., 2015). This operation provides 2 features: (a) to learn students' behavior and (b) to predict the probability of consuming determined learning objects. The learning feature is based on the interactions of student/object which provides the representation of how each student interacts with available LOs. The predicting feature is related to the probability, based on previous behavior of each student, to interact with novel LOs. Thus, both are intrinsically dependent.

3.1. The dataset

The dataset used is composed of interactions of students with learning objects from a Massive Open Online Course (MOOC). Examples of these objects are videos from open courses of many subjects related to computer science. The interaction obtains implicit feedback provided by the time expended per student in a video, for instance. The structure of raw data is shown in Table 1.

The dataset does not contain restrictions, as explained before in Section 1, like mandatory learning objects. In addition, the characteristic hours represent the amount of time spent by a student in an interaction

[1] General Data Protection Regulation, for more details https://gdpr-info.eu/

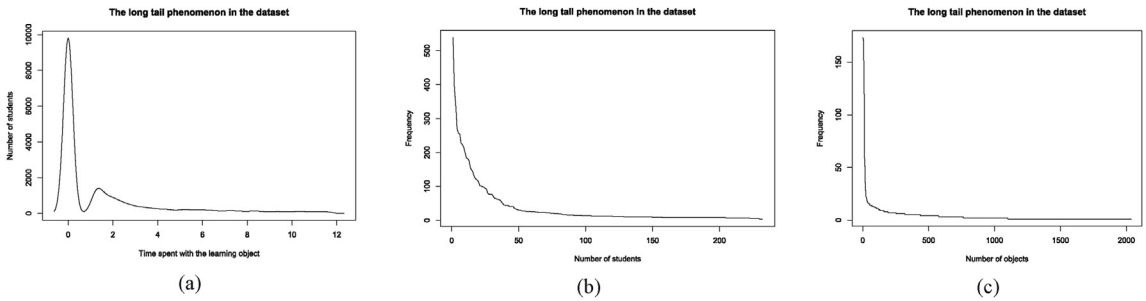


Fig. 2. The (a) overall long tail phenomenon, (b) related to students, and (c) related to learning objects.

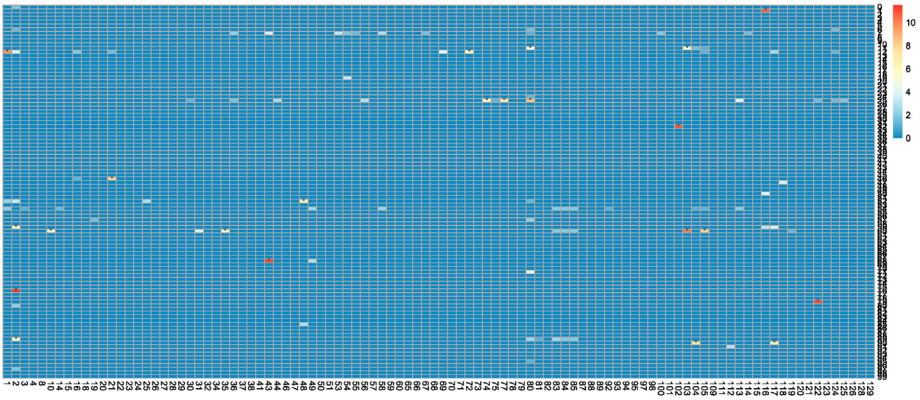


Fig. 3. The sparse matrix resulting from the transposition of the dataset (the firsts 100 students on y axis and firsts 100 objects on x axis). This matrix indicates that few students access few learning objects. Thus, there are many objects and students who have not benefited from the contents that are hidden.

with learning objects. Thus, the student A might spend X and Y amounts of time at the same object A. Furthermore, students can view an object many times and interact with it in different ways (e.g., watching the same object many times, stopping, reloading, forwarding, rewarding, and jumping). Thus, to model the student interest on a given learning object, we aggregate all interactions of the student in an object by a weighted sum of interaction type and applying a log transformation to smooth the distribution[2]. The preference to access the learning object is shown in Fig. 1.

As shown in Fig. 1, students prefer to access the MOOC during the day and the school year. Moreover, the dataset was collected between 2018 and 2019 and the MOOC was available 24/7/365 (except when downtime was planned). The raw data, indicated in Table 1, needs to be transposed to a matrix to be used with the algorithms (Wang et al., 2015). The result is a bidimensional $m \; x \; n$ sparse matrix composed of interactions of students and objects represented by the net amount of time spent in the object. The sparse matrix is shown in Table 2.

The phenomenon that occurs in a recommendation system based on collaborative filtering is called long tail[3] (Strub et al., 2016; Li and She, 2017; Chae et al., 2019; Ordway-West et al., 2018). This phenomenon is ruled by two hypotheses fundamentally different. The first states that most students follow the mass behavior while the minority search specific objects. The second states that most students present the mass and niche behavior. Göbel and Mehm (2013) presented empirical evidence that both hypotheses are valid but are context dependent. Also, the long tail occurs with students and objects. Fig. 2 shows the (a) overall long tail phenomenon, (b) related to students, and (c) related to objects.

As shown in Fig. 2, the long tail represents, in other words, that few students access few objects often. This is a problem to collaborative

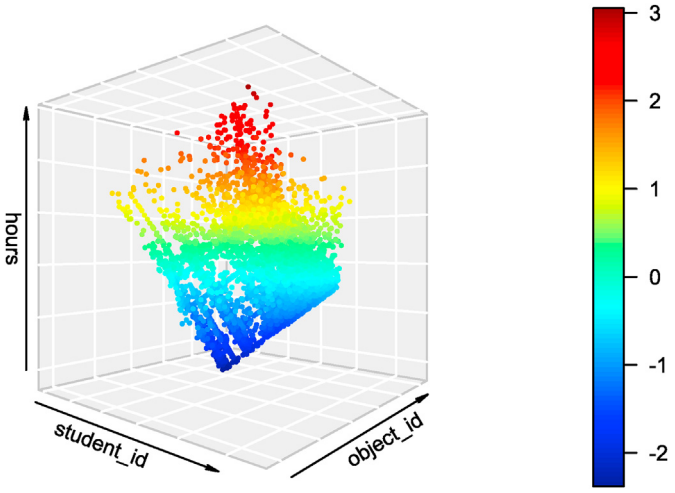


Fig. 4. Cluster analysis implementing the PCA technique that indicates the concentration of the interactions in some students and learning objects. This visualization complements the long tail phenomenon from another point of view. The scale, in log and normalized form, from −2 to 3 represents the time spent in each interaction student/object.

[2] In statistics and image processing, to smooth a dataset is to create an approximating function that attempts to capture important patterns in the data, while leaving out noise or other fine-scale structures/rapid phenomenon.

[3] In probability theory, heavy-tailed distributions are probability distributions whose tails are not exponentially bounded: that is, they have heavier tails than the exponential distribution. In many applications it is the right tail of the distribution that is of interest, but a distribution may have a heavy left tail, or both tails may be heavy.

Table 3
The summary of the pre-processing dataset.

