POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

The Analysis of Student Traces for Q-matrix refinement and Knowledge Tracing

SEIN MINN SEIN MINN
Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie informatique

Juin 2020

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

The Analysis of Student Traces for Q-matrix refinement and Knowledge Tracing

présentée par **Sein Minn SEIN MINN**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Amal ZOUAQ, présidente
Michel DESMARAIS, membre et directeur de recherche
Roger NKAMBOU, membre
Cristóbal ROMERO MORALES, membre externe

DEDICATION

To my beloved family and friends...

ACKNOWLEDGEMENTS

First and foremost, I am most grateful to my advisor, Professor Michel C. Desmarais for his expertise, kindness, and most of all, for his patience and support throughout of my research and Ph.D. studying period. He always refined my research work with great patience, and helped me to go deeper and broader insight of complex research questions.

I would like to thank Professor Yu Yi and Professor Feida Zhu for their guidance and giving me valuable feedback during my visiting time at National Institute of Informatics, Japan and Singapore Management University, Singapore.

I would like to extend my gratitude to Dr. Jill-Jênn Vie and our lab members for their insightful discussions and helpful suggestions.

My thanks and appreciation goes to the thesis committee members, Professor Amal Zouaq, Professor Roger Nkambou and Professor Cristobal Romero.

I want to thank everyone who made this dissertation possible with their help in my Ph.D. study.

I do owe thanks to my family and parents for their patience, confidence and best support on me in the past years.

RÉSUMÉ

Nous assistons à une effervescense de l'auto-apprentissage rendue possible par l'Internet et les environnements d'apprentissage. L'accessibilité des MOOCS et des environnements d'apprentissage informatisés en est une manifestation. En contrepartie, l'apprenant perd le guidage personnalisé d'un tuteur humain et le développement d'environnements d'apprentissage adaptatifs vise à combler cette lacune. Afin d'offrir guidage et personnalisation au long du processus d'apprentissage, il est essentiel de bien évaluer les connaissances acquises de l'apprenant et d'adapter le matériel didactique en conséquence. Les recherches dans les domaines des tutoriels intelligents et de l'analytique des données éducationnelles visent essentiellement à développer des modèles de connaissances pouvant offrir le support à la personnalisation de l'apprentissage. Je propose dans cette thèse de nouvelles approches à la modélisation des connaissances apprenants autour de deux axes.

Le premier porte sur l'objectif de valider les connaissances et compétences sous-jacentes à des tâches à partir de données. La classification de questions et exercices en une taxonomie d'objectifs d'apprentissage est un exemple pratique d'identification de compétences sous-jacentes que les enseignants et pédagogues font couramment. Les chercheurs du domaine de la modélisation cognitive (Cognitive Diagnostic Modeling) vont plus loin en identifiant plusieurs connaissances et compétences derrière un seul problème à résoudre par exemple. Cet exercice est intrinsèquemet difficile et sujet aux erreurs. Les recherches pour faciliter la validation des connaissances sous-jacentes sont connues sous le nom du raffinement d'une Q-matrice qui représente l'alignement des tâches aux connaissances requises. La dernière décennie a été témoin de développements importants autour des approches basées sur les données pour effectuer le raffinement de Q-matrices. Ce processus de raffinement peut être considéré comme un problème de classification : pour chaque alignement tâche-connaissance défini par l'expert, l'algorithme de classification doit décider s'il est correct ou incorrect. Alors que la majorité des algorithmes portent sur une décision par alignement individuel, nous proposons une approche de classification basée sur des algorithme multi-classe où l'ensemble des connaissances requises par une tâche est soumises, plutôt que chaque connaissance individuellement. Les résultats de l'approche démontrent que le raffinement est généralement de meilleur qualité que les techniques de l'état de l'art.

Le second axe vise à améliorer les modèles d'apprentissage profond pour l'évaluation des connaissances de l'apprenant à partir de traces séquentielles du succès ou échecs aux tâches. Nous tablons sur un modèle d'évaluation de connaissances capable de capturer l'évolution temporelle du profil de connaissance qui évolue au long du processus d'apprentissage de l'apprenant. Les algorithmes d'apprentissage profond utilisant une architecture LSTM (*Long Short-Term Memory*) aspirent à cet objectif de mémoriser les informations temporelles et réussissent effectivement à mieux prédire les performances des apprenants. Mais le profil de connaissance constitue un mécanisme plus explicite de l'état de connaissance atteint et plus efficace pour synthétiser cet état. Nous intégrons donc ce mécanisme à une architecture LSTM et à une architecture de réseau de mémoires (*memory networks*) afin de valider cette hypothèse. Le profil de connaissance est modélisé sous forme de classes et cette information est encodée par un vecteur binaire de longueur unitaire (*one-hot*) qui est fourni en entrée aux modèles d'apprentissage profond.

