

TrueLearn: A Python Library for Personalised Informational Recommendations with (Implicit) Feedback

Yuxiang Qiu^{1,*}, Karim Djemili^{1,*}, Denis Elezi^{1,*}, Aaneel Shalman^{1,*},
María Pérez-Ortiz² and Sahan Bulathwela²

¹*Department of Computer Science, University College London, UK*

²*Centre for Artificial Intelligence, University College London, UK*

Abstract

This work describes the *TrueLearn* Python library, which contains a family of online learning Bayesian models for building educational (or more generally, informational) recommendation systems. This family of models was designed following the "open learner" concept, using humanly-intuitive user representations. For the sake of interpretability and putting the user in control, the TrueLearn library also contains different representations to help end-users visualise the learner models, which may in the future facilitate user interaction with their own models. Together with the library, we include a previously publicly released implicit feedback educational dataset with evaluation metrics to measure the performance of the models. The extensive documentation and coding examples make the library highly accessible to both machine learning developers and educational data mining and learning analytic practitioners. The library and the support documentation with examples are available at <https://truelearn.readthedocs.io/en/latest>.

1. Introduction

It has been shown that personalised one-on-one learning could lead to improving learning gains by two standard deviations [1]. With this goal in sight, and the ambition to democratise education to a world population, we require responsible Artificial Intelligence systems that can bring scalable, personalised and governable models to a mass of learners [2]. Up until recently, the go-to solution for scaling education has been Intelligent Tutoring Systems (ITS), heavily relying on testing users for knowledge, which is a practical option for formal courses with a limited number of learning materials involved. However, educational recommender systems have now the opportunity to go one step further, leveraging implicit interaction signals (such as clicks and watch time) to personalise and support learning for informal lifelong learners [3]. This is exactly the focus of the models in this library, with the aim of making these methods more accessible, as publicly available learner models and datasets are currently scarce, and they can open up huge opportunities for education.


ORSUM@ACM RecSys 2023: 6th Workshop on Online Recommender Systems and User Modeling, jointly with the 17th ACM Conference on Recommender Systems, September 19th, 2023, Singapore

*Authors contributed equally to the paper

✉ m.bulathwela@ucl.ac.uk (S. Bulathwela)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

1.1. Our Contribution

This work introduces *TrueLearn*¹, an open-source Python library that packages state-of-the-art online recommendation models, datasets and visualisation tools. Among its diverse use cases, it can be used as a personalization component of e-learning platforms (e.g., YouTube and EdX) to estimate learners’ potential engagement with learning resources and to model their background knowledge, interests, and novelty. The library contains different components that will enable i) creating content representations of learning resources ii) managing user/learner states, iii) modelling the state evolution of learners using interactions and iv) evaluating engagement predictions. Requiring minimal data, its design offers a transparent solution that respects the privacy of its users and enables user interaction. The development of the TrueLearn library aims to provide both the research and developer communities with the opportunity to use the TrueLearn family of models. The paper describes the development process and experiments that demonstrate the utility of this package to the educational data mining community and beyond.

While the motivation for TrueLearn stems from education, the models are applicable to a wide variety of applications that relate to informational recommendations and to model engagement in tasks in which human learning is involved. Additionally, note that the models included are suitable both for implicit and explicit feedback. In our experiments, we used a dataset of video lecture watch patterns, which we use as a proxy for learner engagement, but the same models could be applicable if learners also provided e.g. explicit feedback on the difficulty of the learning material.

2. Related Work

We have researched related work on learner models and how to design usable machine learning libraries to make decisions regarding the design of the TrueLearn library. This section reviews these works and their influence on the development of the library.

2.1. Item Response Theory and Knowledge Tracing

Item Response Theory (IRT) focuses on designing, analysing and scoring ability tests by modelling learner’s knowledge and question difficulty, without considering changes in knowledge over time. The simplest of IRT, the Rasch model [4], computes the probability of scoring a correct answer as a function of the learner’s skill θ_l and the difficulty of the question/resource d_r :

$$P(\text{correct answer}|\theta_l, d_r) = f(\theta_l - d_r) \quad (1)$$

where f is usually a logistic function. TrueSkill model extends IRT to model the skill of multiple users playing a video game [5]. The TrueLearn models implemented in this work extend TrueSkill for learner engagement prediction.