Dataset	Students	Objects	Feedback	Matrix	Sparsity
Open Courses	3,757	5,104	(0:10]	19,175,728	0.9939978

Table 4
The sparse matrix with the missing values filled out.

	object_id				
student_id		**1**	**2**	**…**	**n**
	1	1.10	**1.03**	…	**0.03**
	2	**0.90**	0.30	…	**0.09**
	…	…	…	…	…
	n	**0.01**	**0.61**	…	1.00

*The bold values represent an example of the **predicted** pair student/object's interaction.

filtering because there are many relevant objects to be recommended to many novel students. Moreover, the sparsity of the matrix is relevant to recommendations. Sparsity is related to the number of student/object interactions present in the dataset. A number close to zero means that there are no interactions while a number close to one means that all interactions were reached. The overall sparsity is calculated by Equation (1).

$$\rho = 1 - \left(\frac{number\ of\ not\ null\ elements}{number\ of\ elements}\right) \tag{1}$$

The sparsity of the built matrix is 0.99339978, or about 99%. This is a high sparsity and the greater the sparsity the greater is the difficulty of explaining and predicting student behavior (Bobadilla et al., 2009; Nafea et al., 2019; Wang et al., 2015). This is justified by the fact that with few interactions the built model tends to be less accurate and does not predict the following student/object interaction. Fig. 3 shows the overall sparsity of the transposed raw dataset to the built matrix.

The sparse matrix, shown in Fig. 3, represents the overall behavior of student/object interactions. Nearby zero, no interactions; nearby 10, high interactions (at legend). Interactions, in this context, are related to the time spent by the student in a specific object. Moreover, we investigate the sparsity through the Principal Component Analysis (PCA) technique, and the concentration of interactions' student/object lies to a few students and objects. This perspective is shown in Fig. 4.

Thus, the dataset is a typical sparse matrix consumed by recommendation systems. The dataset presented outliers, which were removed, long tail phenomenon, and noises created by the careless student/object interaction. Moreover, the input was normalized by min/max normalization (Wang et al., 2015). The summary of the dataset is shown in Table 3.

As shown in Table 3 the interaction between 3,757 students and 5,104 objects produces a matrix with 19,175,728 elements and 0.9939978 of sparsity, or approximately, 99% of sparsity.

3.2. The data split technique

We use the holdout data split strategy to provide a sub dataset to train, which contains 80% of all the observations, and another sub dataset to test, which contains the remaining 20% of the observations. Both sub datasets were sampled by simple random sampling without repetition technique. Thus, 3,757 students' observations were divided into 3,005 (around 80%) observations to train and 751 (around 20%) to test. The goals of sampling are to use a procedure that is likely to yield a representative sample of the population (i.e., to limit exposure to sampling error), while holding down sampling costs as much as possible.

3.3. The algorithm

The Deep Auto Encoder (DAE) is a class of artificial neural networks architectures (Sedhain et al., 2015). Thus, it is possible to use almost all hyperparameters to optimize the built model. We investigate the variations of values of the hyperparameters to identify the suitable ones and obtain the more accurate results. The main output of a model built by a DAE is to predict the missing values of the sparse matrix. Using Table 2 as input, an example of output is shown in Table 4.

To obtain the more accurate results of the built model to fill out the missing values of the sparse matrix, we adopted a strategy to optimize the hyperparameters. One way to identify a proper value for a hyperparameter is by determining a range of values for it (Ordway-West et al., 2018). The range can often be deduced from the context. At this point, a process can sample the range evenly and manipulate the value, which allows the model to achieve the most satisfactory performance. When this approach is employed for several hyperparameters at the same time, it is frequently entitled to a grid search (Baldi, 2011; Ordway-West et al., 2018). For each hyperparameter, it chooses a list of values and trains the model with all combinations of the values. This approach works effectively for some hyperparameters, but the number of models to be trained grows rapidly with the number of hyperparameters. This becomes especially problematic for models which include a considerable number of hyperparameters and take a long time to train. This happens with neural networks. The hyperparameters are shown in Table 5.

The range column, in row activation, can assume many types of functions. Some examples are: Binary Step, Linear Activation, Non-Linear, Sigmoid, Hyperbolic Tangent, RELU, SoftMax, and Swish. We used the SELU. The Scaled Exponential Linear Unit (SELU) is an activation function of the most recent ones. To employ this function, it is necessary to use *lecun_normal* to initialize the weight, and if abandonment is desired. The SELU function is shown in Equation (2).

$$SELU(x) = \begin{cases} x, & x > 0 \\ \alpha e^{x} - \alpha, & x \leq 0 \end{cases} \tag{2}$$

That is, if the input value x is greater than zero, the output value becomes x multiplied by lambda λ. If the input value x is less than or equal to zero, we carry out a function that goes up to ∞, which is our output y, when x is zero. Essentially, when x is less than zero, you multiply α by the exponential value x minus the value α and multiply it by the value λ. Besides, the division of the dataset to train and test was 80/

Table 5
The set of hyperparameters used in the experiments.

Hyperparameter	Meaning	Range	CDAE*	DAE-CF	DAE-CI
Activation	Function used by the activation of neuron	Many	SELU	SELU	SELU
Batch Size	The size of sampler present to the network	[1:N]	64	64	64
Dropout	Rate of unused neurons	[0:1]	0.2	0.2	0.2
Epoch	Number of the iterations to train the network	[1:∞]	50	50	50
Factors	Max number of neurons in each layer	[1:∞]	500	15	15
Layers	Number of layers	[1:∞]	[128,256,128]	[512,256,512]	[512,256,512]
Learning Rate	Rate of updating the synapses weights	[0:1]	0.0001	0.0001	0.0001
Regularization	Rate to avoiding overfitting	[0:1]	0.0001	0.01	0.01

*The CDAE, DAE-CF, and DAE-CI represent the values optimized in class of DAE. The number of layers is chosen according to architecture.

20, respectively, as explained before.

3.4. The metrics

The recommendation problem is a generalization of regression as follows (Sedhain et al., 2015; Strub et al., 2016; Li and She, 2017; Chae et al., 2019). In regression, the data are in the form of a matrix with columns representing resources and a column that represents the target variable. The algorithm predicts the values of the target variable based on the values of the predictors. In the recommendation (especially in collaborative filtering), the data are in a matrix with columns representing items. It is equally valid to invert the matrix and produces the columns that represent users. The problem remains the same. There are no distinctions between the columns. The goal is to fill in the missing values in any column. Because of this connection, many of the basic metrics for evaluating regression can be applied. For example, an average square error can be calculated. However, as the typical objective is to obtain the main list of recommendations, these metrics do not reasonably match the expectations of the recommendation system. Then, we investigated the use of metrics from the students' point of view.

The time and attention of students are small. Thus, recommending all objects, even though in an ordered way, is not a suitable strategy. Rather than, recommend the k first objects is more effective. Thus, increasing the learning rate on suitable learning objects represent the most important goal of an adaptive e-learning system. The first metric used to measure the uselessness of k first objects to all students is called Mean Average Precision (MAP). This metric is defined by Equation (3).

$$MAP@K = \frac{1}{|U|}\sum_{u=1}|U|\ \frac{1}{M}\sum_{k=1}^{N} P_u(k) \times rel_u(k) \tag{3}$$

This metric presents the average precision to recommend k objects to a student. Moreover, the average, which is computed to all students, presents the mean of recommendation of k objects to all students. The MAP metric is important to evaluate the ability of the recommendation model to achieve the most appropriate recommendation to all students. As follows, MAP represents both sparse matrix dimensions: students and objects.