D'autre part, les modèles d'apprentissage profond actuels ne tiennent pas compte de la difficulté des tâches, ou items. Or, cette information est très importante afin d'évaluer les connaissances acquises comme les modèles IRT (*Item Response Theory* ou *Théorie des réponse aux items* en français) et PFA (*Performance Factor Analysis*) l'ont démontré. Nous proposons donc des mécanismes aux architectures profondes afin d'intégrer cette information, notamment l'approche DSCMN. À l'instar du profil de connaissance, la difficulté des items est modélisée sous forme de niveaux et cette information est encodée par un vecteur binaire de longueur unitaire (one-hot).

Les résultats démontrent que l'intégration d'un profil de connaissance et du paramètre de difficulté des tâches améliorent tous deux, de manière significative, la performance prédictive des modèles d'apprentissage profond basés à la fois sur l'architecture LSTM et sur les réseaux de mémoires. Une analyse de l'apport individuel de chaque facteur à la performance prédictive démontre que leur contribution à la performance est significative pour chacun.

ABSTRACT

The growth of self-learning, enabled by the availability on the Internet of different forms of didactic material such as MOOCs and tutoring systems, increases in turn the relevance of personalized instructions for students in adaptive learning environment. For providing adaptive and personalized learning instructions, the assessment of student's mastery of a topic and the estimation of when she actually knows how to answer problems correctly is recognized as paramount in the fields of learning analytics and educational data mining community. In this dissertation, I propose novel approaches for building skills and student learning models along two axes. The first axis is to recover and ensure the quality of skills sets behind problems in learning system. The second axis is on improving the predictive accuracy of students' performance based on student ability profile on skills and considering of difficulty of the problem dynamically. Both of these axes are complementary and essential in knowledge assessment of future educational learning systems for equipping intelligent agents to provide adaptive instructions and independent learning environment for students.

The first axis is referred to as the Q-matrix refinement problem and consists in validating an expert-defined mapping of exercises and tasks to underlying skills. The last decade has witnessed a wealth of data driven approaches aiming to refine expert-defined mappings. This refinement can be seen as a classification problem: for each possible mapping of task to skill, the classifier has to decide whether the expert's advice is correct, or incorrect. Whereas most algorithms are working at the level of individual mappings, we introduce an approach based on a multi-label classification algorithm that is trained on the mapping of a task to all skills simultaneously. This approach improves Q-matrix validation methods by using supervised multi-label classifier. Results show it outperforms the existing Q-matrix refinement techniques.

The second axis aims to improve deep learning models of skills assessment based on sequential data. The student skills model needs to capture the temporal nature of student knowledge, changing over time, based on the learning transferred from previous practice. Deep learning has achieved a large amount of success in student performance prediction with models relying on Long short-term memory (LSTM). We proposed two approaches called Deep Knowledge Tracing and Dynamic Student Classification (DKT-DSC) and Dynamic Student Classification on Memory Networks (DSCMN) based on LSTM and key-value memory networks. We apply k-means clustering to capture students' temporal ability profile at each time interval, which serves as a transfer learning mechanism across student's long-term learning process. DKT-

DSC can capture temporal ability profile, utilize ability profile in assessment of knowledge mastery state simultaneously. The second approach, DSCMN, utilizes problem difficulty in prediction of student performance. According to experimental results, these approaches show improvements in student performance prediction over other state-of-the-art methods (such as BKT, PFA, etc.).

The studies in this dissertation are based on data from two distinct tutoring scenarios ASSISTment and Cognitive Tutor in which students learn high school mathematical problems (Algebra) with a computer-based learning system in the educational settings for knowledge tracing problem, and fraction algebra from Tatsuoka's work for Q-matrix refinement problem. My goal in these studies is to improve the assessment of student knowledge and student's performance prediction efficiently and effectively to facilitate the requirements of adaptive learning systems providing quality education to students around the world.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

