Online Programming Education Modeling and Knowledge Tracing

Yuting Sun[1], Liping Wang[1](✉), Qize Xie[1], Youbin Dong[1], and Xuemin Lin[1,2]

[1] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
{51184501146,51194501074,10165101256}@stu.ecnu.edu.cn, lipingwang@sei.ecnu.edu.cn

[2] The University of New South Wales, Sydney, Australia
lxue@cse.unsw.edu.au

Abstract. With the development of computer technology, more and more people begin to learn programming. And there are a lot of platforms for programmers to practice. It's often difficult for these platforms to customize the needs of users at different levels. In this paper, we address the above limitations and propose an intelligent tutoring model, to help programming platforms achieve better tutoring for different levels of users. We first devise a novel framework for programming education tutoring which is combined with programming education knowledge graph, crowdsourcing system and online knowledge tracing. Then, by ontology definition, information extraction and data fusion, we construct a knowledge graph to store the data in a more structured way. During the knowledge tracing stage, we extract behavior features and question knowledge features from a relational database and knowledge graph separately. Meanwhile, we improve the process for student ability evaluation and adapt the Knowledge Tracing algorithm to predict students' behavior on knowledge and questions. Experiment results on real-world user behavior data sets show that through the help of Knowledge Tracing algorithm, we can achieve considerably satisfied results on students' behavior prediction.

Keywords: Knowledge graph construction · Knowledge Tracing · Online programming

1 Introduction

Computer Programming is an important skill for software engineers and students majoring in computer science related fields. There have been many excellent platforms for programming practice exercise and examination, such as LeetCode, Nowcoder, UsacoGate Online Judge, Zhejiang University Online Judge, etc. Recently, those online programming platforms with countless programming questions are the most common ways for programming exercise. However, with the popularity of the online platforms, there are new requirements from different

© Springer Nature Switzerland AG 2020

users. For beginners, they need some help to diagnose bugs in their code. For Senior users, they usually need some guides for their study and avoid trying the same type of questions over and over again. And for teachers, they want to assign homework according to each student's knowledge level. In this paper, we address the above requirements and aim to provide an intelligent tutoring model for programming platforms, by which users with different programming levels will achieve effective one-to-one tutoring.

Intelligent Tutoring System (ITS) has been studied for several decades [10], which incorporate Artificial Intelligence (AI) techniques into education in order to achieve individualization of study process. For online education system, it is necessary to systematically investigate domain data mining and dynamic prediction model. In this paper, We devise a novel intelligent tutoring system for programming platform by intensive construction of the relative domain knowledge graph and introduction of effective knowledge tracing method. To the best of our knowledge, there is not a sophisticated approach for ITS in programming platform. The principle contributions of this paper are as follows:

- We devise a novel framework for programming education tutoring which is combined with programming education knowledge graph, crowdsourcing system and online knowledge tracing. Under the guidance of domain experts, the programming education knowledge graph is strictly constructed, which is utilized to serve for knowledge tracing algorithm and can be complemented by the crowdsourcing system.
- We extract behavior features and question knowledge features from a relation database and knowledge graph separately. Meanwhile, we improve the process for student ability evaluation and utilize the Dynamic Student Classification on Memory Networks [9] algorithm to make predictions on students' feedback on knowledge and questions.
- We conduct extensive experiments and evaluate knowledge tracing algorithms on real-world user behavior datasets to predict students' feedback on programming knowledge and questions separately. Comprehensive comparisons are made between algorithms. The experimental results demonstrate the effectiveness of our proposed method.

The rest of this paper is organized as follows: we briefly introduce the framework for programming education tutoring in Sect. 3. Domain ontology construction, information extraction and data fusion are represented in Sects. 3.1 and 3.2. Knowledge Tracing algorithms are devised in Sect. 4. We compare our method with other representative methods via some experiments, and depict a performance evaluation on student behavior prediction of our approach in Sect. 5. Finally, we conclude this paper in Sect. 6.