Another key metric is Normalized Discounted Cumulative Gain (NDCG). The NDGC is a measure of the gain of an object based on its position in the result list. The underlying assumptions are that; relevant objects are more useful if shown earlier in the result list and highly relevant objects (HRO) are more useful than moderately relevant objects (MRO), which is more useful than irrelevant objects (IO). Thus, HRO $>$ MRO $>$ IO. Each recommendation on the list has a relevance score associated with it; cumulative gain represents the sum of all relevant scores in the recommendation set. Moreover, the position of the score is significant because two recommendation sets $\mathbf{A} = \mathbf{\{3, 3, 2, 1\}}$ and $\mathbf{B} = \mathbf{\{3, 2, 1, 3\}}$ have the same score, even if they present different sequences of objects. Thus, the measure involves discounting the relevance score by dividing it with the *log* of the corresponding position. However, it is still not complete. Depending on several factors, the number of recommendations may vary for each student. At that time, normalization is required. The NDCG is computed by Equation (4).

$$NDCG@K(\mu, \omega) = \sum_{k=1}^{K} \frac{2^{|\omega(k) e l_\mu|} - 1}{log_2(K+1)} \tag{4}$$

The cut-off version of NDCG is NDCG@K, where K represents the number of objects evaluated in the recommendation list. Compared to the MAP metric NDCG carries out an important job at evaluating the position of ranked items. It operates beyond the binary relevant/non-relevant scenario. Thus, both metrics are complementary, allowing the interpretation of the results for different dimensions.

Personalization (P) is a metric to assess if a model recommends many of the same objects to different students. It is the dissimilarity (1 − cosine similarity) between students' lists of recommendations. A high personalization score indicates students' recommendations are different, meaning the model is offering a personalized experience to each one. The P metric is represented by Equation (5).

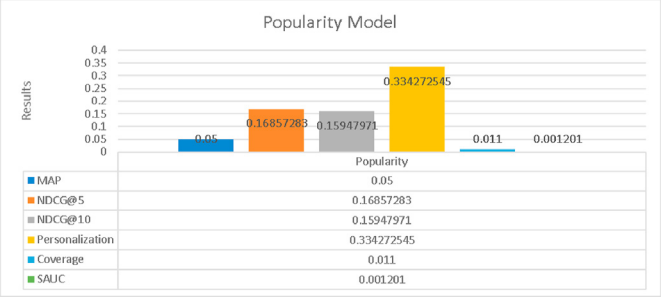


Fig. 5. The res were provided and adaptepularity Model[4].

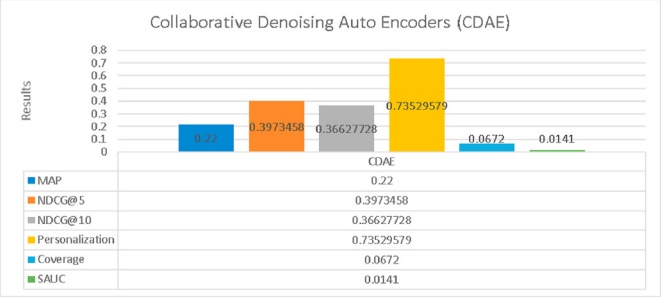


Fig. 6. The evaluation metrics of the Collaborative Denoising Auto Encoders (CDAE).

$$P = \sum_{u=1}^{U} \frac{(1 - cosine\ similarity(u))}{U} \tag{5}$$

Thus, P metric offers a way to compare if the recommendation model is presenting the same recommendations list for each student. The metrics MAP, NDCG, and P present, respectively, the ability of the model to recommend relevant objects, recommend top k ordered objects and the level of personalization of the model built. For all these metrics, the closer the results get to 1, the better the model becomes.

Serendipity is a complex concept which includes other ones: (a) relevance, (b) novelty, and (c) unexpectedness. We define a relevant LO as an LO that a student likes, consumes, or is interested in. The term novel may have different meanings that can be summarized as: (a) novel item, a recently added item that users have not yet rated; (b) forgotten item, an item a user might have forgotten that she/he consumed some time ago; (c) unknown item, an item that a user has never consumed in his/her life; and (d) unrated item, an item that a user has not rated yet regardless of whether she/he is familiar with the item. We used the term novel item corresponding to definition unknown item due to the popularity of the definition in studies on serendipity (Denis et al., 2016). Unexpectedness and surprise are terms frequently used in the literature (Denis et al., 2016), but to the best of our knowledge there is no consensus on the terms' definitions. In this paper, the terms unexpected and surprising refer to LO that significantly differ from the profile of the student regardless of how novel or relevant those LO are. Thus, serendipity might be a way to expand the knowledge of students providing content which is different but relevant. Also, this strategy might keep the focus of students avoiding that they become bored with obvious suggestions that they might have already discovered.

[4] These figures were provided and adapted by ML Flow, an open source platform for the machine learning lifecycle. For more details, please visit https://mlflow.org/

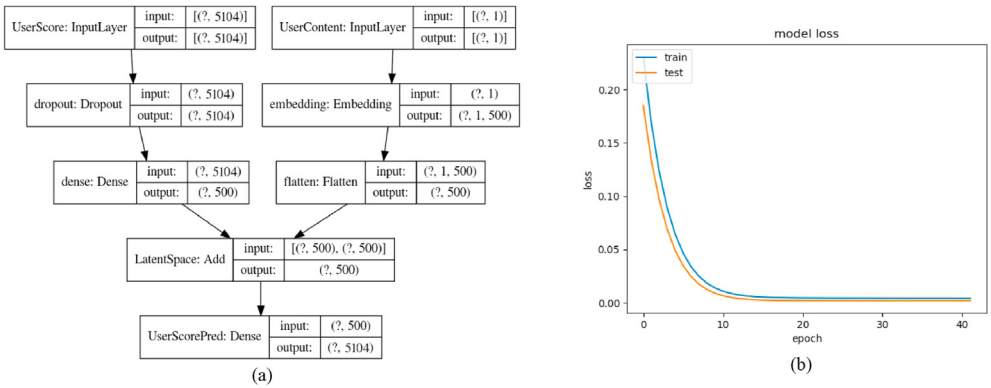


Fig. 7. The built model (a) and training history (b) of the Collaborative Denoising Auto Encoders (CDAE).

Denis et al. (2016) suggested a Serendipitous Personalized Ranking Algorithm (SPR), based on Singular Value Decomposition (SVD). The objective of SPR is to maximize the serendipitous area under the ROC (receiver operating characteristic) curve (SAUC). SAUC is based on Area Under the Curve (AUC) and is defined as follows (Denis et al., 2016) (Equation (6)):

$$SAUC(u) = \frac{1}{\left|I_u^+\right|\left|I \setminus I_u^+\right|} \sum_{i \in I_u^+} \sum_{j \in I \setminus I_u^+} \sigma\left(\hat{r}_{u,i} - \hat{r}_{u,j}\right) \left(pop(j)_{\ j}\right)^{\alpha} \tag{6}$$

where σ is a 0–1 loss function, (σ(x) = 1 if x > 0, σ(x) = 0 otherwise), while $\hat{r}_{u,i}$ is a preference prediction of student *u* for LO_i. Relevant items are represented by I_u^+, consequently, $I \setminus I_u^+$ corresponds to unrated and irrelevant LOs. Popularity weight corresponds to *pop(i)*, *pop(i)* $\infty |U_i|$.