An alternative to IRT for modelling learning is Knowledge Tracing (KT) [6]. Unlike IRT, KT model does not consider question difficulty but instead estimates knowledge acquisition as a function of practice opportunities. Several Bayesian KT (BKT) algorithms that extend the

¹Documentation available at <https://truelearn.readthedocs.io/en/latest>

original KT model have been proposed in the literature. The pyBKT library, a Python library of KT models provides a clear Application Programming Interface advocating the separation of data generation, model fitting and prediction functions [7]. However, conventional KT models do not train using online learning, introducing challenges when scaling to real-time scenarios with a large number of users learning over a long period of time.

2.2. TrueLearn Models

The TrueLearn family of online Bayesian learner models uses implicit feedback from learners to recover their learning state. Prior work has proposed several learner models that capture the learner's interests and the knowledge and novelty of the material. Subsequent work combines these individual models to propose proposed ensembles that can account for these factors simultaneously, improving the predictive performance of individual models [3].

While being data efficient and privacy-preserving by design, TrueLearn models generate humanly intuitive learner representations that are inspired by open learner models. An open learner model is a learner model that has been made accessible to the learner it represents or to other users (e.g. teachers, parents) [8]. This involves generating visualisations to construct an interface that will communicate to learners information about their knowledge and learning path. Open learner models come with definite advantages, such as promoting learner reflection by aiding learners in planning and monitoring their learning and allowing them to compare their knowledge to that of their peers [8]. Open learner models also come with associated challenges, since all learner representations and their visual presentations may not be equally understood by a wide variety of end-users. Among many different visualisations used to present learner knowledge state, user studies have shown that some visualisations are comparatively more user-friendly than others [9, 10]. This work builds on these findings to develop a set of visualisations that aid this communication process.

2.3. Design a Machine Learning Library

To design a user-friendly, easy-to-use and scalable library, we need to avoid commonly used bad design practices, such as rigidity, fragility, immobility and viscosity [11]. Rigidity refers to the tendency for software to be difficult to change. Fragility is the tendency for software to break once it has been updated. Immobility is the inability to reuse code within or across projects. Viscosity refers to the difficulty of retaining the original design when changes to the software are required. Various design principles and patterns are proposed and employed in software engineering to overcome these issues [12].

When designing a machine learning library, we also need to account for unique challenges (e.g. incorporating data, pre-processing, models etc.). A great example of a well-designed machine learning library that has been taken up by both industry and academia recently is `scikit-learn`. `scikit-learn` [13] proposes some general design principles (consistency, inspection and sensible defaults) and interface design (estimators and predictors) [14] for building a scalable and user-friendly machine learning library. Consistency emphasises the importance of establishing a shared and consistent interface across different machine learning models, as this reduces the learning cost of the library. Inspection is concerned with exposing

the model’s parameters and hyperparameters as public attributes, which makes it easier for users to access the internal states of the model. Sensible defaults ensure that the model behaves reasonably well with the default values. The estimator and predictor interfaces in scikit-learn reflect how the library implements these general guidelines. The estimator interface specifies a `fit` function to provide a consistent interface to the training model and exposes the `coef_` attribute to facilitate the inspection of the internal state of the model. The predictor interface specifies the `predict` and `predict_prob` functions as methods for utilising the trained model. Due to the time-tested design decisions that have succeeded in scikit-learn, the design decisions made in developing the TrueLearn library are inspired by these practices.

Because of the concept of duck typing in Python, others’ model implementations can interoperate with scikit-learn (e.g., developers can plug them into scikit-learn’s grid search) without being forced to inherit the above interfaces. This makes scikit-learn extensible and encourages users to reuse code. However, the use of duck typing makes it difficult to perform static program analysis [15], thus postponing the discovery of incorrect implementations until runtime and increasing the likelihood of software bugs [16]. Therefore, in our work, we tend to take a hybrid approach, utilising type annotations [17] throughout the code base while allowing the use of static duck types supported by the `Protocol` class [18]. Compared to traditional duck typing, static duck typing allows the library implementer to represent the requirement for parameters of a method explicitly but also does not force the user to inherit any class, making it easier for users to understand the intent of the method and helping static type checkers to analyse the code [18].

3. Library Overview

This section describes the problem setting, the architecture of the library and how it can be applied in practice. While TrueLearn provides a probability that can be mapped to a binary outcome (engaged/not engaged), the probability prediction on different materials can rank them in relation to the state of the learner, creating personalised recommendations.

3.1. Problem Setting

The scenario educational recommendation focuses on is modelling a learner ℓ in learner population L interacting with a series of educational resources $S_\ell \subset \{r_1, \dots, r_R\}$ where r_x are *fragments/parts* of different educational videos. The watch interactions happen over a period of T time steps, R being the total number of resources in the system. In this system with a total N unique knowledge components (KCs), resource r_x is characterised by a set of top KCs or topics $K_{r_x} \subset \{1, \dots, N\}$. We assume the presence i_{r_x} of KC in resource r_x and the degree d_{r_x} of KC coverage in the educational resource is observable.

