Existing Machine Learning Techniques for Knowledge Tracing: A Review Using the PRISMA Guidelines

Sergio Iván Ramírez Luelmo(✉), Nour El Mawas, and Jean Heutte

CIREL – Centre Interuniversitaire de Recherche en Éducation de Lille, Université de Lille, Villeneuve d'Ascq, France
{sergio.ramirez-luelmo,nour.el-mawas,jean.heutte}@univ-lille.fr

Abstract. Nowadays, Machine Learning (ML) techniques play an increasingly important role in educational settings such as behavioral academic pattern recognition, educational resources suggestion, competences and skills prediction, or clustering students with similar learning characteristics, among others. Knowledge Tracing (KT) allows modelling the learner's mastery of skills and to predict student's performance by tracking within the Learner Model (LM) the students' knowledge. Based on the PRISMA method, we survey and describe commonly used ML techniques employed for KT shown in 51 articles on the topic, among 628 publications from 5 renowned academic sources. We identify and review relevant aspects of ML for KT in LM that contribute to a more accurate panorama of the topic and hence, help to choose an appropriate ML technique for KT in LM. This work is dedicated to MOOC designers/providers, pedagogical engineers and researchers who need an overview of existing ML techniques for KT in LM.

Keywords: Machine Learning · Knowledge Tracing · Learner Model · Technology enhanced learning · Literature review · PRISMA

1 Introduction

Evidence from several studies has long linked having a Learner Model (LM) can make a system more effective in helping students learn, and adaptive to learner's differences [1].

LMs represent the system's beliefs about the learner's specific characteristics, relevant to the educational practice [2], encoded using a specific set of dimensions [3]. Ultimately, a perfect, ideal LM would include all features of the user's behavior and knowledge that effect their learning and performance [4]. Modelling the learner has the ultimate goal of allowing the adaptation and personalization of environments and learning activities [5] while considering the unique and heterogeneous needs of learners. We acknowledge the difference between Learner Profile (LP) and LM in that the former can be considered an instantiation of the latter in a given moment of time [6].

Knowledge Tracing (KT) models students' knowledge as they correctly or incorrectly answer exercises [7], or more generally, based on observed outcomes on their previous

B. Csapó and J. Uhomoibhi (Eds.): CSEDU 2021, CCIS 1624, pp. 73–94, 2022.
https://doi.org/10.1007/978-3-031-14756-2_5

practices [8]. KT is one out of three approaches for student performance prediction [9]. In an Adaptive Educational System (AES), predicting students' performance warrants for KT. This allows for learning programs recommendation and/or level-appropriate, educational resources personalization, and immediate feedback. KT facilitates personalized guidance for students, focusing on strengthening their skills on unknown or less familiar concepts, hence assisting teachers in the teaching process [10].

Machine Learning (ML) is a branch (or subset) of Artificial Intelligence (AI) focused on building applications that learn from data and improve their accuracy over time without being programmed to do so [11]. To achieve this, ML algorithms build a model based on sample data (a.k.a. input data) known as 'training data'. Once trained, this model can then be reused with other data to make predictions or decisions.

ML techniques are currently applied to KT in vast and different forms. The goal of this literature review is to survey all available works in the field of "Machine Learning for Knowledge Tracing used in a Learner Model setup" in the last five years to identify the most employed ML techniques and their relevant aspects. This is, in general terms, what common ML techniques and aspects, designed to trace a learner's mastery of knowledge, also account for the creation, storage, and update of a LM. Moreover, we aim to identify relevant ML aspects to consider insuring KT in a LM. The motivation behind this work is to present a comprehensive panorama on the topic of ML for KT in LM to our target public. To our knowledge, currently there is no research work that addresses the literature review of ML techniques for KT accounting for the LM.

Thus, we decided to focus our literature review on the terms "machine learning", "knowledge tracing" and "learner model", a.k.a. "student model" (SM). Using the PRISMA method [12], we performed this research in the IEEE, Science Direct, Scopus, Springer, and Web of Science databases comprising the 2015–2020 period. The thought behind these choices is to obtain the most recent and high-quality corpus on the topic.

This work differs from other literature reviews [13–15] on two accounts. First, we focus exclusively on ML techniques for KT accounting for the LM. That is, we do not cover pure Data Mining (DM) techniques, nor AI intended for purposes other than KT, such as Natural Language Processing (NLP), gamification, computer vision, learning styles prediction, nor any processes that make pure use of LP data (instead of LM data), nor other User Model data, such as sociodemographic, biometrical, behavioral, or geographical data[1]. Second, we do not compare the mathematical inner workings of ML techniques: we feel that our target public might be unable to exploit appropriately such complex form results. Instead, we shift the focus to a pragmatic report on ML for KT in a LM application and purpose(s).

The remainder of this article is structured as follows. Section 1 of this paper oversees the theoretical framework concerning this paper, namely the definition of ML and its categorization. Section 3.3 details the methodology steps taken. Section 5 presents the findings of this research, Sect. 5 discusses the results and, finally Sect. 5 concludes this paper and presents its perspectives.

[1] Please note that we did include such works, if they also employed ML for KT (the core of this paper).

2 Theoretical Background

In this section we present the theoretical background put in motion behind this research, namely the definition of ML and how it is categorized.

2.1 Machine Learning

ML is a branch (or subset) of AI focused on building applications that learn from data and improve their accuracy over time without being programmed to do so [11]. Additional research [16, 17] to this definition allows us to present Fig. 1 to illustrate and discern the situation of ML against other common terms used in the field.

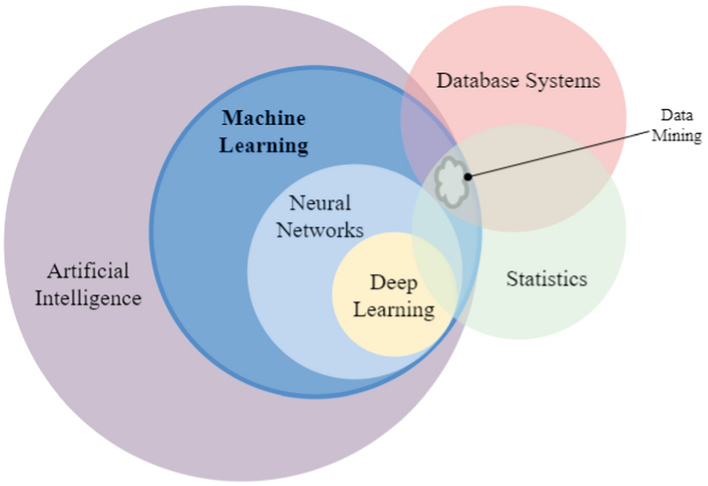


Fig. 1. Situational context of ML [18].

2.1.1 ML Methods/Styles/Scenarios

Although some authors [13, 19] admit several more ML methods (or styles or paradigms or scenarios), we retain the following categorization: Supervised ML, Semi Supervised ML, Unsupervised ML, Reinforcement Learning, and Deep Learning [11]. The first three differentiate each other on the labelling of the input training data while creating the model. The two latter constitute special cases altogether [11, 19, 20].

First, in **Supervised Learning (SL)** labels are provided (metadata containing information that the model can use to determine how to classify it). However, properly labelled data is expensive[2] to prepare, and there is a risk of creating a model so tied to its training data that it cannot handle variations in new input data accurately ("overfitting") [20].

Second, **Unsupervised Learning (UL)** must use algorithms to extract meaningful features to label, sort and classify its training data (which is unlabeled) without human

[2] Mostly in terms of computational resource allocation.

intervention. As such, it is usually used to identify patterns and relationships (that a human can miss) than to automate decisions and predictions. Because of this, UL requires huge amounts of training data to create a useful model [20].