Finally, Coverage represents the percent of items in the training data that the model can recommend on a test set. For example, the recommendation list R = {O_1, O_2, …, O_k}, where *k* is the number of top@k learning objects, for each student is computed considering distinct objects recommended to all students. Thus, this metric represents how effective is the built model to recommend all the available learning objects.

4. Results and discussions

In this section we present the results of experiments using 3 class of DA: Collaborative Denoising Auto Encoders (CDAE), Deep Auto Encoders for Collaborative Filtering (DAE-CF), Deep Auto Encoders for Collaborative Filtering using Content Information (DAE-CI). The Popularity Model experiment is also used for comparison among the three models. These comparisons were made by using the metrics Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG), Serendipity (SAUC) from Serendipitous Personalized Ranking Algorithm (SPR), Personalization, and Coverage to identify the performance from built models related to recommender systems metrics. Also, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) were used to recognize the performance related to continuous targets outputs. All these metrics were presented and formalized in the previous section.

4.1. Popularity model

This model performs recommendations presenting the most popular learning objects. It does not provide adaptability and shows the same recommendation list to all students. The recommendation list is composed of the sum of time spent in each object, in descending order; therefore, the most visualized learning object, in time, is the first one, the second most visualized is the second on the list, and so on. Therefore, this model is not related to each student, being an overall recommendation of all learning objects. The metrics of this model are shown in Fig. 5.

As shown in Fig. 5, the P metric is around 0.334; i.e., this value represents 33% of adaptability and it is not therefore useful. Besides, the metrics MAP, NDCG@5, NDCG@10, and coverage also present low value[5]. However, this model may represent an approach to overcome the cold-start problem to new students. This developed model does not require hyperparameters and is simple to build and explain.

Table 6
The top@10 recommendation list provided by the built model using the Collaborative Denoising Auto Encoders (CDAE).

	score	object_id	learning_object
0	0.737318	4572	Learning Object 52
1	0.677891	3509	Learning Object 418
2	0.633031	1262	Learning Object 214
3	0.619094	501	Learning Object 145
4	0.612064	2	Learning Object 10
5	0.592862	3188	Learning Object 389
6	0.588873	2631	Learning Object 338
7	0.579694	4705	Learning Object 64
8	0.565154	4601	Learning Object 546
9	0.557259	4509	Learning Object 51

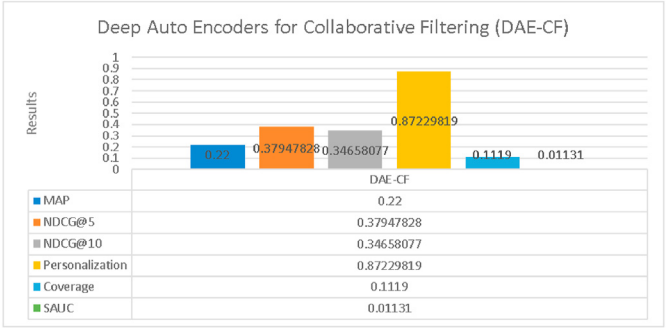


Fig. 8. The evaluation metrics of the Deep Auto Encoders for Collaborative Filtering (DAE-CF).

4.2. Collaborative Denoising Auto Encoders (CDAE)

The CDAE assumes that the interactions of student/object represent a corrupted form of a complete matrix. Thus, this Auto Encoder aims to reconstruct the matrix considering the missing values as corrupted interactions. The CDAE is equally represented as a one-hidden-layer neural network. In the input layer, there are a total of O + 1 nodes, where each of the firsts O nodes corresponds to an object, and the last node is a

[5] We considered the best case of the all built models with their hyperparameters. The worst and average cases were suppressed because they did not present expressive differences related to the best case.

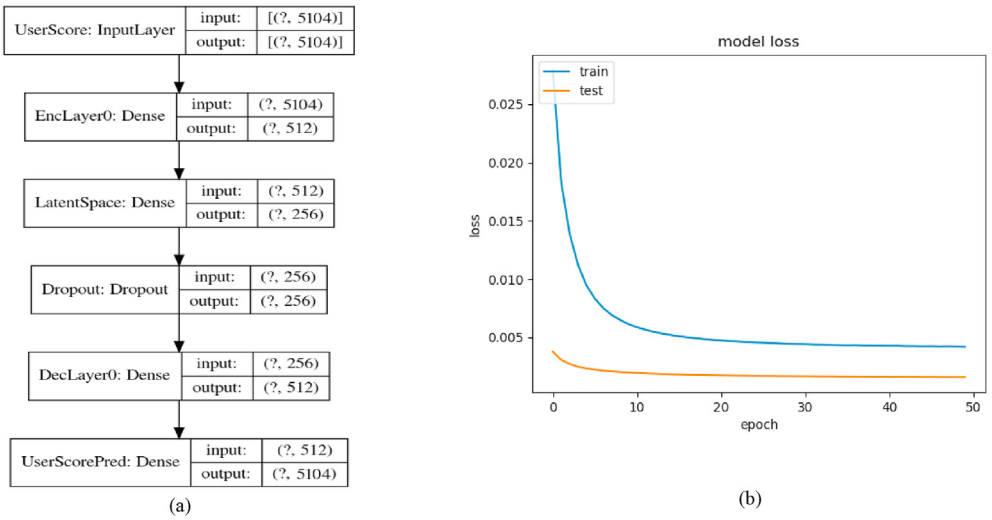


Fig. 9. The built model (a) and training history (b) of the Deep Auto Encoders for Collaborative Filtering (DAE-CF).

student-specific node which means the node and its associated weights are unique for each student s ∈ S in the data. The metrics of this model are shown in Fig. 6.

As shown in Fig. 6, the P metric is around 0.733; i.e., this value represents 73% of adaptability. Also, the metrics MAP, NDCG@5, NDCG@10, and coverage presented showed important increases. Thus, related to the Popularity Model, the CDAE provides a significant improvement in adaptability of the learning objects to the preferences of the student. The built model and the history of training are shown in Fig. 7.

The built model allows eight layers: UserScore, UserContent, Dropout, Embedding, Dense, Flatten, LatentSpace, and UserScorePred. All of them have input and output sub layers. The UserScore is responsible for getting all 4,856 interactions for each student with the learning objects. For instance, student A has the interaction vector V = {10, NA, …, 5}, where each position of the vector represents the time spent by the student with the object. NA represents the lack of interaction. The UserContent represents the layer that contains the O + 1 nodes, as explained before. In this manner, the default value is 1. The Dropout layer represents the number of unnecessary neurons that will be removed. The Embedding layer groups the O + 1 nodes into a cluster based on factors hyperparameter (Table 5). The Dense and Flatten layers compress the previous layers into the LatentSpace. Finally, this last layer decodes the LatentSpace into UserScorePred; which contains the preference of each student. The recommendation list, to student 10, is showing in Table 6.

As shown in Table 6, the column *score* presents the relevancy of the learning object to student 10. The first columns represent the position of the object in the recommendation list.

4.3. Deep Auto Encoders for Collaborative Filtering (DAE-CF)

The DAE-CF demonstrates the tendency to generalize beyond CDAE or other kinds of architecture with few layers. Otherwise, this architecture uses the nonlinear activation functions and techniques to avoid overfitting, such as regularization and dropout. The built model by DAE-CF presents an improvement around 10% when compared to CDAE. This improvement is due, specially, to the deep layers. The metrics of this model are shown in Fig. 9.