The key idea is to model the probability of engagement $e_{\ell, r_x}^t \in \{1, -1\}$ between learner ℓ and resource r_x at time t as a function of the learner interest $\theta_{\ell_1}^t$, knowledge $\theta_{\ell_{\text{NK}}}^t$ based on the top KCs covered K_{r_x} using their presence i_{r_x} , and depth of topic coverage d_{r_x} .

3.2. Architecture

The TrueLearn library consists of several modules that contain programming logic to execute different tasks using the library. Figure 1 outlines the main structure of the TrueLearn library. These modules are described below.

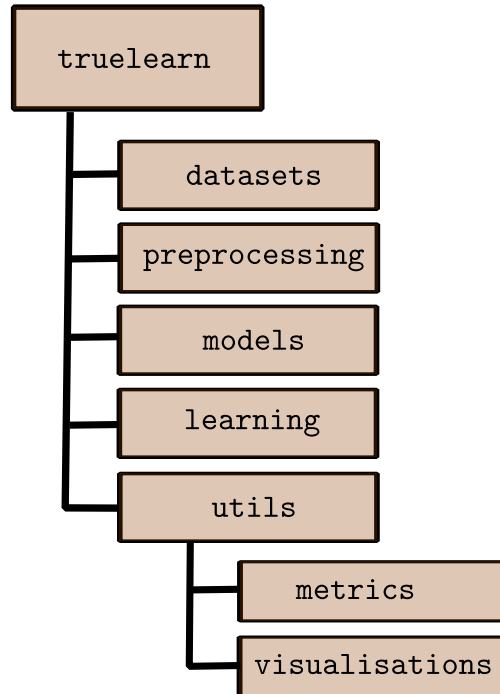


Figure 1: Library Structure of TrueLearn Library.

3.2.1. Datasets

The TrueLearn dataset module integrates tools for both downloading and parsing learner engagement datasets. Currently, PEEK [19], a publicly available learner engagement dataset is integrated. This module serves as a helper to integrate publicly available datasets that can be used for conducting experiments, evaluating model performance, and analysing learner data using TrueLearn’s visualisation capabilities.

3.2.2. Pre-processing

The pre-processing module contains utility classes designed specifically for extracting content representations from educational materials. The extracted representations serve as the foundation for creating KCs that can be used with IRT, KT and TrueLearn models. At present, utility functions that create Wikipedia-based KCs that are used in TrueLearn experiments [20] are included.

3.2.3. Models

The *models* module houses the class that can store the learner model. In this context, the learner model refers to the data structure that represents the learner's state (e.g. knowledge or interest). This learner model in the library is loosely coupled with the learning algorithms which makes this object reusable with many other learning algorithms that go beyond the TrueLearn algorithms currently included in the library. This means that the output of other learning algorithms such as BKT and IRT can still be used to create learner representations using this class.

3.2.4. Learning

The learning module contains the implementation of TrueLearn algorithms that can perform training and prediction of learner engagement with transcribed videos [21, 3]. Each classifier within this module follows an implicit interface inspired by the scikit-learn design. For training, `fit` function is used. For prediction, `predict` and `predict_proba` functions are used to generate a binary label and a probability value respectively. Currently, i) a set of baseline models, ii) TrueLearn algorithms that model interest, novelty and knowledge in isolation, and iii) an ensemble model that combines the isolated models are implemented.

3.2.5. Metrics

To evaluate the learning algorithm performance, the metrics module provides an interface to several key classification metrics including precision, accuracy, recall and F1 score. We use the scikit-learn API to support evaluation metrics. This opens up the opportunity to easily incorporate more evaluation metrics without having to put significant effort into testing and maintaining them in the future.

3.2.6. Visualisations

To effectively depict the learner state, *nine* different visualisations have been developed. These visualisations can be used with the ability to sort the output based on specific study topics (KCs), based on learners' proficiency. Figure 2 provides a preview of a subset of available visualisations. Out of these visualisations, seven are interactive visualisations that allow the end user to click and hover over the output to explore more details. However, they can also be saved as static images. The remaining two, namely, the Bubble Chart and Word Cloud, are exclusively static representations due to the limitations of the libraries used for their implementation. This module also provides the functionality to export these visualisations, where dynamic output can be saved in HTML format while the static output can be saved in various image formats such as PNG, JPEG and SVG.