2 Related Work

Knowledge Graph Construction. There are two ways to construct Knowledge Graph [12], top-down and bottom-up. Top-down [18] construction refers to extracting ontology and pattern information from high-quality data and adding them to the knowledge graph with the help of structured data sources such as encyclopedia websites; the so-called bottom-up construction refers to extracting resource patterns from publicly collected data by certain technical means, selecting new patterns with high confidence and adding them to the knowledge base after manual review.

The bottom-up construction [15] usually takes three steps: information extraction, knowledge fusion, and knowledge processing. Information extraction [4] is to extract entities from various types of data sources. It's usually an automatic technology to extract structured information from structured or semi-structured data, involves entity extraction, relation extraction and attribute extraction. Some classical approaches for information extraction are based on (1) regularization, e.g. RoadRunner [5], (2) Template deduction [2], (3) Conditional random field [20], (4) Generalized hidden Markov model [19]. Some latest work for information extraction such as MGNER [17] for Multi-Grained Named Entity Recognition, DSGAN [13] using Generative Adversarial Networks for relation extraction. Knowledge fusion is to integrate the acquired new knowledge to eliminate contradictions and ambiguities. Knowledge processing is to obtain structured and organized knowledge, which mainly includes three aspects: ontology construction, knowledge reasoning and quality evaluation. Ontology construction can be divided into three main methods: manual ontology construction, automatic ontology construction and semi-automatic ontology construction. There are a lot of ontology construction methods been proposed, such as IDEF5, TOVE [1], MEHONTOLOGY, SKELETON, KACTUS, the Seven-Steps [8] method, etc. Among them, the Seven-Steps are proposed by Stanford University, which are to determine the scope of domain ontology, reuse existing ontology, determine domain terms, define hierarchical relationships, define attributes, define facets, and fill-in examples.

Knowledge Tracing. In the field of education, how to model students' mastery of knowledge is a crucial problem. Knowledge Tracing [3] aims to estimate students' level of mastery of a certain knowledge, based on his/her history interactions with questions. An accurate knowledge tracing enables us to grasp the needs of students and carry out accurate problem solving.

Traditional knowledge tracing are based on the first-order Markov Model, such as Bayesian Knowledge Tracing [6]. BKT models the knowledge state of a student as a set of binary variables, each variable represents if a student understands a knowledge. As a student keeps practice, the mastery of knowledge will also change dynamically. However, BKT can't handle the situation that one question might involve several knowledge.

Knowledge Tracing Machine [16] is a sequence prediction model. KTM uses the Factorization Machine [14] to solve the problem of sparse features, and capture the correlations between features. It can accurately and rapidly estimate students' performance and deal with questions with multiple knowledge. Also, KTM can handle questions with multiple skills.

With the popularity of deep learning [7] in recent years, it's also been applied to Knowledge Tracing. Deep Knowledge Tracing [11] uses the Long Short Term Memory to represent students' behavior. It can reflect the long-term knowledge dependencies. Unlike BKT assumed that once students master a certain knowledge, they will never forget it, DKT takes into account that students may forget what they have learned before after a period of time. It can capture students' recent performance to predict the results of their answers and make more use of students' recent performance.

3 Knowledge Graph for Programming Education

We investigate a programming education tutoring system which is constructed by a knowledge graph, crowdsourcing system and knowledge tracing algorithm with intelligence, real-time and dynamic. The framework of programming education tutoring system is shown in Fig. 1.