As shown in Fig. 8, the P metric is around 0.872; i.e., this value represents around 87% of adaptability. Also, the metrics MAP, NDCG@5, NDCG@10, and coverage presented a significant increase. Thus, related to CDAE, the DAE-CF provides a significant improvement in adaptability of the learning objects to the preferences of the student. The build model and the history of training is shown in Fig. 9.

The built model includes six layers; UserScore, EncLayer0, LatentSpace, Dropout, DecLayer0, and UserScorePred. All of them have the input and output sub layers. The UserScore is the same as CDAE. The EncLayer0 is the first layer to encode the previous layer into a factored matrix. The number 0 in the name represents the first layer; thus, the second layer has extension 1, the third extension 2, and so on. The LatentSpace characterizes the finishing stage of efficient compression of sparse matrix present to the UserScore layer. The Dropout layer represents the number of unnecessary neurons that will be removed. Although more precisely, it does not mean that such neurons are unnecessary. The pruning process is done at random, that is, there is a probability of eventual removal of neurons that are useful to the model. The DecLayer0 represents the first layer that starts decoding the LatentSpace into a subsequent layer. Finally, this last layer decodes the LatentSpace into UserScorePred, which contains the preference of each student. The recommendation list, to student 10, is showing in Table 7.

As shown in Table 7, the column *score* presents the relevancy of the learning object to the student 10. The first columns represent the position of the object into the recommendation list.

4.4. Deep Auto Encoders for Collaborative Filtering using content information (DAE-CI)

The DAE-CI is a variation of DAE-CF adding content information. As follows, this architecture is a hybrid implementation. To build the model, the information store in the *learning_object* includes collaborative filtering. This information represents the interaction of each student with learning objects and is used to include content information to the student level. The model built by DAE-CI worsens around 5% when compared to

Table 7
The top@10 recommendation list provided by the built model using the Deep Auto Encoders for Collaborative Filtering (DAE-CF).

	score	object_id	learning_object
0	0.842951	3188	Learning Object 389
1	0.828034	2	Learning Object 10
2	0.794385	4675	Learning Object 612
3	0.764501	4572	Learning Object 52
4	0.728598	1328	Learning Object 22
5	0.683430	501	Learning Object 145
6	0.606181	1262	Learning Object 214
7	0.583904	1992	Learning Object 28
8	0.568082	972	Learning Object 188
9	0.552525	445	Learning Object 14

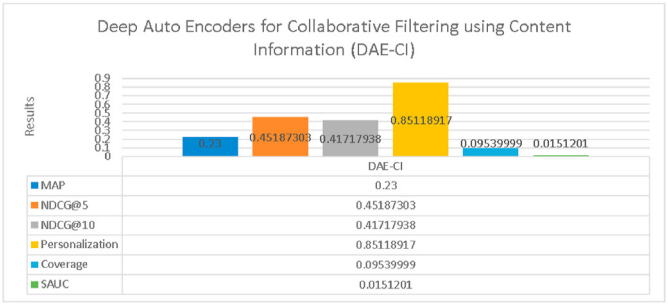


Fig. 10. The evaluation metrics of the Deep Auto Encoders for Collaborative Filtering using Content Information (DAE-CI).

Table 8
The top@10 recommendation list provided by the built model using the Deep Auto Encoders for Collaborative Filtering using Content Information (DAE-CI).

	score	object_id	learning_object
0	1.278175	4401	Learning Object 5
1	1.191606	555	Learning Object 150
2	1.040922	4594	Learning Object 54
3	0.927504	4994	Learning Object 90
4	0.918778	4391	Learning Object 499
5	0.897038	202	Learning Object 118
6	0.888085	4738	Learning Object 67
7	0.886164	4402	Learning Object 50
8	0.865007	136	Learning Object 112
9	0.858167	4885	Learning Object 801

DAE-CF. This decrease is due, especially, to the duplicate information included. The *learning_object* does not provide varied information and, in that case, reduces the accuracy of the built model. However, if we utilize more information into the dataset, this architecture may represent an interesting option. The metrics of this model are shown in Fig. 10.

As shown in Fig. 6, the P metric is around 0.851; i.e., this value represents around 85% of adaptability. Also, the metrics MAP, NDCG@5, NDCG@10, and coverage presented an important increase. Thus, related to DAE-CF, the DAE-CI has a lower performance in adaptability of the learning objects to the preferences of the student. This decrease in the performance might be explained by the values anonymized into the *learning_object* feature. The values were replaced by "Learning Object N", where N is an integer variable which represents an increment into different learning objects found. Thus, this approach does not provide a suitable differentiation between learning objects. The built model and the history of training is shown in Fig. 11.

The built model allows eleven layers: ItemContent, Embedding, Flatten, ItemLatentSpace, UserScore, EncLayer0, UserLantentSpace, LatentSpace, Dropout, DecLayer0, and UserScorePred. All of them have input and output sub layers. The ItemContent layer acquires the furthest information from content. The Embedding and Flatten layers compress the content of all objects into ItemLatentSpace. The UserScore, EncLayer0, UserLantentSpace, LatentSpace, Dropout, DecLayer0, and UserScorePred layers provide the same objective of previous models. The recommendation list, to student 10, is showing in Table 8.

As shown in Table 8, the column *score* presents the relevancy of the learning object to the student 10. The first columns represent the position of the object into the recommendation list.

4.5. Discussion

Table 9 summarizes the results of the metrics to the four built models. The most representative value is highlighted in bold. Both Deep Auto Encoders (DAE-CF and DAE-CI) present better results. The fundamental reason for this result is related to the number of deep layers which allows the built model to fit better the input with the target output. Besides, these additional layers might be modified by using different activation functions, such as RELU rather than SELU (Section 3.3). This provides a deep learning of data and can generalize better than CDAE.

As shown in Fig. 12, the DAE-CI presents better results in the metrics NDCG@5 and NDCG@10 compared to DAE-CF. This means the built

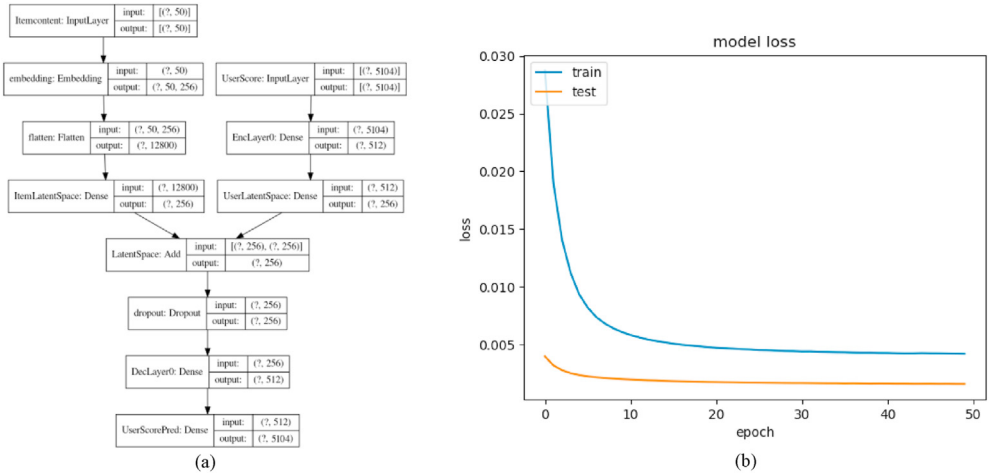


Fig. 11. The built (a) and training history (b) of the Deep Auto Encoders for Collaborative Filtering using Content Information (DAE-CI).