3.3. Visualising the Learner State

Several visualisations that communicate the AI's learner state representation to a human user are implemented as part of the TrueLearn library. These visualisations are inspired by the

open learner model concept where models are developed to maintain a humanly intuitive representation [22]. Furthermore, the visualisations utilise user-friendly cues and conventions to minimise the learning curve for the human learner. The visualisations mainly represent the *current state* of the learner while there are also visualisations that can depict how the state of the learner evolves over time (where the x-axis of the plot is time). The i) bar plot, ii) bubble plot, iii) dot plot, iv) pie plot, radar plot, rose plot and tree plot are implemented. Figure 2 previews these visualisations. The TrueLearn family of algorithms represents a state using a mean and a variance value. In two-dimensional plots such as bar plots and dot plots, the mean is the y-axis and the confidence intervals mark the variance. In circle-based plots such as bubble plots and rose plots, the radius of the circle represents the mean. The intensity of the colour maps to the variance of the estimate (dark being low variance).

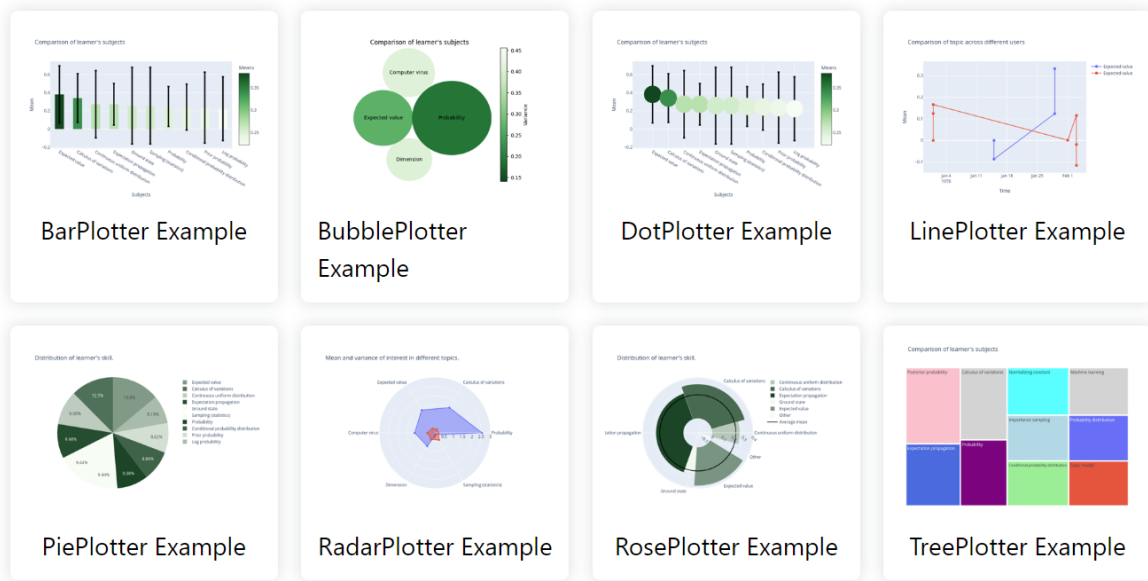


Figure 2: Multiple visualisations are available in TrueLearn to present the learner state in a humanly-intuitive way. Many different forms of visualisations are available to present the current learner state, while the line plotter also allows visualising the learner state's evolution over time.

4. Usage of TrueLearn Python Library

The TrueLearn Python library has two main uses, which are depicted in Figure 3.

4.1. Personalising E-learning/ Information

This Python package makes it very easy to incorporate personalisation into video-based e-learning platforms. The pre-processing module allows extracting KCs/topics from text transcriptions of video-based learning materials by associating them with Wikipedia topics. When

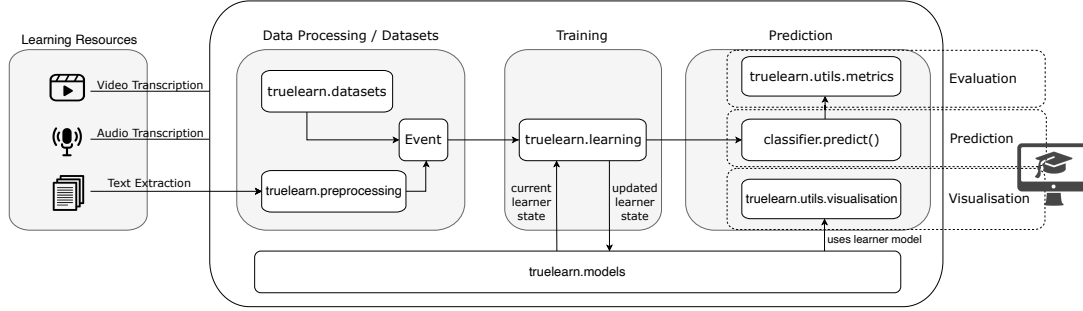


Figure 3: Potential usage scenarios of the TrueLearn Python package to build an educational recommender within an e-learning system.