MF	Matrix Factorization
ALS	Alternative Least Square
BKT	Bayesian Knowledge Tracing
BMF	Boolean Matrix Factorization
CDM	Cognitive Diagnosis Models
DCM	Diagnostic Classification Models
DINA	Deterministic-Input Noisy-And
DINO	Deterministic-Input Noisy-Or
BR	Binary Relevance Method
RAKEL	RAndom k-labELsets
LP	Label Power-set
RNN	Recurrent Neural Networks
DKT	Deep Knowledge Tracing
DKT-DSC	Deep Knowledge Tracing and Dynamic Student Classification
DKVMN	Dynamic Kev Value Memory Network
DSCMN	Dynamic Student Classification on Memory Networks
EDM	Educational Data Mining
G-DINA	generalized DINA models
IRT	Item Response Theory
ITS	Intelligent Tutoring System
KC	Knowledge Component
KLI	Knowledge Learning Instruction
KT	Knowledge Tracing
LCA	Latent Class Analysis
LFA	Learning Factor Analysis
LSTM	Long Short term Memory
LVM	Latent Variable Models
MCMC	Markov chain Monte Carlo
MF	Matrix Factorization
MOOC	Massive Open Online Course
NMF	Non-negative Matrix Factorization
PFA	Performance Factor Analysis

CHAPTER 1 INTRODUCTION

1.1 Adaptive Learning Environment

Nowadays, personalization is everywhere, from the recommendation of music, news, books, movies, etc. Learning can also be personalized with adaptive learning instructions to improve learner experience and enhance individual learning gains. It has long been recognized that individualized adaptive learning is much more effective than learning in classroom [2].

Intelligent tutoring systems address the challenge of large-scale personalization for real world human learning process. The most successful systems are currently used by tens or hundreds of thousands of students a year with growing numbers [3].

A successful adaptive learning environment requires two types of adaptations:

- The ability to provide highly specific, immediate, and effective supports during problem solving.
- The ability to structure adaptive learning content according to the individual skill proficiency of each student.

In the quest to better support the development of adaptive learning environments, Learning Analytics has captured the attention of data miners for more than a decade. Researchers from education and computer science have been trying to optimize complex and time-consuming design of adaptive learning environments by using advanced data mining and machine learning techniques. They aim to deliver means for Intelligent Tutoring Systems (ITS) for assisting the students more effectively. Guiding problem solving efficiently and effectively is a recurring topic in educational research.

Our work falls within the Learning Analytics field and aims to provide tools and techniques for student modeling that will help build adaptive learning environments, or Intelligent Tutoring Systems. We briefly review the field of Intelligent tutoring systems and its major challenges.

1.2 Knowledge Tracing (KT) and Knowledge Components (KC)

One of the main challenges for adaptive learning system designers is tracing the student's knowledge state (level of proficiency in particular knowledge or skill) in order to provide more supportive pedagogical instruction adaptively in student's learning process. This is referred to as Knowledge Tracing (KT). Accurate Knowledge tracing (KT) is a major focus in the educational research community, as it is pivotal in providing appropriate and adaptive personalized guidance during the student's learning process. KT aims to assess mastery of skills over time and use this information for personalized learning materials of students with various ability profile, whether in MOOCs, in a tutoring system, or in a web environment.

1.2.1 Knowledge Components Representation and Granularity

Knowledge Tracing naturally implies a representation of skills. We often refer to skills as Knowedge Components, KCs.

The student's knowledge can be characterized as a quantitative measure over all KCs: a probability that a student will master the set of KCs in a domain. Alternatively, it can be a qualitative value (good, average, and poor), or binary value (known and unknown).

A common means of representing a knowledge state is to define it as a subset of all possible skills. In a discrete framework, the knowledge state will be a subset, or as a qualitative label over each skill (mastered/unmastered, good/average/poor). In a probabilistic and continuous framework, a knowledge state is represented as a probability distribution over the full set of skills [4]. But in all cases, the knowledge state is represented in reference to the full set of possible skills.

Next comes the question of how fine-grain are the Knowledge Components (KCs). For example, when a problem such as

$$1 + 2 \times 3.5 = ?$$

is given to a student, we can consider she has to master the KCs or skills:

- *integer addition*
- *integer multiplication*
- *decimal notation*
- *decimal multiplication*
- *decimal addition*

for solving that problem. This would be considered a fine-grain representation. In a coarser definition, we could consider the a single skill involved in this task:

- *arithmetics*

and state that to successfully solve this problem, the students needs to have a “good” level of mastery of the arithmetics.