Table 9
Resulting metrics presented in the experiments.

Metric	Popularity	CDAE	DAE-CF	DAE-CI
MAP	0.05	0.22	0.22	**0.23**
NDCG@5	0.16857283	0.3973458	0.37947828	**0.45187303**
NDCG@10	0.15947971	0.36627728	0.34658077	**0.41717938**
Personalization	0.33427254509	0.73529579	**0.87229819**	0.85118917
Coverage	0.011000000000	0.0672	**0.1119**	0.09539999
SAUC	0.001201000	0.014100000	0.011310000	**0.0151201000**

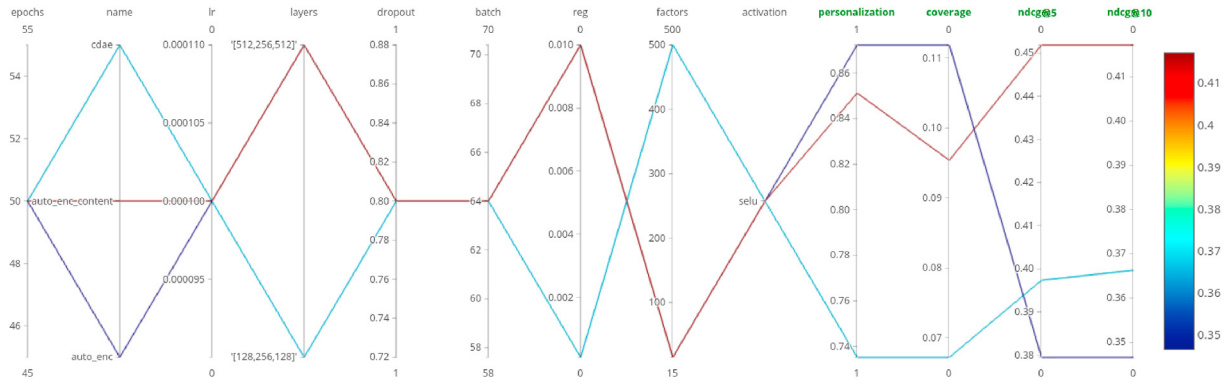
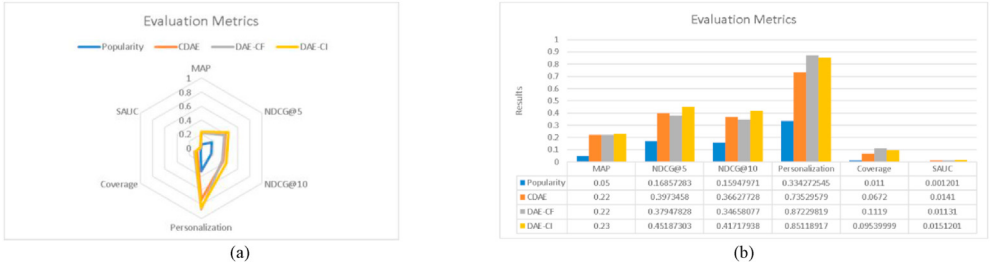
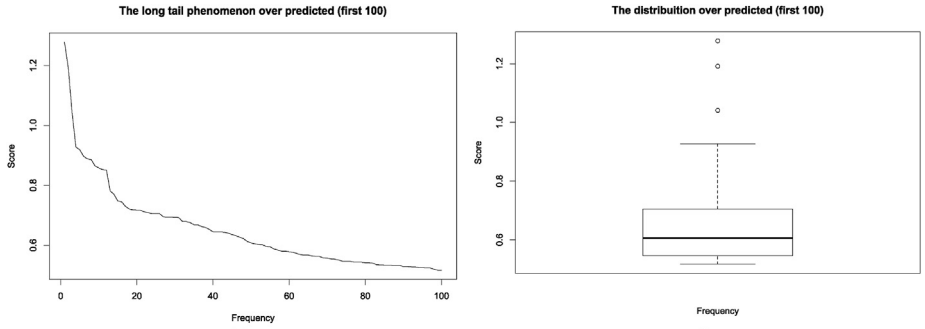


Fig. 12. The results of CDAE, DAE-CF, and DAE-CI related to the metrics NDCG@5, NDCG@10, Personalization, and Coverage.



(a) (b)

Fig. 13. The results of Popularity Model, CDAE, DAE-CF, and DAE-CI to the metrics MAP, NDCG@5, NDCG@10, Personalization, Coverage, and SAUC. In (a) the coverage of each built model. In (b) the comparison between metrics.



(a) (b)

Fig. 14. The behaviors of the score related to objects (a) and frequency (b). It is important to note that the long tail phenomenon occur, also, in the score metric. This an evidence that the resulting recommendation list is based on this phenomenon.

model is more effective at recommending learning objects based on its position; i.e. more relevant objects at the top of list. Conversely, DAE-CF built models are more effective at offering high adaptability for each student and covering more learning objects into the repository.

With respect to the Popularity Model, the Auto Encoder can offer an elevated level of adaptability and recommendations. Broadly speaking, the metrics are improved with the use of the Auto Encoder, especially deep ones. The deep layers increase the ability of the neural network to generalize and offer more relevant recommendations for each user. Fig. 13 shows the comparison between the four developed models over the evaluation metrics.

Furthermore, we investigate the behavior of scores presented on the recommendation list. Its behavior is exponential with the firsts objects owning the high score and decaying after five ones. Additionally, the frequency of the score is like a long tail phenomenon, many objects with low relevance and few ones with high relevance. These behaviors are indicated in Fig. 14.

In conclusion, we present a concept called preference signature. The essential idea behind that is providing the most granular class of adaptability, building a preference signature for each student. This signature presents his/her learning preference in order to receive the learning objects. Table 10 shows the input of the student 10 (the interactions of the student with the firsts 100 learning objects).

Table 10
The list with interactions of the student 10 with the firsts 100 learning objects.

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	1.2	0.0	0.0	0.0	0.0
0.0	1.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.7
0.0	0.0	0.0	0.0	0.0	0.0	2.9	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	5.5	0.0	0.0	0.0	0.0	0.0
1.2	0.0	6.8	0.0	0.0	8.5	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Based on these results, it is possible to build a preference signature for

Table 11
The preference signature of the student 10 with the firsts 100 learning objects (this table was filled by the auto encoder with the entry of table 10).