a learner starts watching a video, the interaction signals can be recorded from the web application. The TrueLearn package can instantiate a learner model for each individual learner in the platform and use the interaction logs to update the learner model. The online learning algorithms can continuously fit the events to the learner model. The updated model can be used to predict engagement with a set of potential future videos (or any other type of educational material) to rank them and provide back to the learner as recommendations. Additionally, the learner may request to see their current state at any given point. The visualisation module can be used with the current learner state to create both static and interactive learner-state visualisations that can be presented to the end-user.

4.2. Offline/Online Evaluation of Informational Recommenders

Academics and researchers can use the TrueLearn library for conducting both online and offline evaluations of educational/informational recommendation algorithms. If the researchers need to benchmark a new learning algorithm, they can implement the learner algorithm using the common interface provided by the TrueLearn library. Then the Python library can be integrated with a web application to run online experiments and record user interactions. Similarly, offline evaluations can be done either by using i) an existing PEEK dataset or ii) integrating a new dataset.

5. Experiments

In order to validate the accuracy of the implementation, we ran a few small-scale experiments attempting to replicate the results published in prior work [3]. We used the PEEK dataset [19] to evaluate the performance of the primary TrueLearn models proposed in prior work, namely, i) TrueLearn Interest, TrueLearn Novelty, and TrueLearn INK. The experimental protocol was similar to the one used earlier. We also used a sequential experimental design. For each learner, its engagement at time t is predicted using its engagement at times 1 to $t - 1$. We used the hold-out validation technique in our experiments where the training data is used for hyperparameter tuning. The best hyperparameter combination based on the F1-Score is

identified. This combination is used with the test set to evaluate the final predictive performance. Since the engagement is predicted as a binary label in the PEEK dataset, the predictions for each event can be combined into a confusion matrix to compute accuracy, precision, recall, and F1 score. Same as in the prior publications, we calculate the weighted average of each learner’s metrics based on their number of events.

5.1. Empirical Evaluation

The empirical results obtained are reported in Table 1. The reported metric is the predictive performance each model obtained in the test set of the PEEK dataset.

Table 1

Predictive performance of TrueLearn Interest, TrueLearn Novelty, and TrueLearn INK algorithms are evaluated using Precision (Prec.), Recall (Rec.) and F1 Score (F1). The best and second best performance is indicated in **bold** and *italic* faces respectively.

Model	Acc.	Prec.	Rec.	F1
TrueLearn Interest	58.13	52.08	<i>78.61</i>	63.00
TrueLearn Novelty	<i>64.78</i>	<i>58.52</i>	80.91	65.53
TrueLearn INK	78.32	64.32	64.03	<i>64.00</i>

5.2. Visualisations

We designed visualisations, to empower students to recognise, contrast, and track the development of their skill level across subjects of their choice. It employs our algorithms’ prediction of their skill level (‘mean’) and calculation of a certain level in these predictions (‘variance’). These computations stem from learners’ interaction with diverse learning resources and subject matters, classified based on Wikipedia topics. The library is designed to enable learners to effortlessly generate dynamic and static visualisations and potentially export static ones to provide learners with visually enriched insights, thereby promoting a learner-centric, self-regulated study experience. Figure 4 previews the learner knowledge state generated from one of the learners in the PEEK dataset test data.

Our approach was guided by a thorough examination of seminal research on impactful learning visualisations. By incorporating the results from the questionnaire responses [10], we arrived at a selection of nine distinct visualisations: Bar Charts, Line Charts, Dot Plots, Pie Charts, Rose Charts, Bubble Charts, Tree Maps, Radar Charts, and Word Clouds.

6. Discussion

We discuss the TrueLearn library using several functional and non-functional aspects of the TrueLearn Python library.