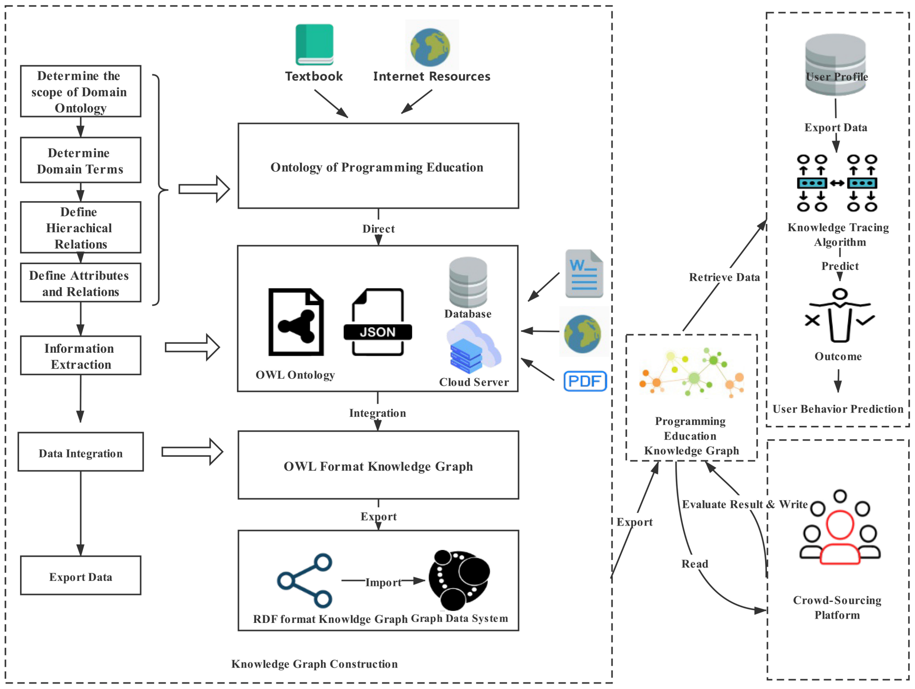

Fig. 1. Framework for knowledge graph construction and online knowledge tracing

For knowledge graph construction, during the preparation stage, textbooks, exercises, solutions for some difficult questions, related Internet resources, etc. are collected as the data sources. We carefully define the ontology of programming education using our domain knowledge and extract the information from different data sources. For unstructured data, regular expressions corresponding to textual content are devised. Then different formats of structural data are integrated and an RDF (Resource Description Framework) format knowledge graph is exported.

Moreover, in order to pursue a dynamical and updated knowledge graph, we devise a relevant crowdsourcing system which can publish tasks to related users and evaluate the answers. For the programming platform, in order to achieve more accurate results, both domain information in the knowledge graph and user behavior data in a relational database are served for programming tracing algorithms and online decision modules.

3.1 Domain Ontology Construction

The well-known knowledge graphs such as freebase, Wikidata, YaGo, etc. are widely utilized, but they haven't well covered the domain knowledge in programming education. To satisfy the domain requirement, we adapt the Seven-Steps method [8], and use four steps to define ontology: (1) determine the scope of domain ontology, (2) determine domain terms, (3) define hierarchical relationships, (4) define attributes and relationships.

Table 1. Modules and terms defined in programming education knowledge graph

Module	Terms
Data Structure (4 levels)	Linear Structure, List, Matrix, Tri Matrix, Queue, Stack, String, Tree Structure, Basic Tree Terms, Forest, Tree, Binary Tree, Multiway Tree, Tree Storage Structure, Graph, Basic Graph Term, Directed Graph, Undirected Graph, AOV, AOE, Graph Storage Structure, Graph Traversal, Minimal, Spanning Tree, Shortest Path, Searching, Basic Searching Term, Hash Searching, Linear Searching, Tree Searching, Sorting, Basic Sorting Term, External Sort, Internal Sort, Insert Sort, Merge Sort, Radix Sort, Select Sort, Swap Sort
Programming Language (4 levels)	Array, Branch Statement, Constant and Variable, Basic DataType, Constant, Variable Initialization, Variable Assignment, Variable Type, Custom Type, Dynamic Memory Management, Function, Customized Function, Library Function, IO Function, String Function, Recursive Function, Loop Statement, Operators and Expressions, Arithmetic Operator, Shift Operator, Expression Evaluation, Implicit Type Conversion, Positional Operator, Logical Operator, Relational Operator, Assignment Operator, Conditional Operator, Pointer, Preprocessing, Predefined Symbol, Conditional Compilation, File Contains, Macro
Dataset (2 levels)	Graph Structure Data, Linear Structure Data, Tree Structure Data, Others
Relevant Knowledge (3 levels)	Algorithm, Code Segment, Exercise, Extended Knowledge, Completion, Multiple Choice, Short Answer, True/False question

The ontology is constructed using the bottom-up approach. In the first step, for programming education domain scope determination, four major modules are identified and modules are guaranteed to be independent of each other as far as possible. Here, the defined four modules are data structure, programming language, datasets for experiment, and relevant knowledge(which can be reused as exercises separately). In the next two steps, terms belonging to each module are defined and arranged by hierarchical structure. The detailed definitions of terms in each module are listed in Table 1. Finally, based on the definition of modules and terms, the data property and object property are identified according to the domain knowledge.