Third, **Semi Supervised Learning (SSL)** is at the middle point of the two previous methods: it uses a smaller labelled dataset to extract features and guide the classification of a larger, unlabeled dataset. It is usually used when not enough labelled data is made available (or it is too expensive) to train a preferred, Supervised Model [21].

Fourth, **Reinforcement Learning (RL)** is a behavioral machine learning model akin to SL, but the algorithm is not trained using sample data but by using trial and error. A sequence of successful outcomes will be reinforced to develop the best recommendation or policy for a given problem. RL models can also be deep learning models [11].

Lastly, **Deep Learning (DL)** is a subset of ML (all DL is ML, but not all ML is DL). DL algorithms define an artificial neural network[3] that is designed to learn the way the human brain learns. DL models require a large amount of data to pass through multiple layers of calculations, applying weights and biases in each successive layer to continually adjust and improve the outcomes. DL models are typically unsupervised or semi-supervised [11]. For clarity reasons, the figure illustrating this ML categorization is available in the Appendix.

In this subsection we covered the ML definition and a categorization of ML techniques. In the following subsection we deepen into the relevant aspects in ML for KT in LM.

2.1.2 ML for KT in LM

An overwhelming number of ML techniques have been designed and introduced over the years [13]. They usually rely on more common ML techniques, within optimized pipelines. As such, we identify the **ML techniques** (or algorithms) upon which any new research is based.

Additionally to performing KT in LM, researchers have acknowledged that ML techniques can reliably determine the initial parameters when instantiating a LM [23, 24]. This led us to consider this purpose when reviewing ML techniques. Different ML techniques are applied at different stages of the ML pipeline, and not all stages are responsible for KT (other applications can be NLP, computer vision, automatic grading, demographic student clustering, mood detection, etc.) We differentiate purposes related to KT and/or learner modelling, specifically if the ML technique is used for (1) either grade, skills, or knowledge prediction (and hence later, clustering, personalizing, or suggesting resources), (2) either for LM creation (or instantiation), or (3) both.

Studies highlight the importance of justifying the **rationale** when choosing a ML technique [25–27]. We note such rationale, when made explicit, and contrast it to other authors' rationale for commonalities, on the same technique. This allows us to weigh and present known, favorable, and unfavorable features specific to ML techniques applied to KT accounting for the LM.

Research studies stress the ultimate importance of the input data (**dataset**) and the effects of the chosen programming language **software** employed for ML [25, 28]. Indeed,

[3] A quite complete and updated chart of many neural networks was made available by [22].

ML techniques require input data for creating a model. The feature engineering of this input data (dataset) might be determinant for a ML project to succeed or fail [25]. We list and verify the availability of all public datasets presented in the reviewed articles. Furthermore, the choice of the programming language for ML plays a role in collaboration, licensing, and decision-making processes: it helps to determine the most appropriate choices for ML implementation (purchasing licenses, upgrading hardware, hiring a specialist, or considering self-training). Hence, we highlight the family of ML programming languages used by researchers on their proposals.

Thus, based on this state-of-the-art, we identify relevant aspects to consider in ML for KT in LM: the **ML technique** employed, its **purpose**, the contextual, known **rationale** for choosing it, the **programming language software** used for ML, and the **dataset**(s) employed for KT. We consider that these aspects are relevant for our target public when choosing a ML technique for KT in LM.

3 Review Methodology

This review of literature follows the PRISMA [12] methodology, comprising: Rationale, Objectives & Research questions, Eligibility criteria, Information sources & Search strategy, Screening process & Study selection, and Data collection & Features.

3.1 Rationale, Objectives and Research Questions

The goal of this literature review is to present a comprehensive panorama on the topic of ML for KT in LM. This is, in general terms, what ML techniques designed to trace a learner's mastery of skill also account for the creation, storage, and update of the LM.

This article aims thus to answer the following two research questions (RQ):

- RQ1: What are the most employed ML techniques for KT in LM?
- RQ2: How do the most employed ML techniques fulfil the considered relevant aspects to insure KT in LM?

3.2 Eligibility Criteria, Information Sources and Search Strategy

In this section we describe the inclusion and exclusion criteria used to constitute the corpus of publications for our analysis. We also detail and justify our choice of in-scope publications, the search terms, and the identified databases.

In this research, we focus on recent ML techniques (and/or algorithms) that explicitly "learn" (with minimal or no human intervention) from its data input to perform KT, while accounting for the LM. Thus, we do not cover all predictive statistical methods (as they are not all ML), nor pure DM techniques, nor AI intended for purposes other than KT (such as NLP, gamification, computer vision, learning styles prediction, etc.), nor any processes that make pure use of LP data (instead of LM data), nor other User Model data, such as sociodemographic, biometrical, behavioral, or geographical data.

On one hand, our Inclusion criterion are: Works that present a ML technique for KT while accounting for the LM, in the terms presented in the previous paragraph. On the other hand, our chosen Exclusion criterion consist of: Works written not in English, under embargo, not published or in the works. We choose to keep subsequent works on the same subject from the same research team because they represent a consolidation of the techniques employed.

We performed this research at the end of October 2020 in the following scientific databases: IEEE, Science Direct, Scopus, Springer, and Web of Science, comprising 2015–2020. The thought behind these two choices is to have the most recent and quality-proven scientific works on the subject. Our general search terms were:

```
(("learner model" OR "student model" OR "knowledge trac-
ing") AND "machine learning")
```

They were declined for the specificities of each scientific database (search engines parse and return verbal, noun, plural, and continuous forms of search terms). We used their 'Advanced search' function, or we queried them directly, if they allowed it. Some direct queries did not allow for year filtering, so we applied it manually on the results page. For accessibility reasons, we explicitly selected "Subscribed content" results for the scientific databases supporting it.

3.3 Screening Process and Study Selection

The paper selection process happened as follows: First, we gathered all the results in two known Citation Manager programs to benefit from the automatic metadata extraction, the report creation, and duplicate merging. We also used a spreadsheet to record, based on Sect. 2.1.2, the following information: doi, title, year, purpose, ml_method, method_rationale, software, data_source, and observations. Second, we screened the abstracts of all 708 results: three categories appeared: obvious Out-of-scope results, clear Eligible results, and Pending (verification needed) results. Third, using the institutional authentication, we downloaded all the papers in the Eligible and Pending categories. Fourth, we read the full papers in the Eligible and Pending categories and re-classified them as Eligible or Out-of-scope, as needed (Fig. 2).

3.4 Data Collection and Features

In this section we review the relevant features of interest described in Subsect. 2.1.2 found in the reviewed literature.

During the full text read, we extracted the following information from the selected papers: (1) ML technique employed; (2) purpose of the ML technique; (3) rationale for employing that specific ML technique; (4) software employed for ML; and (5) dataset employed for KT, if any.

We note here that rarely a single, known technique ML is employed, but it is rather implemented in a pipeline, connected with another secondary ML (probabilistical, or DM) techniques. In such cases, we focused on the technique(s) employed for KT and on the reasons given for choosing it over other techniques acknowledged by the authors.