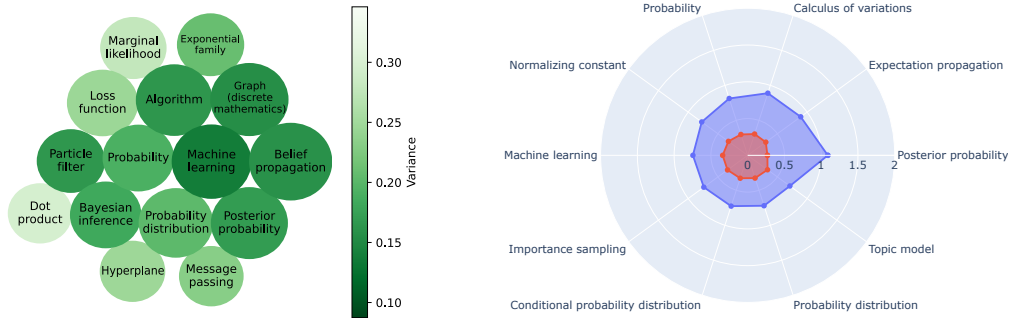


Figure 4: Multiple visualisations are available in TrueLearn to present the learner state in a humanly-intuitive way. The top 15 (most knowledge acquired) topics/KCs of a learner in a (left) bubble plot representation, where the size of the circle aligns with the mean value of the knowledge parameter and the intensity of the colour maps to the variance (lighter for higher variance) and (right) radar plot representation, where the mean skill is red and the variance is blue.

6.1. Library Design and Stability

Throughout the development process, we adhere to design principles discussed in related work, such as consistency, inspection, and hybrid typing. By providing a consistent training and prediction interface for all classifiers in the TrueLearn library, we have achieved the consistency principle. Simultaneously, for easy inspection, we exposed the internal state of objects by means of property attributes, public attributes, and getters/setters. The reason why we use getters/setters instead of public attributes in the classifier is to better facilitate the hybrid typing approach. With these methods, we can perform type and value checks when the hyperparameters of the classifier are modified, which ensures the robustness of the classifier implementation and guides the user to pass in the correct hyperparameters. Considering that users may want to implement their version of the KC (in order to include custom topic fields needed by their subject domain), the TrueLearn classifier interacts with the KC by using static duck typing whenever possible, which promotes interoperability between TrueLearn and the client code as well as the scalability of TrueLearn itself.

TrueLearn benefits from 100% test coverage achieved through a combination of integration, unit, and documentation tests. These tests were run on Python versions ranging from 3.7 to 3.11 on all major operating systems to ensure compatibility. By making use of periodic testing with Continuous Integration, we can more greatly ensure that TrueLearn will work as intended regardless of the operating environment.

6.2. Maintainability

Modularity is an important aspect of the TrueLearn library, achieved by using a collection of Base classes that define a common interface and shared functionality. This design establishes a foundation that can be easily extended and customised moving forward. To ensure the straightforward integration of the TrueLearn library, our documentation includes a wide range of examples showcasing the functionality offered by our API and visualisations. These examples

are accompanied by the necessary code to generate them, allowing for a clear understanding of their implementation. To further enhance maintainability, we have taken several additional measures. Firstly, we focused on minimising external dependencies wherever feasible to reduce the risk of compatibility issues and make it easier to maintain our codebase independently. Code consistency and readability are further enhanced by following the PEP 8 guidelines [23], which define a set of best practices for Python code.

Furthermore, we provide a comprehensive API reference, offering detailed information for each class and function. The project website² describes relevant information for both a potential end-user or a contributor to familiarise with the library. Each of these comes with its own fine-grained examples along with descriptions of their purpose. To better explain the rationale of the technical and design decisions made, information is provided concerning the library's design and style guide in the *contributing* section to minimise the learning curve associated with future development.

6.3. Relevance and Impact

Modelling learner state in a humanly intuitive manner, requiring minimal data and exclusively relying on individual user actions, TrueLearn offers a transparent learner model that respects the privacy of its users and can scale to lifelong education. The development of the TrueLearn library aims to provide both the research and developer communities with the opportunity to seamlessly use the TrueLearn family of models in their work. The learner models utilise Wikipedia entity-based entity linking to create KCs that are based on a publicly available knowledge base. The content annotation also can scale to thousands of materials created in different modalities (video, text, audio etc.).

The impact of TrueLearn is two-fold. For development and research, the TrueLearn library employs a design that conforms with popular machine learning libraries such as `scikit-learn` and `pyBKT` [14, 7]. The documentation is extensive and contains detailed examples that help the implementation. For end-use, the models employ probabilistic graphical models that are data efficient while providing humanly intuitive visualisations that trigger meta-cognition. The online learning algorithm updates the learner state in real-time helping better personalisation. A platform implementing TrueLearn can scale to a large population of users and support them through lifelong education.