3.2 Information Extraction and Data Fusion

After the domain ontology construction, we extract entities, attributes, and relations from different data sources which are combined with unstructured, semi-structured and structured data. We devise different extraction policies separately according to the structure of the data source. For the Data Structure and Programming modules defined in ontology, the data source mainly comes from Encyclopedia websites such as Baidu Encyclopedia and Wiki encyclopedia which are structured or semi-structured. In this case, the solution is straight-forward. An instance-class mapping table is constructed using the domain terms determined in Sect. 3.1 to accomplish the entity exaction. At the same time, a data property mapping table is created for attributes and relations extraction. These results are stored as JSON format temporarily.

For the relevant knowledge module defined in ontology, the source of the data is mainly from test questions set in the format of word and PDF. We devise the corresponding regular expressions to match the required data. In this process, both text and images embedded in the question are handled and stored into the relation database temporarily.

After information extraction, we achieve three independent data, that is domain ontology in OWL (Web Ontology Language) format (in Sect. 3.1), data that relates to Data Structure and Programming language in JSON format, and Relevant Knowledge data stored in relational databases. The JSON format data is converted to OWL format by instance-class mapping table (as shown in Algorithm 1) and the data stored in relation database is also converted to OWL format. Finally, the OWL file consists of instances of programming education Knowledge Graph, which can be exported to the graph database system.

Algorithm 1 is devised to determine the corresponding ontology of extracted information temporarily stored in JSON format. Given the clean data stored in JSON format, Algorithm 1 automatically completes the mapping from instance to ontology by data property-JSON mapping table and instance-class mapping table.

Algorithm 1. Ontology Instances generation

```
input: $JSONFile$: JSON Foramt File, $DDTable$: Data Property-JSON mapping
    table, $ECTable$ :Instance class mapping table
output: $OWLFile$: Ontology Instances
 1: function CONVERTJSON($JSONFile, DDTable, ECTable$)
 2:     construct an empty JSON object vector $v$
 3:     for each JSON object $o$ in JSON file do
 4:         construct an empty object $n$
 5:         for each attribute of $o$ do
 6:             if the key of attribute $a$ can be found in $DDTable$ then
 7:                 save $a$ to object $n$
 8:         save object $n$ to vector $v$
 9:     for each JSON object $n$ in vector $v$ do
10:         if attribute name can be found in $ECTable$ then
11:             Find the class name $c$ corresponding to name
12:             Construct corresponding $OWLdata$
13:             append OWL data to the end of OWL file
14:         else
15:             Find the class name $c$ corresponding to Chinese name
16:             Construct corresponding $OWLdata$
17:             Append OWL data to the end of the OWL file
18:     return OWLFile
```

4 Student's Behavior Prediction

Problem Formulation. We formulate students' behavior prediction problem as follows. Given m users $\{u_1, u_2, \ldots, u_m\}$, n questions $\{q_1, q_2, \ldots, q_n\}$, l knowledge $\{k_1, k_2, \ldots, k_l\}$, behavior prediction on question aims to get the correct rate of student i on question s. Behavior prediction on knowledge aims to evaluate the ability of student i on knowledge j. Our goal is to predict students' behavior on questions or knowledge based on students' records on online programming platform.