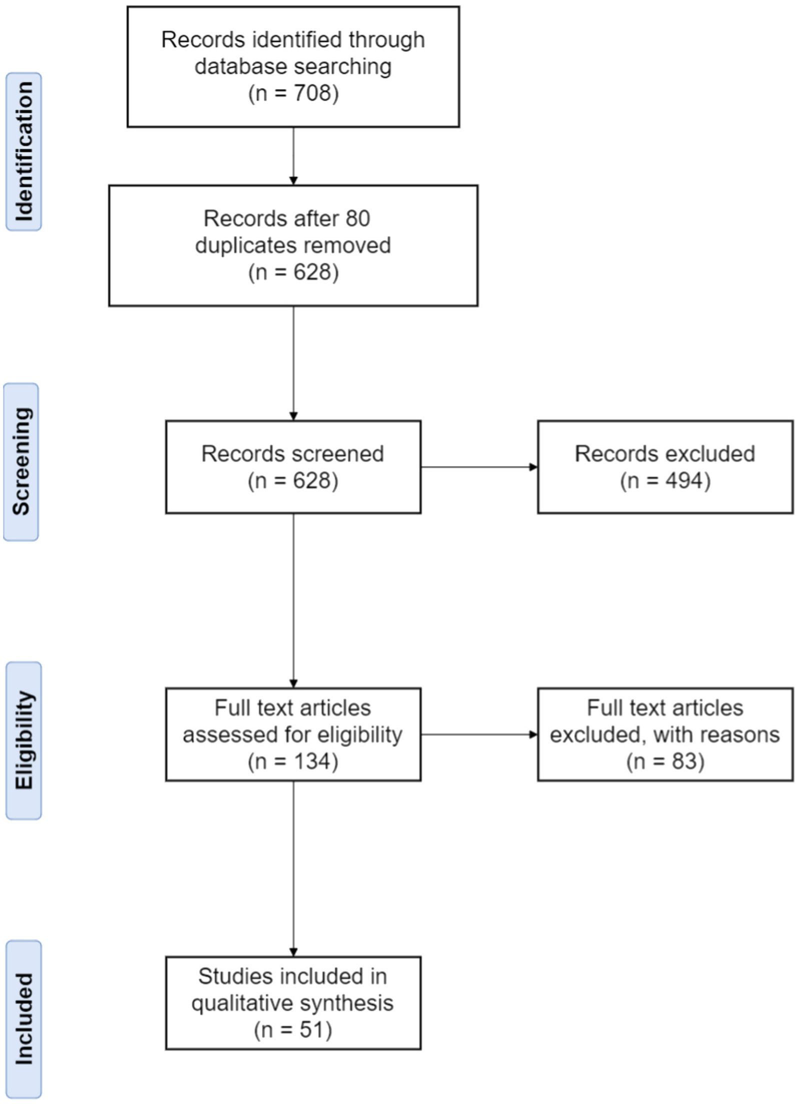

Identification
Screening
Eligibility
Included
Records identified through database searching (n = 708)
Records after 80 duplicates removed (n = 628)
Records screened (n = 628)
Records excluded (n = 494)
Full text articles assessed for eligibility (n = 134)
Full text articles excluded, with reasons (n = 83)
Studies included in qualitative synthesis (n = 51)

Fig. 2. PRISMA Flow diagram of the publication screening process [18].

We surveyed the software used to perform the calculation of ML and we grouped them by programming language, which is a rather meaningful description, compared to combinations of libraries and platforms. We think this result shows a clear tendency on the necessary requirements to implement and perform ML for KT in LM.

We surveyed all datasets presented in the 51 reviewed papers and checked for their existence. We understand that our target public may not have data made available to perform ML for KT accounting for the LM and we feel that this resource may be invaluable when evaluating their results.

In this section we presented our literature review methodology, the considered features, and the train of thought behind them. The following section details our literature review results.

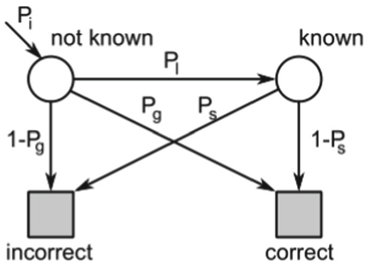
4 Results

We aggregated the data collected (described in the previous section) to make it easier to digest.

First, we present the seven most employed[4] ML techniques for KT in LM found in the reviewed publications. These comprise based-upon techniques for the paper proposal, techniques used as starting bases within a pipeline, and techniques employed when comparing ML techniques.

Bayesian Knowledge Tracing (BKT) [8] is the most classical method used to trace students' knowledge states. It is a special case of a Hidden Markov Model (HMM) [29]. In BKT, skill is modeled as a binary variable (known/unknown) and learning is modeled by a discrete transition from an unknown to a known state. The basic structure of the model, as well as its update and prediction equations are depicted in Fig. 3, the probability of being in the known state is updated using a Bayes rule based on an observed answer. The basic BKT model uses the following data [9, 29, 30]:

- Global learner data: P_i is the probability that the skill is initially learned (also known as *p-init,* or *P(L*$_0$*)*), P_g is the probability of a correct answer when the skill is unlearned (a guess, a.k.a. *p-guess* or *P(G)*), P_s is the probability of an incorrect answer when the skill is learned (a slip, a.k.a. *p-slip* or *P(s)*), and P_l is the probability of learning a skill in one step (a.k.a *p-transit, P(T)*), assumed constant over time.
- Local learner data: probability θ that a learner is in the known state.
- Global domain data: a definition of Knowledge Components (KCs) (sets of items). There are no relations among KCs, i.e., the parameters for individual KCs are independent.



update equation:

if $c = 1$: $\theta' := \frac{\theta(1-P_s)}{\theta(1-P_s)+(1-\theta)P_g}$

if $c = 0$: $\theta' := \frac{\theta P_s}{\theta P_s+(1-\theta)(1-P_g)}$

$\theta := \theta' + (1-\theta')P_l$

prediction equation:

$P_{correct} = P_g \cdot \theta + (1-P_s) \cdot (1-\theta)$

Fig. 3. Basic structure and equations for the BKT model [29] (*c* is the correctness of an observed answer).

[4] With more than five applications in the last five years.

Parameter fitting for the global learner parameters (the tuple P$_i$, P$_l$, P$_s$, P$_g$) is typically done using the standard expectation-maximization algorithm, alternatively using a stochastic gradient descent or discretized brute-force search. The specification of KC is typically done manually, potentially using an analysis of learning curves [9, 29].

Deep Knowledge Tracing (DKT) was proposed by [31] to trace students' knowledge using Recurrent Neural Networks (RNNs), achieving great improvement on the prediction accuracy of students' performance. It uses a Long Short-term Memory (see next item) to represent the latent knowledge space of students dynamically. DKT uses large numbers of artificial neurons for representing latent knowledge state along with a temporal dynamic structure and allows a model to learn the latent knowledge state from data [32]. It is defined by the following equations [31, 32]:

$$h_t = \tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{1}$$

$$y_t = \sigma\left(W_{yh}h_t + b_y\right) \tag{2}$$

In DKT, both tanh and the sigmoid function (σ) are applied element wise and parameterized by an input weight matrix W$_{hx}$, recurrent weight matrix W$_{hh}$, initial state h$_0$, and readout weight matrix W$_{yh}$. Biases for latent and readout units are represented by b$_h$ and b$_y$ [31, 32].

Long Short-Term (LSTM) is a variant of RNN, effective in capturing underlying temporal structures in time series data and long-term dependencies more effectively than conventional RNN [33]. LSTM builds up memory by feeding the previous hidden state as an additional input into the subsequent step. While typical RNN consist of a chain of repeating modules of NN, in LSTM, instead of having a single NN layer, there are three major interacting components: forget, input, and output (i_t, f_t, and o_t, respectively) [33]. In LSTM, latent units retain their values until explicitly cleared by the action of the 'Forget gate'. Thus, they retain more naturally information for many time steps, which is believed to make them easier to train. Additionally, hidden units are updated using multiplicative interactions, and they can thus perform more complicated transformations for the same number of latent units. This makes the model particularly suitable for modeling dynamic information in student modeling, where there are strong statistical dependencies between student learning events over long-time intervals. The equations for an LSTM are significantly more complicated than for an RNN [31]:

$$i_t = \sigma\left(W_i\left[x_t, h_{t-1}\right] + b_i\right) \tag{3}$$

$$f_t = \sigma\left(W_f\left[x_t, h_{t-1}\right] + b_f\right) \tag{4}$$

$$o_t = \sigma\left(W_o\left[x_t, h_{t-1}\right] + b_o\right) \tag{5}$$

$$c\_in_t = \tanh\left(W_{c\_in}\left[x_t, h_{t-1}\right] + b_{c\_in}\right) \tag{6}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot c\_in_t \tag{7}$$

$$h_t = o_t \odot \tanh(c_t) \tag{8}$$

$$k_t = \sigma(W_{kh}h_t + b_k) \tag{9}$$

Although LSTM has a certain capability of learning relatively long-range dependency, it still has trouble remembering long-term information [34].