The TrueLearn models advocate open learner modelling and employ open data sources such as Wikipedia helping the democratisation of education [2]. The data and computational efficiency of the models also lead to minimised carbon footprint. The models are currently used in building a platform that connects Open Educational Resources to lifelong learners supporting lifelong, equitable education. In an era where AI is reaching a position that can impact society significantly, TrueLearn library unlocks the power of personalisation of information while taking into account multiple human values that go beyond knowledge management (e.g. climate responsibility, privacy, transparency to name a few).

²Available at <https://truelearn.readthedocs.io/en/latest/>

6.4. Limitations

While getting inspired by scikit-learn library, the learning algorithms in the TrueLearn library are not compatible with some helper functions available in scikit-learn (such as grid search). Building seamless compatibility with these utilities will enable the TrueLearn library to be adopted by a wider audience while minimising the development effort required to support such powerful features. The exclusive support of online learning algorithms can also be seen as a limitation of the current library as there are many batch learning algorithms that are proposed for educational recommendation [24, 25]. The library also doesn't support state-of-the-art deep learning algorithms [26, 27]. These limitations are to be addressed in the future.

7. Conclusion

This work showcases *TrueLearn*, a Python library that models the learner knowledge and interest states to predict the engagement of learners with educational videos. The library contains several online learning models including ensemble models that model multiple factors that affect learner engagement together. It also includes a set of visualisations that can be used to interpret the learner's interest/knowledge state. The learner representations and state visualisations are comparable to outputs of knowledge tracing models, except, TrueLearn uses watch time interactions rather than relying on test taking. The empirical results demonstrate that the new implementation of the library achieves similar performance to the prior work that introduced these algorithms assuring correctness. The new implementation encourages educational data mining practitioners to use this library to incorporate educational video recommendations in e-learning systems. Researchers are encouraged to extend this library with new datasets and online learning algorithms for educational or informational recommendations.

7.1. Future Work

The immediate future work entails running several user studies to evaluate the effectiveness of the visualisations and identifying ways to improve them. Incorporating the library into a real-world e-learning platform to run online evaluations is also a top priority. We also aim to evaluate the performance of the TrueLearn models in the context of information recommenders that present informational content such as news and podcasts to demonstrate the generalisability of the models. Incorporating models that can exploit explicit feedback simultaneous to implicit feedback can also enhance the library and its utility. In the long term, we aim to add more general informational recommendation algorithms to the library and mobilise the research community to contribute various models, pre-processing techniques and evaluation metrics that the library can benefit from.

Acknowledgments

This work is partially supported by the European Commission-funded project "Humane AI: Toward AI Systems That Augment and Empower Humans by Understanding Us, our Society

and the World Around Us" (grant 820437) and the X5GON project funded from the EU's Horizon 2020 research programme grant No 761758.

References

- [1] B. S. Bloom, The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring, *Educational Researcher* 13 (1984) 4–16. URL: <http://www.jstor.org/stable/1175554>.
- [2] S. Bulathwela, M. Pérez-Ortiz, C. Halloway, J. Shawe-Taylor, Could AI Democratise Education? Socio-Technical Imaginaries of an EdTech Revolution, in: *In Proc. of the NeurIPS Workshop on Machine Learning for the Developing World (ML4D)*, 2021.
- [3] Bulathwela, Sahan and Pérez-Ortiz, María and Yilmaz, Emine and Shawe-Taylor, John, Power to the Learner: Towards Human-Intuitive and Integrative Recommendations with Open Educational Resources, *Sustainability* 14 (2022). URL: <https://www.mdpi.com/2071-1050/14/18/11682>. doi:10.3390/su141811682.
- [4] G. Rasch, Probabilistic Models for Some Intelligence and Attainment Tests, volume 1, 1960.
- [5] R. Herbrich, T. Minka, T. Graepel, Trueskill(tm): A bayesian skill rating system, in: *Advances in Neural Information Processing Systems 20*, MIT Press, 2007, pp. 569–576.
- [6] A. T. Corbett, J. R. Anderson, Knowledge tracing: Modeling the acquisition of procedural knowledge, *User modeling and user-adapted interaction* 4 (1994) 253–278.
- [7] O. Bulut, J. Shin, S. N. Yildirim-Erbasli, G. Gorgun, Z. A. Pardos, An introduction to bayesian knowledge tracing with pybkt, *Psych* 5 (2023) 770–786.
- [8] S. Bull, J. Kay, Open Learner Models, volume 308, 2010, pp. 301–322. doi:10.1007/978-3-642-14363-2_15.
- [9] S. Bull, P. Brusilovsky, J. Guerra, Which learning visualisations to offer students?, in: V. Pammer-Schindler, M. Pérez-Sanagustín, H. Drachsler, R. Elferink, M. Scheffel (Eds.), *Lifelong Technology-Enhanced Learning*, Springer International Publishing, Cham, 2018, pp. 524–530.
- [10] S. Bull, P. Brusilovsky, J. Guerra, R. Araújo, Individual and peer comparison open learner model visualisations to identify what to work on next, 2016.
- [11] R. C. Martin, Design principles and design patterns, *Object Mentor* 1 (2000) 597.
- [12] E. Gamma, R. Helm, R. Johnson, J. M. Vlissides, *Design patterns: elements of reusable object-oriented software*, Pearson Deutschland GmbH, 1995.
- [13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the *Journal of machine Learning research* 12 (2011) 2825–2830.
- [14] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, G. Varoquaux, API design for machine learning software: experiences from the scikit-learn project, *CoRR abs/1309.0238* (2013). URL: <http://arxiv.org/abs/1309.0238>. arXiv:1309.0238.
- [15] N. Milojkovic, M. Ghafari, O. Nierstrasz, It's duck (typing) season!, in: *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, 2017, pp. 312–315. doi:10.1109/ICPC.2017.10.