1.2781755	1.1916058	1.0409219	0.9275036	0.9187776	0.8970384	0.8880845	0.8861635	0.8650075	0.8581665
0.8520871	0.8513905	0.7809544	0.7697405	0.7473381	0.7446624	0.7296307	0.7205836	0.7175525	0.7169326
0.7160413	0.7109861	0.7085849	0.7049262	0.7049110	0.7047389	0.6942966	0.6936854	0.6934537	0.6927639
0.6918837	0.6788331	0.6788318	0.6752425	0.6682408	0.6679753	0.6619343	0.6592174	0.6525638	0.6443356
0.6440960	0.6437874	0.6431290	0.6398829	0.6353902	0.6314064	0.6264353	0.6218643	0.6126253	0.6077056
0.6044728	0.6026431	0.6014149	0.5958386	0.5949369	0.5876704	0.5841696	0.5802777	0.5798503	0.5780840
0.5769091	0.5725408	0.5686121	0.5669655	0.5669616	0.5644537	0.5626519	0.5621841	0.5579706	0.5566605
0.5543797	0.5534146	0.5504782	0.5466366	0.5464506	0.5463822	0.5435224	0.5432371	0.5431470	0.5417905
0.5412520	0.5399067	0.5347409	0.5340176	0.5332670	0.5327663	0.5324275	0.5322449	0.5320356	0.5290830
0.5289596	0.5277051	0.5271041	0.5255498	0.5249646	0.5245923	0.5244858	0.5196915	0.5168995	0.5164964

Table 12
The comparison of the previous works. Many of them present endogenous metrics.

Author	Year	Summary	Method	Metrics
Bobadilla et al. (2009)	2009	Memory-based method to calculate the proximity of content of learning objects using KNN	Memory-based	MAE = 2%
Hwang et al., (2013)	2013	Content-based method to context-aware recommendation of learning objects	Content-based	Specific metric proposed by authors
Krauss (2016)	2016	An approach to extend the traditional user-item-matrix of a recommendation engine by a third dimension – the time value	Cosine similarity	Without explicit metrics
Protasiewicz et al. (2016)	2016	The developed system helps to collect data characterizing potential reviewers, retrieving information from relational and unstructured data, and formulating a set of recommendations	Cosine similarity	Precision = 97.96% Recall = 54.27% F-score = 69.84%
Barbieri et al. (2017)	2017	The authors propose switching SVD to a Stacked Denoising Auto encoder to avoid the noise of users' evaluations	Stacked Denoising Auto encoder	MAE and RMSE = 70% in the best case
Nafea et al. (2019)	2019	The authors combine students actual rating with their learning styles to recommend personalized course learning objects (LOs)	Cosine similarity	MAE = 22% and RMSE = 38% in the best case
Medio et al. (2020)	2020	This paper proposes a hybrid recommender system	Content filtering and Collaborative filtering	Precision = 22% Recall = 39% F1 = 28%
Our	2020	Three classes of deep auto encoder to fill out a sparse matrix composed by the interactions of student/object in a MOOC	Deep Auto Encoder	MAE = 81% in the best case RMSE = 86% in the best case

each student. This signature is the output of Deep Auto Encoder. For example, the vector of all LOs from student 10 which is 0.0 is replaced by the probability of that student consuming that LO. This preference signature, to the firsts 100 learning objects, is shown in Table 11.

Therefore, the use of Deep Auto Encoder (DEA-CF/CI) can represent an approach to build adaptive e-learning recommendation system models. This class of artificial neural networks allows building high personalized models to recommend relevant objects, covering many objects into the repository, and providing a preference signature to each student based on his/her previous interactions. Moreover, this class of architecture is relevant to overcome the challenges to provide a recommendation list, including objects with relevance (supported by the metrics MAP and NDCG), and also recommending objects with novelty, serendipity, and diversity (supported by the metric Coverage and SAUC). In conclusion, this class of architectures improves the adaptability, providing a preference signature for each student (supported by Personalization).

Our results can be compared with previous works. Many of them are focused on endogenous metrics like Precision, Recall, F-Score, and RMSE. We proposed not only endogenous, but also metrics to provide evaluation of more effective models. Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are two of the most common metrics used to measure accuracy for continuous variables, thus, we used them to compare with other authors. The comparison of previous works is present in Table 12.

These results are shown in a compiled way, in Fig. 15. Broadly speaking, the metrics are sparse, i.e., used by one work but not done by another. The best results, to Precision, Recall, and F-Score were achieved by Protasiewicz et al. (2016).

As shown in Fig. 15, our results, based on metrics MAE and RMSE, presented an improvement related to Barbieri (Barbieri et al., 2017) of around 11% and 15%, respectively. We did not use the metrics Precision, Recall, and F-Score because the target output is continuous.

4.6. Educational implications

From a pragmatic point of view, our approach helps to improve the outcomes of students. The reasons for this improvement can be attributed to some features of our approach. The first feature is relevance; as discussed, an LO needs to be relevant or it is useless. Based on students' signature of preference it is possible to present relevant LOs provided by LORs and with curricular restrictions (for example, in a lesson A the LO_a and LO_b are mandatory but LO_c and LO_d are elective. Besides, the top@k of the LOs can be used to control the volume of content to be assimilated by students. The second feature is novelty; an LO needs to be new or not used in the past. Using the sparse matrix is possible to identify what LOs were used before and to recommend a new one. However, to present previous LO might be a way to improve cognition by repetition and this can be controlled by our approach. Third one, serendipity; LO, which is, not expect, but relevant, to the student. This can be done to keep the focus because something new might be present at any time. The last feature is diversity; diverse uses types of LO for improving the coverage of the subject to be learned. This can be used to increment the knowledge, widely and deeper, from a subject (Hwang et al., 2013; Krauss, 2016; Protasiewicz et al., 2016; Barbieri et al., 2017; Nafea et al., 2019; Medio et al., 2020; Gomede et al., 2020; Sedhain et al., 2015; Strub et al., 2016). Also, the learning objects can be standardized by use of Shareable Content Object Reference Model (SCORM) and provided by a Learning Objects Repository (LOR). The learning objects can be created by professors (or other professionals) and deployed into a LOR, then, our

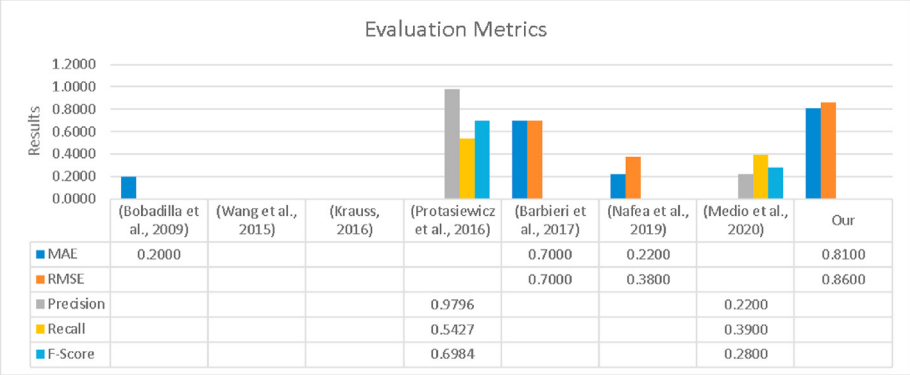


Fig. 15. The comparison of the previous works with our results. Many of them present endogenous metrics.

approach can retrieve them from many LORs.

5. Conclusions, practical impacts, and further works

This work aims at contributing with an adaptive e-learning recommendation system with which we investigated four approaches to develop models. The first one, Popularity Model, represents the base of comparison with the other ones. The models designed by Auto Encoders were compared using relevant metrics and with the hyperparameters optimized. The class of Deep Auto Encoders shows better results using deep layers. The Auto Encoders for Collaborative Filtering (DAE-CF) presented better personalization and coverage of learning objects. Finally, Deep Auto Encoders for Collaborative Filtering using Content Information (DAE-CI) presented better recommendation lists of learning objects. All these models can be applied to the recommendation and adaptability problem and to build the preference signature of each student.