Bayesian Networks (BNs) are graphical models designed to explicitly represent conditional independence among random variables of interest and exploit this information to reduce the complexity of probabilistic inference [35]. They are a formalism for reasoning under uncertainty that has been widely adopted in AI [36]. Formally, a Bayesian network is a directed acyclic graph where nodes represent random variables and links represent direct dependencies among these variables.

If we associate to each node X_i in the network a Conditional Probability Table (CPT) that specifies the probability distribution of the associated random variable given its immediate parent nodes parents (X_i), then the BN provides a compact representation of the Joint Probability Distribution (JPD) over all the variables in the network:

$$P(X_1, \ldots, X_n) = \prod\nolimits_{i=1}^{n} P(X_i|Parents(X_i)) \tag{10}$$

A few examples of simple BN and their associated equations are shown in Fig. 4.

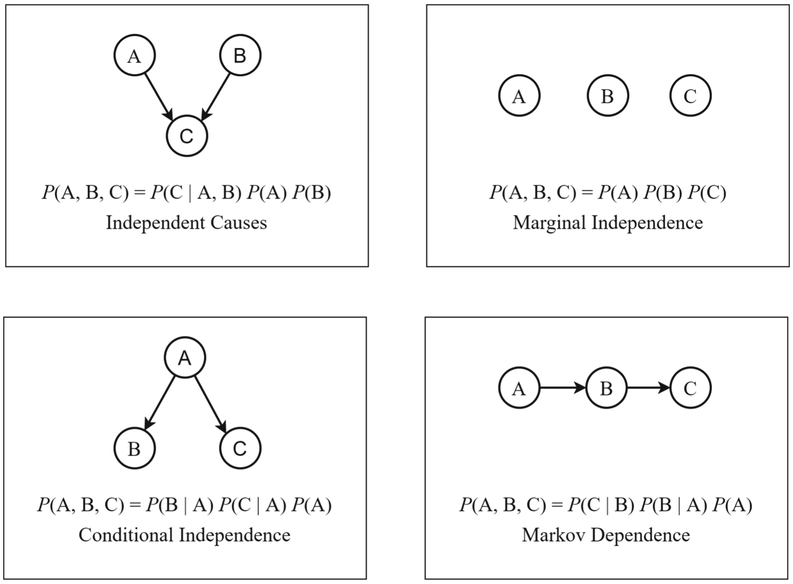

A
B
C
P(A, B, C) = P(C | A, B) P(A) P(B)
Independent Causes
A
B
C
P(A, B, C) = P(A) P(B) P(C)
Marginal Independence
A
B
C
P(A, B, C) = P(B | A) P(C | A) P(A)
Conditional Independence
A
B
C
P(A, B, C) = P(C | B) P(B | A) P(A)
Markov Dependence

Fig. 4. Examples of BN and their associated equations.

Static BNs track the belief over the state of variables that don't change over time as new evidence is collected, i.e., the posterior probability distribution of the variables given the evidence. Dynamic BNs on the other hand, track the posterior probability of variables whose value change overtime given sequences of relevant observations [36].

Support Vector Machines (SVM) are one of the most robust prediction methods, based on statistical learning frameworks [37]. The primary aim of this technique is to map nonlinear separable samples onto another higher dimensional space by using different types of kernel functions. The underlying idea is that when the data is mapped to a higher dimension, the classes become linearly separable [38]. SVM try to reduce the probability of misclassification by maximizing the distance between two class boundaries (positive vs. negative) in data [39]. Assume that a dataset used for training is represented by a set $j = \{(x_i, y_i)\}_{i=1}^{l}$, where $(x_i, y_i) \in R^{n+1}$, l is the number of samples, n is the number of features and a class label $y_i = \{-1, 1\}$. The separating hyperplane, defined by the parameters w and b, can be obtained by solving the following convex optimization problem [40, 41]:

$$\min \frac{1}{2}\|w\|^2 \tag{11}$$

$$s.t.\ y_i\left(w^T \varphi(x_i) + b\right) \geq 1 \qquad i = 1, 2, \ldots\ldots, l \tag{12}$$

For actualizing SVM for more than two classes two possible strategies can be used: One-Against-All (OAA) and One-Against-One (OAO). In OAA, to unravel an issue of n classes, n binary problems are solved rather than fathoming a single issue. Each classifier is basically used to classify one single class; that is why values on that specific class will grant positive response and [data] points on other classes will give negative values on that classifier. In the case of OAO, for n course issues, $\frac{n(n-1)}{2}$ SVM classifiers are built and each of them is prepared to partition one class from another. When classifying an unknown point, each SVM votes for a class and the class with most extreme votes is considered as the ultimate result [41].

They key advantage of SVM is that they always find the global minimum because there are no local optima in maximizing the margin. Another benefit is that the accuracy does not depend on the dimensionality of data [38]. This is a clear advantage when the class boundary is non-linear, as other classification techniques will produce too complex models for non-linear boundaries [38]. They distinctively afford balanced predictive performance, even in studies where sample sizes may be limited.

Dynamic Key Value Memory Network (DKVMN) is a Memory Augment Neural Network-based model (MANN), which uses the relationship between the underlying knowledge points to directly output the student's mastery of each knowledge point [42]. DKVMN uses *key-value* pairs rather than a single matrix for the memory structure. Instead of attending, reading, and writing to the same memory matrix in MANN, the DKVMN model attends input to the *key* component, which is immutable, and reads and writes to the corresponding *value* component [42]:

At each timestamp, DKVMN takes a discrete exercise tag q_t, outputs the probability of response $p(r_t|q_t)$, and then updates the memory with an exercise-and-response tuple (q_t, r_t). Here, q_t also comes from a set with Q distinct exercise tags and r_t is a binary value. [42] affirms that there are N latent concepts $\{c^1, c^2, \ldots, c^N\}$ underlying the exercises, which are stored in the *key* matrix M^k (size $N \times d_k$) whereas the students' mastery levels of each concept $\left\{s_t^1, s_t^2, \ldots, s_t^N\right\}$ (concept states) are stored in the *value* matrix M_t^v (size $N \times d_v$), which changes over time [42]. Thus, DKVMN traces the knowledge of a student by reading and writing to the *value* matrix using the correlation *weight* computed from the input exercise and the *key* matrix. Equations for the read and write process can be found in detail in [42].

Performance Factor Analysis (PFA) [43] is one specific model from a larger class of models based on a logistic function [29]. In PFA, the data about learner performance are used to compute a skill estimate. Then, this estimate is transformed using a logistic function into the estimate of the probability of a correct answer. The update and prediction equations are depicted in Fig. 5. The PFA model uses the following data [29]:

- Global learner data: parameters γ_k, δ_k specifying the change of skill associated with correct and wrong answers for a given KC_k.
- Local learner data: a skill estimate θ_k for each KC_k.
- Global domain data: a KC difficulty parameter β_k, a Q-matrix Q specifying item-KC mapping; $Q_{ik} \in \{0, 1\}$ denotes whether an item i belongs to KC_k.

$$\theta_k := \theta_k + Q_{ik}\big(\gamma c + \delta(1-c)\big) \text{ for each } k. \tag{13}$$

$$P_{correct} = \frac{1}{(1+e^{-m})}, \text{ where } m = \sum_k Q_{ik}(\beta_k + \theta_k). \tag{14}$$

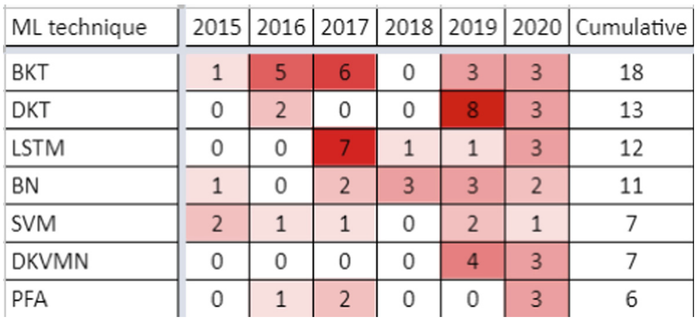
Fig. 5. Update (θ_k) and prediction ($P_{correct}$) equations for the PFA model, according to [29] (c is the correctness of an answer, i is the index of an item).