- [16] Z. Chen, Y. Li, B. Chen, W. Ma, L. Chen, B. Xu, An empirical study on dynamic typing related practices in python systems, in: Proceedings of the 28th International Conference on Program Comprehension, ICPC '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 83–93. URL: <https://doi.org/10.1145/3387904.3389253>. doi:10.1145/3387904.3389253.
- [17] G. van Rossum, J. Lehtosalo, Łukasz Langa, Type Hints, PEP 484, 2014. URL: <https://peps.python.org/pep-0484/>.
- [18] I. Levkivskyi, J. Lehtosalo, Łukasz Langa, Protocols: Structural subtyping (static duck typing), PEP 544, 2017. URL: <https://peps.python.org/pep-0544/>.
- [19] S. Bulathwela, M. Perez-Ortiz, E. Novak, E. Yilmaz, J. Shawe-Taylor, PEEK: A Large Dataset of Learner Engagement with Educational Videos, in: Proc. of RecSys Workshop on Online Recommender Systems and User Modeling (ORSUM'21), 2021. URL: <https://arxiv.org/abs/2109.03154>.
- [20] S. Bulathwela, M. Pérez-Ortiz, E. Yilmaz, J. Shawe-Taylor, TrueLearn: A Family of Bayesian Algorithms to Match Lifelong Learners to Open Educational Resources, in: AAAI Conf. on Artificial Intelligence, AAAI 20, 2020.
- [21] S. Bulathwela, M. Pérez-Ortiz, E. Yilmaz, J. Shawe-Taylor, Towards an Integrative Educational Recommender for Lifelong Learners, in: AAAI Conference on Artificial Intelligence, AAAI 20, 2020.
- [22] S. Bull, M. D. Johnson, M. Alotaibi, W. Byrne, G. Cierniak, Visualising multiple data sources in an independent open learner model, in: Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, July 9-13, 2013. Proceedings 16, Springer, 2013, pp. 199–208.
- [23] G. van Rossum, B. Warsaw, N. Coghlan, Style Guide for Python Code, PEP 8, 2001. URL: <https://www.python.org/dev/peps/pep-0008/>.
- [24] C. Lin, M. Chi, Intervention-bkt: Incorporating instructional interventions into bayesian knowledge tracing, in: A. Micarelli, J. Stamper, K. Panourgia (Eds.), Proc. of Int. Conf. on Intelligent Tutoring Systems, 2016.
- [25] Z. A. Pardos, S. M. Gowda, R. S. Baker, N. T. Heffernan, The sum is greater than the parts: Ensembling models of student knowledge in educational software, SIGKDD Explor. Newsl. 13 (2012) 37–44. doi:10.1145/2207243.2207249.
- [26] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, J. Sohl-Dickstein, Deep knowledge tracing, in: Advances in Neural Information Processing Systems, 2015.
- [27] Z. A. Pardos, W. Jiang, Designing for serendipity in a university course recommendation system, in: Proceedings of the Tenth International Conference on Learning Analytics & Knowledge, LAK '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 350–359. URL: <https://doi.org/10.1145/3375462.3375524>. doi:10.1145/3375462.3375524.