We adopt the DSCMN [9] algorithm to help make predictions. The basic idea of DSCMN is to use time intervals to divide the sequence of questions that students have done (the number of time intervals is how many times a student attempted in the segment), and then calculate student ability based on segment. After this, cluster segments of students with similar abilities, assign clusters as an extra feature in the input. The features we were using are (1) knowledgeId, (2) question difficulty, (3) cluster, one-hot encoding is applied in each feature, and concat these one-hot encodings as input. And then bring the Recurrent Neural Network to trace student knowledge in each segment.

We first calculate the average score of student i towards a knowledge j in time interval z in Eq. 1, the weight w_j represents the important degree of the question that knowledge j belongs to, which is calculated by the correct rate of the question. $R(x_j)_{1:z}$ is the difference between how much student i performs on knowledge j (see Eq. 2), $d^i_{1:z}$ is a learning ability vector of student i on each knowledge for time interval 1 to z.

$$\text{Average score}\left(x_j\right)_{1:z} = \sum_{t=1}^{z} \frac{w_j x_{jt}}{\left|N_{jt}\right|} \tag{1}$$

$$R\left(x_j\right)_{1:z} = \frac{\left(\text{Average score}\left(x_j\right)_{1:z} - 50\right)}{50} \tag{2}$$

$$d_{1:z}^{i} = \left(R\left(x_1\right)_{1:z}, R\left(x_2\right)_{1:z}, \ldots, R\left(x_n\right)_{1:z}\right) \tag{3}$$

After calculating the learning ability vector, we use the k-means algorithm to assign segments of students to clusters. Student learning ability will change over time, so different time segments of a student can belong to different clusters (see Eq. 4).

We calculate students' abilities based on learning ability vectors $d_{1:z}^{i}$, and cluster students with similar abilities. We first choose k random centroids, and update them by the distance with each time interval, we calculate by iteration to get final centroids. After we got centroids, we can assign each time interval segment of student i into the nearest cluster by Eq. 4. We selected part of student records for visualization on Fig. 2. There are 6 clusters in the figure, the color Cyan means students have no interactions in the corresponding time intervals. Students with the same color means similar ability.

$$\text{Cluster}\ \left(Stu_i, Seg_z\right) = \underset{C}{\arg\min} \sum_{c=1}^{K} \sum_{d_{1:z-1}^{i} \in C_c} \left\| d_{1:z-1}^{i} - \mu_c \right\|^2 \tag{4}$$

We conduct experiments on DSCMN on different time intervals to find the best length of time intervals to separate student sequences. Experiment results show that when time intervals are set to 50, we got the best performance. (See Fig. 3).

And then the cluster for students at different time intervals will add as an extra feature in the input. A RNN model has been used to make predictions in each segment (Eq. 5). We can predict a question that involves multiple knowledge, by averaging the predictions of knowledge involved as the prediction to a question. In Eq. 5, v_t is the success and failure levels of knowledge k_t until time t − 1. v_t is calculated by $\frac{\sum_{1}^{t-1} \text{score}(u,k_t)}{|k_t|}$, W_{hx} represents the input weight matrix, W_{hh} represents the recurrent weight matrix, W_{yh} represents the readout weight matrix, b_h, b_y are biases for latent and readout units. In Eq. 6, y_t represents the probability of answering correctly on a question with knowledge.

$$h_t = \tanh\left(W_{hx}\left[x_{t-1}, k_t, v_t\right] + W_{hh} h_{t-1} + b_h\right) \tag{5}$$

$$y_t = \sigma\left(W_{yh} h_t + b_y\right) \tag{6}$$

Lastly, the sigmoid function is used to make predictions. After we get the predictions of the correct rate of knowledge i on students t, we can calculate the predictions of question q on the student, by averaging the correct rate of knowledge that belongs to q.