Parameter fitting for parameters β, γ, δ is usually done using standard logistic regression. The Q-matrix is also usually manually specified, but can be also fitted using automated techniques like matrix factorization [29].

This list answers then RQ1. "What are the most employed ML techniques for KT in LM?". Figure 6 shows a yearly heatmap of the most used techniques: the number indicates the total number of applications[5] in all 51 combined-and-reviewed papers, per year. DKT was applied eight times in 2019 (emerging from two consecutive zero years) while BKT was mostly applied in 2016 and 2017, five and six times respectively, decreasing since. LSTM peaked in 2017, with 7 applications, and has decreased since. BN remains with a steady application since 2017. For clarity reasons, the 29 ML techniques found in the 51 papers issued from this study are available in the Appendix.

Second, we noted the rationale (if any) given by authors when choosing a ML technique. We do not account for the rationale of the paper's unique ML proposal if its improvements are related to parameter fine-tuning, or if the justification is à posteriori. Instead, we account rationale for the general application of the original, unmodified technique. Also, very few publications detail the shortcomings of their choice. We grouped these rationales (Fig. 7) in the following categories:

[5] Programming and teaching the ML model with input data.



ML technique	2015	2016	2017	2018	2019	2020	Cumulative
BKT	1	5	6	0	3	3	18
DKT	0	2	0	0	8	3	13
LSTM	0	0	7	1	1	3	12
BN	1	0	2	3	3	2	11
SVM	2	1	1	0	2	1	7
DKVMN	0	0	0	0	4	3	7
PFA	0	1	2	0	0	3	6

Fig. 6. Yearly heatmap of the most employed ML techniques [18].

R1-Uses Less Data and/or Metadata. These techniques handle sparse data situations better compared to others, according to the authors, e.g. DKT [44].

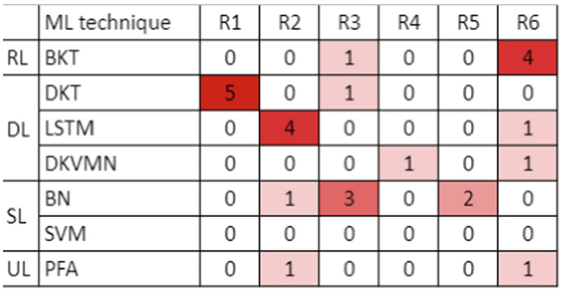
R2-Extended Tracing. These techniques provide additional attributes and/or dimensional tracing with ease when compared to other techniques, according to authors, e.g. LSTM [45].

R3-Popularity. These techniques were chosen because of their popularity, e.g. BN [24].

R4-Persistent Data Storage. These techniques explicitly save their intermediate states to long-term memory, e.g. DKVMN [46].

R5-Input Data Limitations. These techniques have been acknowledged to lack when the number of peers is "too high", e.g. BN [47].

R6-Modelling Shortcomings. Techniques in this category face difficulties when modelling either forgetting, guessing, multiple-skill questions, time-related issues, or have other modelling shortcomings, e.g. BKT [48].



	ML technique	R1	R2	R3	R4	R5	R6
RL	BKT	0	0	1	0	0	4
DL	DKT	5	0	1	0	0	0
	LSTM	0	4	0	0	0	1
	DKVMN	0	0	0	1	0	1
SL	BN	0	1	3	0	2	0
	SVM	0	0	0	0	0	0
UL	PFA	0	1	0	0	0	1

Fig. 7. Heatmap of most employed ML techniques, categorized by Method (SL, UL, SSL, RL, DL) and number of publications sharing a Rationale (R1–R6) [18].

A heatmap illustrating the number of publications mentioning each of these rationales, for each of the most common ML techniques, is shown in Fig. 4. This heatmap includes the ML categorization presented in Sect. 2.1.1 (SL, UL, SSL, RL, DL).

BKT faced mostly R6 rationales (four occurrences) and it alone conformed all of the RL techniques found in this study. DKT and BKT were mostly commented on R1 and R2, with five and four occurrences respectively. This leads to the DL categorization (DKT + LSTM + DKVMN) to be extensively justified in the literature, while UL (PFA) is sparsely commented, and SVM not at all, despite its non-negligeable number of applications (seven). BN had the highest R3 count of all (three occurrences) and because of the absence of SVM comments, it carries all the justifications related to SL.

Third, we looked over the intended purpose of the ML implementation, besides the intended KT. In one hand, out of the 51 publications reviewed, seven (~15%) employ ML for initializing the LM (e.g., for another course, academic year, or for determining the ML parameters in a pipeline) by accounting previous system interactions, grades, pre-tests, or other data. In the other hand, most publications (44–85%) perform some form of prediction. Finally, only one proposal (~2%) incorporates both a prediction and/or recommendation mechanism as well. A pie chart of ML techniques' purpose distribution is presented in Fig. 8.

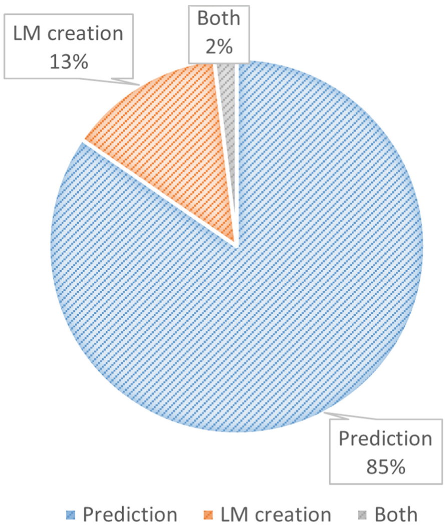


Fig. 8. Pie chart distribution of ML purpose [18].

Fourth, we surveyed the software used to perform the ML calculations. Note that many publications (~50%) do not mention their software of choice. Python (comprising Keras, TensorFlow, PyTorch and scikit-learn) is the largest group, with 13 papers. *Ad-hoc* solutions follow with five papers, and finally C, Java (-based), Matlab and R solutions, with 2 publications each. Outliers were SPSS and Stan, with one (1) paper each. A pie chart illustrating the distribution of programming languages is shown in Fig. 9.

Fifth, we highlighted (and checked for existence) the public datasets employed, shown in Table 1. All the datasets we found in the literature were online and accessible when reviewed. We made the version distinction (yearly or by topic) of datasets from the same source (such as DeepKnowledgeTracing and ASSISTments, respectively) because

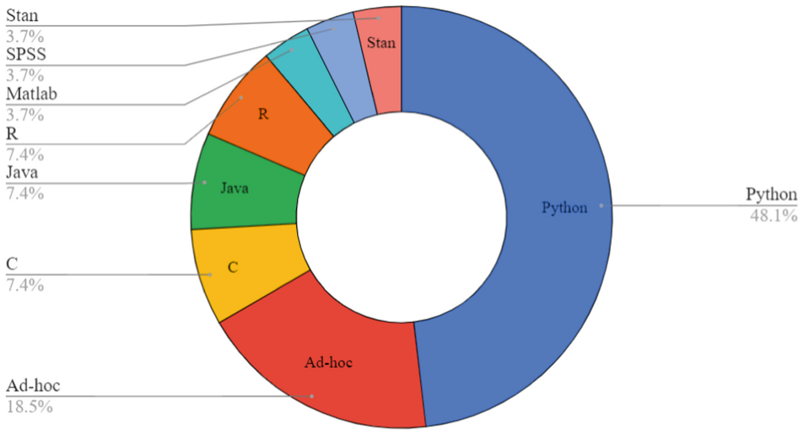


Fig. 9. Pie chart distribution of ML programming language [18].

they differ on either the number of features, or the dimensioning, or dataset creation method.