As practical impacts, we may highlight the use of Deep Auto Encoders to build adaptive e-learning recommendation systems to improve the experience and the outcomes of students. This may increase the results of the educational environment, helping teachers, tutors, administrators, researchers, and/or parents (Gomede, Gaffo, Briganó, Barros, & Mendes, 2018). This approach enables teachers and/or tutors to better recognize their students and provide more accurate content in an adapted format. It enables administrators to improve systems management to get high grades for their institutions. Also, it enables researchers to obtain more relevant content for their research and avoid information flooding. Finally, it enables parents to follow their children's preferences and help them achieve better results.

In conclusion, it is possible to explore additional recommendation approaches for other educational agents. To teachers, we can search for new adaptive ways to deeply present their subject matter to their students; to students, we can look for new adaptive forms to have problems with increased difficult levels presented to them; to researchers, we can look for new open and relevant problems; and to parents, we can search to present forms to help them follow their children and participate in leading them to a more thorough experience with the activity of learning. In terms of engineering, we might explore the use of Learning Objects Repositories to share many Learning Objects from several subjects, levels, idioms, cultures, and so on to improve the quality of recommendations.

Author contributions

This paper was developed with the active contribution of the doctorate candidate Everton Gomede, the orientation of Leonardo Mendes, and Rodolfo Barros. Mendes and Barros are both advisors of the candidate Gomede, and besides heading the research group, are the main responsible for the propositions and hypotheses being proposed and tested. Mendes research areas are in Intelligent Cities and applied problems in Computational Intelligence. Barros main research fields are in Governance, Strategic Planning and applied Computational Intelligence. Everton Gomede's main research field is applied Computational Intelligence for Smart Cities. His main contribution for the paper was in the processing of data with computational intelligence techniques and mathematical models. All participants gave important contributions to the formulation of the hypotheses and the definition of the data analytics for the educational dataset. Based on The CRediT Roles, the contributions are Gomede: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, and Writing – original draft; Barros and Mendes: Project administration, Resources, Supervision, Validation, Writing – review & editing.

Statements on open data and ethics

The participants were protected by hiding their personal information in this study. They were voluntary and they knew that they could withdraw from the experiment at any time. Also, the name of learning objects was suppressed to avoid issues related to privacy (GDPR). The data can be provided upon requests by sending e-mails to the corresponding author.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.caeai.2021.100009.

References

Baldi, P. (2011). Autoencoders, unsupervised learning, and deep architectures. In *UTLW'11: Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning workshop, 27* pp. 37–50).

Barbieri, J., Alvim, L. G. M., Braida, F., & Zimbrão, G. (2017). Autoencoders and recommender systems: COFILS approach. *Expert Syst. Appl., 89*, 81–90. https://doi.org/10.1016/j.eswa.2017.07.030

Bobadilla, J., Serradilla, F., & Hernando, A. (2009). Collaborative filtering adapted to recommender systems of e-learning. *Knowl. Base Syst., 22*(4), 261–265. https://doi.org/10.1016/j.knosys.2009.01.008

Chae, D. K., Kim, S. W., & Lee, J. T. (2019). Autoencoder-based personalized ranking framework unifying explicit and implicit feedback for accurate top-N recommendation. *Knowl. Base Syst., 176*, 110–121. https://doi.org/10.1016/j.knosys.2019.03.026

Denis, K., Shuaiqiang, W., & Jari, V. (2016). A survey of serendipity in recommender systems. *Knowl. Base Syst., 111*, 180–192. https://doi.org/10.1016/j.knosys.2016.08.014

Dockterman, D. (2018). Insights from 200+ years of personalized learning. *NPJ Sci. Learn., 3*(1), 1–6. https://doi.org/10.1038/s41539-018-0033-x

Göbel, S., & Mehm, F. (2013). *Personalized, adaptive digital educational games using narrative game-based learning objects. Entertainment for Education Digital Techniques and Systems-Lecture Notes in Computer Science, 6249*. https://doi.org/10.1007/978-3-642-14533-9_45

Gomede, E., Gaffo, F. H., Briganó, G. U., Barros, R. M., & Mendes, L. D. S. (2018). Application of computational intelligence to improve education in smart cities. *Sensors, 18*(1), 267. https://doi.org/10.3390/s18010267

Gomede, E., Barros, R. M., & Mendes, L. S. M. (2020). Use of deep multi-target prediction to identify learning styles. *Appl. Sci., 10*(5), 1756. https://doi.org/10.3390/app10051756

Hwang, G., Yang, L., & Wang, S. (2013). A concept map-embedded educational computer game for improving students' learning performance in natural science courses. *Comput. Educ., 69*, 121–130. https://doi.org/10.1016/j.compedu.2013.07.008

Krauss, C. (2016). Smart learning: time-dependent context-aware learning object recommendations. In *The 29th AAAI FLAIRS Conference of the Florida Artificial Intelligence Research Society, 5* pp. 501–504).

Li, X., & She, J. (2017). Collaborative variational autoencoder for recommender systems. In *KDD '17: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 305–314). https://doi.org/10.1145/3097983.3098077

Medio, C., Limongelli, C., Sciarrone, F., & Temperini, M. (2020). MoodleREC: A recommendation system for creating courses using the moodle e-learning platform. *Computers in Human Behavior, 104*, 106168 https://doi.org/10.1016/j.chb.2019.106168.

Nafea, S. M., Siewe, F., & He, Y. (2019). On recommendation of learning objects using Felder-Silverman Learning Style Model. *IEEE Access, 7*, 163034–163048. https://doi.org/10.1109/ACCESS.2019.2935417

Ordway-West, E., Parveen, P., & Henslee, A. (2018). Autoencoder evaluation and hyperparameter tuning in an unsupervised setting. In *2018 IEEE International Congress on Big Data (Big Data Congress)* (pp. 205–209). https://doi.org/10.1109/BigDataCongress.2018.00034

Protasiewicz, J., Pedrycz, W., Kozłowski, M., Dadas, S., Stanisławek, T., Kopacz, A., & Gałe,zewska, M. (2016). A recommender system of reviewers and experts in reviewing problems. *Knowl. Base Syst., 106*, 164–178. https://doi.org/10.1016/j.knosys.2016.05.041

Sedhain, S., Menon, A. K., Sanner, S., & Xie, L. (2015). Autorec: autoencoders meet collaborative filtering. In *WWW '15 Companion: Proceedings of the 24th International Conference on World Wide Web* (pp. 111–112). https://doi.org/10.1145/2740908.2742726

Shih, J. L., Chuang, C. W., & Hwang, G. J. (2010). An inquiry-based mobile learning approach to enhancing social science learning effectiveness. *Journal of Educational Technology & Society, 13*(4), 50–62.

Shih, J. L., Chuang, C. W., & Hwang, G. J. (2010). An inquiry-based mobile learning approach to enhancing social science learning effectiveness. *J. Educ. Technol. Soc., 13*(4), 50–62.

Strub, F., Gaudel, R., & Mary, J. (2016). Hybrid recommender system based on autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (pp. 11–16). https://doi.org/10.1145/2988450.2988456

Wang, H., Wang, N., & Yeung, D. Y. (2015). Collaborative deep learning for recommender systems. In *KDD '15: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1235–1244). https://doi.org/10.1145/2783258.2783273