1.2.2 Knowledge Components, Skills, Items

For our purpose, KCs, or skills, can represent factual knowledge, problem-solving abilities, recognition of patterns and situations, etc. In general, they are the factors associated with an individual that determine the outcome of that individual over a task or item. One, or many, KCs can be associated with a single task.

KCs are latent factors that determine the outcome of a student being tested on an item. KCs are latent to the extent that they are never observed. Only the outcome of individual performing tasks is observed.

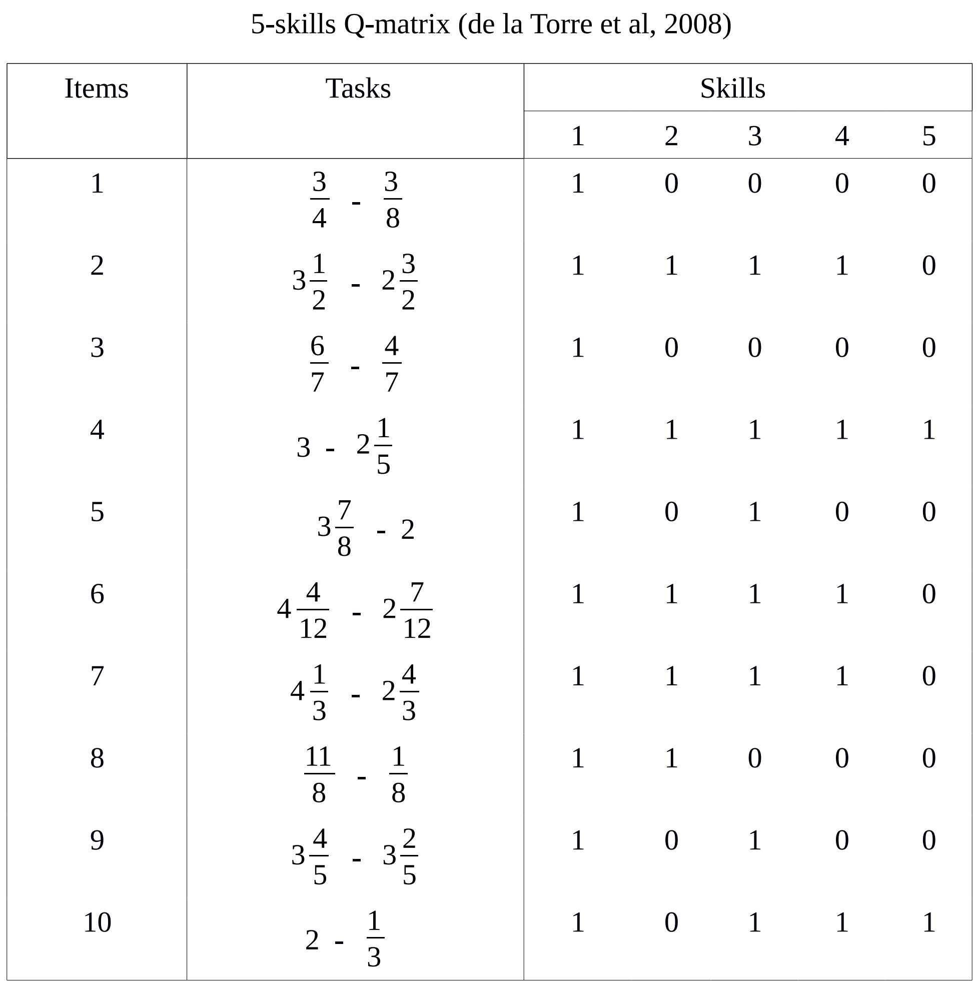
Therefore, a mapping of KCs to tasks is essential to student skills modeling and it has traditionally been done by experts. However, instead of using expert defined knowledge components and KC-item mapping, some researchers use latent factor analysis techniques and revert to using data such driven methods to automate this mapping and to help define, or refine the definition of KCs [5, 6, 7, 8, 9, 10]. For example, the Matrix Factorization (MF) approach, which we study in more details later, decomposes the student performance outcome matrix over a set of items, into a Q-matrix that maps items to skills, and a profile matrix that maps skills to students. It relies on standard matrix algebra for this factorization process. By knowing Q-matrix and profile matrix, we are able to predict the student performance.

The probability of getting a correct answer depends mainly on the mastery level of the skills behind a problem. Additionally she may also require a special skill for the proper integration of all the skills together. In the Knowledge-Learning-Instruction (KLI) framework [11], that kind of special skill is defined as an "integrative knowledge component" that integrates with all other KCs to produce a correct response.