We make the distinction from our previous paper [18] in that the datasets "MOOC [Big Data and Education on the EdX platform]" and "Hour of Code" are not available anymore[6] online and thus, do not appear anymore in Table 1. Moreover, the "DataShop" dataset (https://pslcdatashop.web.cmu.edu/) was removed as well because it points to a repository of learning interaction data with no specific dataset.

Moreover, for each available singular dataset we catalogued the number of files it encompasses, the file format in which it is saved, its total size (in multiples of bytes), and lastly, the list of available features (the mixed uppercase and lowercase features' labels are 'as found' within the datasets):

- ASSISTments2009. 1 CSV file (61.4 MB) with the following features:
  order_id, assignment_id, user_id, assistment_id, problem_id, original, correct, attempt_count, ms_first_response, tutor_mode, answer_type, sequence_id, student_class_id, position, type, base_sequence_id, skill_id, skill_name, teacher_id, school_id, hint_count, hint_total, overlap_time, template_id, answer_id, answer_text, first_action, bottom_hint, opportunity, opportunity_original
- ASSISTments2013. 1 CSV file (2.8 GB) with the following features:
  problem_log_id, skill, problem_id, user_id, assignment_id, assistment_id, start_time, end_time, problem_type, original, correct, bottom_hint, hint_count, actions, attempt_count, ms_first_response, tutor_mode, sequence_id, student_class_id, position, type, base_sequence_id, skill_id, teacher_id, school_id, overlap_time, template_id, answer_id, answer_text, first_action, problemlogid, Average_confidence(FRUSTRATED), Average_confidence(CONFUSED), Average_confidence(CONCENTRATING), Average_confidence(BORED)
- ASSISTments2015. 1 CSV file (17.4 MB) with the following features:
  user_id, log_id, sequence, correct

[6] As of end of October 2021.

Table 1. Public datasets found.

Name	URL
ASSISTments2009	https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010
ASSISTments2013	https://sites.google.com/site/assistmentsdata/home/2012-13-school-data-with-affect
ASSISTments2015	https://sites.google.com/site/assistmentsdata/home/2015-assistments-skill-builder-data
KDD Cup	https://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp
DataShop: OLI Engineering Statics - 1.14 (Statics2011)	https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=507
The Stanford MOOCPosts Data Set	https://datastage.stanford.edu/StanfordMoocPosts/
DeepKnowledgeTracing dataset	https://github.com/chrispiech/DeepKnowledgeTracing
DeepKnowledgeTracing dataset - Synthetic-5	https://github.com/chrispiech/DeepKnowledgeTracing/tree/master/data/synthetic

- KDD Cup. 6 TXT files (8.4 GB) with the following features:
  Row, Anon Student Id, Problem Hierarchy, Problem Name, Problem View, Step Name, Step Start Time, First Transaction Time, Correct Transaction Time, Step End Time, Step Duration (sec), Correct Step Duration (sec), Error Step Duration (sec), Correct First Attempt, Incorrects, Hints, Corrects, KC(Default), Opportunity(Default)
- DataShop: OLI Engineering Statics – (Fall 2011). 1 TXT file (171 MB) with the following features:
  Row, Sample Name, Transaction Id, Anon Student Id, Session Id, Time, Time Zone, Duration (sec), Student Response Type, Student Response Subtype, Tutor Response Type, Tutor Response Subtype, Level (Sequence), Level (Unit), Level (Module), Level (Section1), Problem Name, Problem View, Problem Start Time, Step Name, Attempt At Step, Is Last Attempt, Outcome, Selection, Action, Input, Input, Feedback Text, Feedback Classification, Help Level, Total Num Hints, KC (Single-KC), KC Category (Single-KC), KC (Unique-step), KC Category (Unique-step), KC (F2011), KC Category (F2011), KC (F2011), KC Category (F2011), KC (F2011), KC Category (F2011), School, Class, CF (oli:activityGuid), CF (oli:highStakes), CF (oli:purpose), CF (oli:resourceType)
- The Stanford MOOCPosts Data Set. 11 CSV files (3.28 MB) without headers within the files.
- DeepKnowledgeTracing dataset. 2 CSV files (2.58 MB) without headers within the files.
- DeepKnowledgeTracing dataset - Synthetic-5. Over 40 CSV files (15.64 MB) without headers within the files.

Thus, the elements presented here-in, namely the ML techniques, their chosen rationale, their KT in LM purpose, the most usual programming language software employed, and the subsequent required datasets, found in the 51 reviewed publications constitute the answer to "RQ2: How do the most employed ML techniques fulfil the considered relevant aspects (identified in Sect. 2.1.2) to insure KT in LM?".

5 Discussion

In this section we present our observations on the ML techniques addressed in the precedent section, issued from the 51 reviewed publications. This discussion covers the five elements mentioned in Subsect. 2.1.2.

ML Technique: We begin by noting that, in the reviewed papers, rarely a clear, well-defined, single ML technique is employed: very often additions or variants are employed (which make the point of the paper). Research teams seem to focus their attention on fine-tuning parameters (to improve prediction) rather than on expanding the application of ML for KT to other educational data sources or contexts. Authors recognize that additional features (or dimensions) would encumber the learning phase for limited gains, compared to parameter fine-tuning. As such, many papers propose pipelines ('chains') of ML techniques to optimize the process without increasing the calculation load. Performance improvements aside, this brings up two inconveniences: the difficulty of identifying the ML technique suitable for KT, and the difficulty to evaluate and compare any two papers employing different pipelines, as the intermediary inputs and outputs of the chain elements are quite different between papers.

ML Purpose: We distinguish two families of stated purposes in the reviewed ML techniques for KT: prediction and LM creation. Prediction is often portrayed as a probability, which can be interpreted as a mastery (or degree) of a skill (0–100), a grade (0–10), or a likelihood (0–1) of getting the answer right (in binary answers). In LM creation, ML predicts parameters for initializing the LM. We noticed that clustering, personalization, and/or resource suggestion (or other ML techniques, such as NLP) were performed once the predicting phase had taken place.

ML Choice Rationales: We condense the rationales exposed by the authors when choosing a ML technique. We omit rationales based on novelty, status-quo, or vague generalities, e.g., "nobody had done it before", "the existing system already uses this mechanism", "because it helps predict students' performance", respectively. The choice of BKT's was mostly driven by popularity, although it had issues on learners' individuality, multi-dimensional skill support and modelling forgetting. BN also seemed to be a common, popular choice. Its main advantage was its ability to model uncertainty, although it seems to reach its limits if the number of students is kept "relatively low". On the contrary, DKT may benefit from large datasets and has proven being able to model multi-dimensional skills, although lacking in consistent predicted knowledge state across time-steps. DKVMN (based on LSTM) can model long-term memory and mastery of knowledge at the same time, as well as finding correlations between exercises and concepts, although it does not account for forgetting mechanisms. LSTM appears to additionally handle tasks other than KT satisfactory. It also models forgetting mechanisms over long-term dependencies within temporal sequences. It is then well suited for time series data with unknown time lag between long-range events. PFA does not consider answers' order (which is pedagogically relevant), nor models guessing, nor multiple-skills questions. Finally, RNNs are well suited for sequential data with temporal relationships, although long-range dependencies are difficult to learn by the model, hence the resurgence of LSTM.