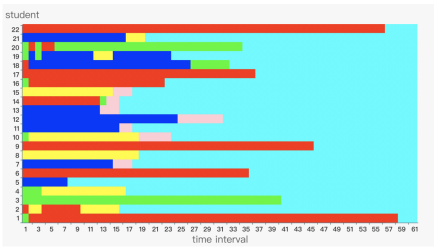

Fig. 2. Students' learning ability evolution over each time interval(50 attempts per time interval, each cluster is represented by different color)

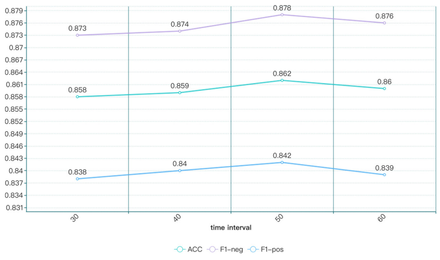

Fig. 3. Performance at different time intervals

5 Experiment

5.1 Experiment Setups

Dataset. The dataset we used contains 202 users, 184 questions and 48 knowledge. The data are extracted from Mynereus programming Platform[1]. There are a total 86772 records, including records of students over a year of practice history on the website. These questions are C language and Data Structure programming oriented. And we have in total 105 terms, 156 inter-class attributes, 35 intra-class attributes, 8 inter modules relations, 2615 instances, and 8135 inter-instance relations in our Knowledge graph. Student records are extracted from a relational database, and the relations of questions and knowledge are from our knowledge graph.

Metrics. We choose (1) Accuracy, (2) AUC, (3) F1-score as metrics in our experiments.

Table 2. Confusion matrix

Actual/Predict values	Positive	Negative
Positive	True Positive	False Positive
Negative	False Negative	True Negative

Accuracy. The accuracy is calculated as follows:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

1 http://code.mynereus.com.

AUC. AUC is the Area Under the ROC Curve. AUC considers the classifier's classification ability for both positive and negative cases, and can still make a reasonable evaluation for the classifier when the samples are unbalanced.

F1-Score. F1-score takes into account both the precision and recall of the classification model.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{8}$$

5.2 Results

We conduct two sets of experiments here. The first experiment is to predict students' behavior on questions, the second experiment is to predict students' behavior on knowledge. We select DKT-DSC and KTM as our comparison algorithms in the first experiment. Note that DSCMN is the derivation version of DKT-DSC.

Experiments for Predict Students' Behavior on Questions. The results for predicting students' behavior on questions are on Table 3. In DSCMN, we consider different questions that have different difficulty, and use the correct rate of each question as weight, and achieve the best result here. We also adapt KTM to make predictions on questions. KTM uses Factorization Machine to capture relevance between different features, the idea of Factorization Machine based on matrix decomposition, it captures relations between different features as in Eq. 9. Because the structure of KTM is very different from DSCMN and DKT-DSC, we select the best model in KTM to compare with the above two algorithms. In KTM, we use features (1) students, (2) questions, (3) knowledge, (4) attempts, (5) wins, (6) fails. Attempts are how many times a student answered a question, wins are how many times the student answered the question correctly.

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \tag{9}$$

Table 3. Experiment results on predict students' behavior on questions

Model	ACC	AUC	F1-neg	F1-pos
DKT-DSC: student, question	0.748	0.808	0.753	0.744
DSCMN: student, question, knowledge	0.808	0.880	0.816	0.799
DSCMN: student, question, knowledge(weighted)	**0.818**	**0.882**	**0.825**	**0.809**
KTM: student, question, attempts, wins, fails	0.785	0.842	0.787	0.782

Experiments for Predict Students' Behavior on Knowledge. The experiment results of predict students' behavior on knowledge are presented on Table 4. Instead of predict success or failure on questions, DSCMN focus on students' performance on knowledge. In practice sometimes we care more about student mastery on knowledge rather than a single question.

It is noteworthy that we improved the performance of DSCMN a lot by adding weight to different questions. DSCMN outperfoms the other two algorithms in every model, achieving an accuracy of 0.818, 0.882 on AUC on predicted questions, 0.865 on accuracy and 0.942 on AUC on predicted knowledge.