Software for ML: Python (all frameworks and libraries merged) is the most common programming language employed for ML, more than doubling the number of papers employing *ad-hoc* languages. In this subject, we think that employing platform-specific programming languages for ML assures a lack of code portability (implying licensing issues, steep learning curve, little replicability, code isolation, and other situations) and thus, little to no adoption of research proposals adopting this approach. However, specialized ML software, designed by worldwide experts on the field, with a large user base maintaining it, backed up by large and specialized ML corporations, tends to be performance-optimized for diverse hardware and software and quasi bug-free. An *ad-hoc* solution developed in-house by a comparatively small team of developers cannot compete with such an opponent. We were taken aback by two facts: the sparse use of specialized mathematical software (Matlab, R, SPSS) in ML, and to learn that about 50% of all reviewed publication do not specify what software was employed for their ML calculations, leaving little room for independent replication, results verification, and additional development.

Datasets: We noticed that frameworks proposal papers aim to prove the performance of their approach using publicly available datasets. An overview of the found public datasets can be found in Table 1, in the previous section. The datasets found in the publications (chosen by papers' authors) are static (*i.e.*, they are not part of a "live" system), mostly contain grades or other similar evaluation measurements (but no behavioral or external sensor data) and provide the non negligeable advantages of being explained in detail (their data structure) and having their data already labelled, often by experts. This contrasts with the "organic" data employed in publications where ML is addressed for an existing, live system, even if it is for testing purposes. Both variants could benefit from each other's approaches, but this would require diverse, detailed, copious high-quality data that many institutions simply cannot afford to generate nor stock, let alone analyze.

One of most recurring datasets is the ASSISTment [49] (employed in 11 publications), of which there are different versions. A noteworthy fact is that this dataset has been acknowledged to have two main kind of data errors: (1) duplicate rows (which are removed if acknowledged by the authors) and (2) "misrepresented" skill sequences. Drawbacks of the latter issue (2) have been discussed: while this does not affect the final prediction, it nevertheless might conduce the learner to being presented with less questions on one of the merged skills (the less mastered) because the global (merged) mastery of skill is achieved mainly through the mastery of the most known skill [50, 51]. This raises the importance of the data cleaning process [25], which processing time is not negligeable and should be accounted at early data mining stages.

6 Conclusion and Perspectives

The aim of this research work is to present existing ML techniques for KT in LM employed during the last five years. It also helps to prepare our target public to the complex task of choosing a ML for KT technique by outlying the current trends in the research field. To reach this objective, we used the PRISMA guidelines for systematic reviews methodology, which led us to conclude that the following five ML for KT techniques were the most employed in the State-of-the-Art during the last five years: BKT (18 applications), DKT (13 applications), LSTM (12 applications), BN (11 applications), and SVM (7 applications). However, the reasons behind choosing any given ML technique were not really detailed in reviewed publications. We also noticed that combinations of ML techniques arranged in pipelines are a common practice, and that the most recent research (2019–2020) favored pipeline and/or parameter optimization over new techniques implementation. The use of public datasets is recurrent: they contain usually grades or other similar evaluating metrics, but no other pedagogical relevant data. On this subject, we insist that extensive data cleaning and other pre-treatments are highly recommended before using these public datasets. Finally, our results show that the ML programming language of choice is Python (all libraries & frameworks combined).

This review of literature is inscribed in the context of the "Optimal experience modelling" research project, conducted by the University of Lille. This research project [52] aims to model and trace the Flow psychological state, alongside KT, via behavioral data, using the generic Bayesian Student Model (gBSM), within an Open Learner Model.

The current challenge is to incorporate the ML relevant aspects highlighted in this study, and the behavioral and psychological aspects (log traces and Flow state determination) specifically linked to the project. Namely, a ML technique supporting the gBSM, capable to initialize the LM and perform KT, supported by the most common programming language for ML, based on a sound rationale. The originality of such research lies in the use of live, behavioral, Flow-labelled data issued from the French-spoken international MOOC "Project Management"[7].

Acknowledgements. This project was supported by the French government through the Programme Investissement d'Avenir (I-SITE ULNE/ANR-16-IDEX-0004 ULNE) managed by the Agence Nationale de la Recherche.

Appendix

The Appendix is composed of: (a) the ML categorization figure, (b) the summary table of ML for KT in LM (for clarity reasons, the extensive column 'rationale' has been removed), and (c) the full table of the 29 ML techniques.

It can be found at the following address:
https://nextcloud.univ-lille.fr/index.php/s/DpJwFtRHg399pXm.

[7] https://moocgdp.gestiondeprojet.pm/.

References

1. Corbett, A., Anderson, J., O'Brien, A.: Student modeling in the ACT programming tutor, Chap. 2. In: Nichols, P., Chipman, S., Brennan, R. (eds.) Cognitively Diagnostic Assessment. Lawrence Erlbaum Associates, Hillsdale (1995)
2. Giannandrea, L., Sansoni, M.: A literature review on intelligent tutoring systems and on student profiling. Learn. Teach. Med. Technol. **287**, 287–294 (2013)
3. Nakić, J., Granić, A., Glavinić, V.: Anatomy of student models in adaptive learning systems: a systematic literature review of individual differences from 2001 to 2013. J. Educ. Comput. Res. **51**(4), 459–489 (2015)
4. Wenger, E.: Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge. Morgan Kaufmann, Burlington (2014)
5. El Mawas, N., Gilliot, J.-M., Garlatti, S., Euler, R., Pascual, S.: As one size doesn't fit all, personalized massive open online courses are required. In: McLaren, B.M., Reilly, R., Zvacek, S., Uhomoibhi, J. (eds.) CSEDU 2018. CCIS, vol. 1022, pp. 470–488. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21151-6_22
6. Martins, A.C., Faria, L., De Carvalho, C.V., Carrapatoso, E.: User modeling in adaptive hypermedia educational systems. J. Educ. Technol. Soc. **11**(1), 194–207 (2008)
7. Swamy, V., et al.: Deep knowledge tracing for free-form student code progression. In: Penstein Rosé, C., et al. (eds.) AIED 2018. LNCS (LNAI), vol. 10948, pp. 348–352. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93846-2_65
8. Corbett, A., Anderson, J.: Knowledge tracing: modeling the acquisition of procedural knowledge. User Model. User-Adap. Inter. **4**(4), 253–278 (1994)
9. Yudelson, M.V., Koedinger, K.R., Gordon, G.J.: Individualized Bayesian knowledge tracing models. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS (LNAI), vol. 7926, pp. 171–180. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39112-5_18
10. Zhang, J., Li, B., Song, W., Lin, N., Yang, X., Peng, Z.: Learning ability community for personalized knowledge tracing. In: Wang, X., Zhang, R., Lee, Y.-K., Sun, L., Moon, Y.-S. (eds.) APWeb-WAIM 2020. LNCS, vol. 12318, pp. 176–192. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-60290-1_14
11. IBM: What is Machine Learning?. IBM Cloud Learn Hub, 18 déc. 2020. https://www.ibm.com/cloud/learn/machine-learning. (consulté le 23 déc. 2020)
12. Moher, D., Liberati, A., Tetzlaff, J., Altman, D.G.: The PRISMA Group: Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement. PLOS Med. **6**(7), e1000097 (2009). https://doi.org/10.1371/journal.pmed.1000097
13. Das, K., Behera, R.N.: A survey on machine learning: concept, algorithms and applications. Int. J. Innov. Res. Comput. Commun. Eng. **5**(2), 1301–1309 (2017). https://doi.org/10.15680/IJIRCCE.2017.0502001
14. Olsson, F.: A literature survey of active machine learning in the context of natural language processing. SICS Technical report, p. 59 (2009)
15. Shin, D., Shim, J.: A systematic review on data mining for mathematics and science education. Int. J. Sci. Math. Educ. **19**, 639–659 (2020). https://doi.org/10.1007/s10763-020-10085-7
16. Chakrabarti, S., et al.: Data mining curriculum: a proposal (version 1.0). In: ACM SIGKDD, 30 avr. 2006 (2006). Consulté le: 24 déc. 2020. [En ligne]. Disponible sur: https://www.kdd.org/curriculum/index.html
17. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015). https://doi.org/10.1016/j.neunet.2014.09.003

18. Ramírez Luelmo, S.I., El Mawas, N., Heutte, J.: Machine learning techniques for knowledge tracing: a systematic literature review. In: Proceedings of the 13th International Conference on Computer Supported Education, vol. 1, pp. 60–70 (2021). https://doi.org/10.5220/0010515500600070
19. Mohri, M., Rostamizadeh, A., Talwalkar, A.: Foundations of Machine Learning, 2nd edn. MIT Press, Cambridge (2018)
20. Brownlee, J.: A tour of machine learning algorithms. Machine Learning Mastery, 11 août 2019. https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/ (consulté le 28 déc. 2020)
21. van Engelen, J.E., Hoos, H.H.: A survey on semi-supervised learning. Mach. Learn. **109**(2), 373–440 (2020). https://doi.org/10.1007/s10994-019-05855-6
22. van Veen, F., Leijnen, S.: The Neural Network Zoo. The Asimov Institute (2019). https://www.asimovinstitute.org/neural-network-zoo/ (consulté le 23 déc. 2020)
23. Eagle, M., et al.: Estimating individual differences for student modeling in intelligent tutors from reading and pretest data. In: Micarelli, A., Stamper, J., Panourgia, K. (eds.) ITS 2016. LNCS, vol. 9684, pp. 133–143. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39583-8_13
24. Millán, E., Jiménez, G., Belmonte, M.-V., Pérez-de-la-Cruz, J.-L.: Learning Bayesian networks for student modeling. In: Conati, C., Heffernan, N., Mitrovic, A., Verdejo, M.F. (eds.) AIED 2015. LNCS (LNAI), vol. 9112, pp. 718–721. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19773-9_100
25. Chicco, D.: Ten quick tips for machine learning in computational biology. BioData Min. **10**(1), 35 (2017). https://doi.org/10.1186/s13040-017-0155-3
26. Wen, J., Li, S., Lin, Z., Hu, Y., Huang, C.: Systematic literature review of machine learning based software development effort estimation models. Inf. Softw. Technol. **54**(1), 41–59 (2012). https://doi.org/10.1016/j.infsof.2011.09.002
27. Winkler-Schwartz, A., et al.: Artificial intelligence in medical education: best practices using machine learning to assess surgical expertise in virtual reality simulation. J. Surg. Educ. **76**(6), 1681–1690 (2019). https://doi.org/10.1016/j.jsurg.2019.05.015
28. Domingos, P.: A few useful things to know about machine learning. Commun. ACM **55**(10), 78–87 (2012). https://doi.org/10.1145/2347736.2347755
29. Pelánek, R.: Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modeling techniques. User Model. User-Adap. Inter. **27**(3–5), 313–350 (2017). https://doi.org/10.1007/s11257-017-9193-2
30. van de Sande, B.: Properties of the Bayesian knowledge tracing model. J. Educ. Data Min. **5**(2), 1–10 (2013)
31. Piech, C., et al.: Deep knowledge tracing. In: Advances in Neural Information Processing Systems, vol. 28, pp. 505–513 (2015). [En ligne]. Disponible sur: https://proceedings.neurips.cc/paper/2015/file/bac9162b47c56fc8a4d2a519803d51b3-Paper.pdf
32. Minn, S., Yu, Y., Desmarais, M.C., Zhu, F., Vie, J.-J.: Deep knowledge tracing and dynamic student classification for knowledge tracing. In: 2018 IEEE International Conference on Data Mining (ICDM), pp. 1182–1187 (2018). https://doi.org/10.1109/ICDM.2018.00156
33. Mao, Y., Lin, C., Chi, M.: Deep learning vs. Bayesian knowledge tracing: student models for interventions. J. Educ. Data Min. **10**(2), 28–54 (2018)
34. Daniluk, M., Rocktäschel, T., Welbl, J., Riedel, S.: Frustratingly short attention spans in neural language modeling. arXiv preprint arXiv:1702.04521 (2017)
35. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, 2nd edn. Morgan Kaufmann, Los Angeles (1988)
36. Conati, C.: Bayesian student modeling. In: Nkambou, R., Bourdeau, J., Mizoguchi, R. (eds.) Advances in Intelligent Tutoring Systems, pp. 281–299. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14363-2_14

37. Vapnik, V.: Statistical Learning Theory. Wiley, Hoboken (1998)
38. Hämäläinen, W., Vinni, M.: Classifiers for educational data mining, Chap. 5. In: Handbook of Educational Data Mining, pp. 57–74. CRC Press, Boca Raton, USA (2010)
39. Lee, Y.-J.: Predicting students' problem solving performance using support vector machine. J. Data Sci. **14**(2), 231–244 (2021). https://doi.org/10.6339/JDS.201604_14(2).0003
40. Salzberg, S.L.: Book Review: C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc. (1993). Kluwer Academic Publishers (1994). [En ligne]. Disponible sur: http://server3.eca.ir/isi/forum/Programs%20for%20Machine%20Learning.pdf
41. Janan, F., Ghosh, S.K.: Prediction of Student's Performance Using Support Vector Machine Classifier (2021)
42. Zhang, J., Shi, X., King, I., Yeung, D.-Y.: Dynamic key-value memory networks for knowledge tracing. In: Proceedings of the 26th International Conference on World Wide Web, pp. 765–774 (2017)
43. Pavlik, P.I., Cen, H., Koedinger, K.R.: Performance factors analysis – a new alternative to knowledge tracing. Online Submission, p. 8 (2009)
44. Zhang, J., King, I.: Topological order discovery via deep knowledge tracing. In: Hirose, A., Ozawa, S., Doya, K., Ikeda, K., Lee, M., Liu, D. (eds.) ICONIP 2016. LNCS, vol. 9950, pp. 112–119. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46681-1_14
45. Sha, L., Hong, P.: Neural knowledge tracing. In: Frasson, C., Kostopoulos, G. (eds.) BFAL 2017. LNCS, vol. 10512, pp. 108–117. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67615-9_10
46. Trifa, A., Hedhili, A., Chaari, W.L.: Knowledge tracing with an intelligent agent, in an e-learning platform. Educ. Inf. Technol. **24**(1), 711–741 (2019). https://doi.org/10.1007/s10639-018-9792-5
47. Sciarrone, F., Temperini, M.: K-OpenAnswer: a simulation environment to analyze the dynamics of massive open online courses in smart cities. Soft. Comput. **24**(15), 11121–11134 (2020). https://doi.org/10.1007/s00500-020-04696-z
48. Crowston, K., et al.: Knowledge tracing to model learning in online citizen science projects. IEEE Trans. Learn. Technol. **13**(1), 123–134 (2020). https://doi.org/10.1109/TLT.2019.2936480
49. Razzaq, L., et al.: Blending assessment and instructional assisting. In: Proceedings of the 2005 Conference on Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology, NLD, pp. 555–562 (2005)
50. Pelánek, R.: Metrics for evaluation of student models (2015). https://doi.org/10.5281/zenodo.3554666
51. Schatten, C., Janning, R., Schmidt-Thieme, L.: Vygotsky based sequencing without domain information: a matrix factorization approach. In: Zvacek, S., Restivo, M.T., Uhomoibhi, J., Helfert, M. (eds.) CSEDU 2014. CCIS, vol. 510, pp. 35–51. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25768-6_3
52. Ramírez Luelmo, S.I., El Mawas, N., Heutte, J.: Towards open learner models including the flow state. In: Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization, Genoa Italy, pp. 305–310 (2020). https://doi.org/10.1145/3386392.3399295