Table 4. Experiment results on predict students' behavior on knowledge

Model	ACC	AUC	F1-neg	F1-pos
DKT-DSC: student, knowledge	0.853	0.934	0.869	0.834
DSCMN: student, question, knowledge	0.862	0.938	0.878	0.842
DSCMN: student, question, knowledge(weighted)	**0.865**	**0.942**	**0.879**	**0.849**

6 Conclusion and Future Work

In this paper, we devise a novel framework for programming education tutoring which is combined with programming education knowledge graph, crowdsourcing system and online knowledge tracing. The constructed knowledge graph can represent the programming data and better capture relations between students, questions and knowledge, etc. We successfully extract domain features and improve the Knowledge Tracing algorithm to help us predict students' behavior for knowledge and questions. And, we conduct extensive experiments and evaluate knowledge tracing algorithms on real-world user behavior data sets. Comprehensive comparisons are made between algorithms. The experimental results demonstrate the effectiveness of our proposed method. In the future, based on the predictions we got before, we can recommend students' questions and enhance the online algorithm dynamically.

Acknowledgment. This work was partially supported by NSFC 61401155 and NSFC 61502169. The first author thanks University Côte d'Azur, France and Inria Sophia Antipolis Méditerranée, France where she conducted her master's final year project and internship.

References

1. Research on document clustering based on BP neural net. Computer Science (2002)
2. Arasu, A., Garcia-Molina, H.: Extracting structured data from web pages. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 337–348 (2003)

3. Corbett, A.T., Anderson, J.R.: Knowledge tracing: modeling the acquisition of procedural knowledge. User Model. User Adap. Inter. **4**(4), 253–278 (1994)
4. Cowie, J., Lehnert, W.: Information extraction. Commun. ACM **39**(1), 80–91 (1996)
5. Crescenzi, V., Mecca, G., Merialdo, P., et al.: Roadrunner: towards automatic data extraction from large web sites. VLDB. **1**, 109–118 (2001)
6. Freudenthaler, C., Schmidt-Thieme, L., Rendle, S.: Bayesian factorization machines (2011)
7. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
8. Li, W., Han, J., Pei, J.: CMAR: accurate and efficient classification based on multiple class-association rules. In: Proceedings 2001 IEEE International Conference on Data Mining, pp. 369–376. IEEE (2001)
9. Minn, S., Desmarais, M.C., Zhu, F., Xiao, J., Wang, J.: Dynamic student classification on memory networks for knowledge tracing. In: Yang, Q., Zhou, Z.-H., Gong, Z., Zhang, M.-L., Huang, S.-J. (eds.) Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 163–174. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-030-16145-3_13
10. Nwana, H.: Intelligent tutoring systems: an overview. Artif. Intell. Rev. (1990)
11. Piech, C., et al.: Deep knowledge tracing. In: Advances in Neural Information Processing Systems, pp. 505–513 (2015)
12. Qiao, L., Yang, L., Hong, D., Yao, L., Zhiguang, Q.: Knowledge graph construction techniques. J. Comput. Res. Dev. **53**(3), 582–600 (2016)
13. Qin, P., Xu, W., Wang, W.Y.: Dsgan: Generative adversarial training for distant supervision relation extraction. arXiv preprint arXiv:1805.09929 (2018)
14. Rendle, S.: Factorization machines with libFM. ACM Trans. Intell. Syst. Technol. (TIST) **3**(3), 1–22 (2012)
15. Van Der Vet, P.E., Mars, N.J.: Bottom-up construction of ontologies. IEEE Trans. Knowl. Data Eng. **10**(4), 513–526 (1998)
16. Vie, J.J., Kashima, H.: Knowledge tracing machines: factorization machines for knowledge tracing. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 750–757 (2019)
17. Xia, C., et al.: Multi-grained named entity recognition. arXiv preprint arXiv:1906.08449 (2019)
18. Zhao, Z., Han, S.K., So, I.M.: Architecture of knowledge graph construction techniques. Int. J. Pure Appl. Math. **118**(19), 1869–1883 (2018)
19. Zhong, P., Chen, J.: A generalized hidden Markov model approach for web information extraction. In: 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings) (WI 2006), pp. 709–718. IEEE (2006)
20. Zhu, J., Nie, Z., Wen, J.R., Zhang, B., Ma, W.Y.: 2d conditional random fields for web information extraction. In: Proceedings of the 22nd International Conference on Machine Learning, pp. 1044–1051 (2005)