1.2.3 Q-matrix

The term Q-matrix was proposed by Tatsuoka[12]. A Q-matrix is a representation of the mapping of KCs to items (tasks, test questions, etc.) and is a term commonly used in the literature of psychometric and knowledge engineering. Typically, it is a binary matrix and represents relationships between problems/items/tasks/exercises and their associated skills. When questions are proposed to students, the Q-matrix indicates which skills are needed to answer these questions.

Figure 1.1 shows an example of a 5-skills Q-matrix from [8], in which a value of 1 represents



5-skills Q-matrix (de la Torre et al, 2008)

Items	Tasks	Skills				
		1	2	3	4	5
1	$\frac{3}{4} - \frac{3}{8}$	1	0	0	0	0
2	$3\frac{1}{2} - 2\frac{3}{2}$	1	1	1	1	0
3	$\frac{6}{7} - \frac{4}{7}$	1	0	0	0	0
4	$3 - 2\frac{1}{5}$	1	1	1	1	1
5	$3\frac{7}{8} - 2$	1	0	1	0	0
6	$4\frac{4}{12} - 2\frac{7}{12}$	1	1	1	1	0
7	$4\frac{1}{3} - 2\frac{4}{3}$	1	1	1	1	0
8	$\frac{11}{8} - \frac{1}{8}$	1	1	0	0	0
9	$3\frac{4}{5} - 3\frac{2}{5}$	1	0	1	0	0
10	$2 - \frac{1}{3}$	1	0	1	1	1

Figure 1.1 Q-matrix

the requirement of the corresponding skill to solve the problem, and 0 otherwise.

There are many ways to define the underlying skills for a single set of problems. For example, for the well-known Tatsuoka faction algebra set of 20 problems that we use in our studies [12], De La Torre [8] defines the following skills:

(1) convert a whole number to a fraction;

(2) separate a whole number from a fraction;

(3) simplify before subtracting;

(4) find a common denominator;

(5) borrow from whole number part;

(6) column borrow to subtract the second numerator from the first;

(7) subtract numerators;

(8) reduce answers to simplest form.

Others have defined a 4-skills version of the Q-matrix [6]:

(1) convert a whole number to a fraction,

(2) separate a whole number from a fraction,

(3) simplify before subtracting, and

(4) find a common denominator

And yet, we can also find a 3-skills breakdown of the same items [13]:

(1) borrowing from a whole number,

(2) separating a whole number from a fraction and

(3) determining a common denominator

Assignments of these skills to items is a non-trivial and error-prone process. If the Q-matrix is wrongly defined by experts, it will effect the whole student model undoubtedly. Incorrect specification in Q-matrix will result in erroneous diagnosis of student knowledge states [14, 15]. Therefore, validating a Q-matrix would be highly useful and important in providing accurate and effective knowledge assessment and predicting of student performance.

If a problem requires multiple skills, there are two different ways in which each skill can contribute to correctly answer that problem:

- *Compensatory model.* Each skill contributes to the chances of success to the corresponding item.
- *Non-compensatory model.* Each skill is necessary to succeed at the item.

In this thesis, we focus on the non-compensatory model which is also referred to as *conjunctive*. More explanation are given in Chapter 2.

1.3 Motivation

In this dissertation, we address two tasks that are critical in assessment of student knowledge mastery, Q-matrix refinement of the item-to-skill mapping and Knowledge Tracing.

Q-matrix refinement : Q-matrices are generally experts defined and prone to errors. Q-matrix refinement aims to recover and validate Q-matrices [12] that will help to improve the quality of skills in building student models. Therefore, tools to help a tutor, or a designer of a learning environment, validate a given mapping of skills to tasks would be highly valuable. Validating Q-matrix can be seen as improving the domain model statistically [7, 8, 16, 17].

Knowledge Tracing : KT is paramount to intelligent learning environments and numerous different modelling methods have been introduced for that purpose [18, 19, 20, 21, 22]. It takes the general form of determining which skills are considered mastered by a given student based on past performance. Q-matrices and Knowledge Tracing work in unison for achieving fine-grain assessment of skill mastery.

The state-of-the-art for Q-matrix refinement through data driven techniques focuses on skill-item mapping. Instead of this mapping, we will focus on a technique, multi-label classification, that allows us to do refinement based on the whole set of skills for a single item. As will be shown, this approach improves over the existing techniques.

In order to achieve this general goal, we will apply deep learning techniques in assessment of student knowledge level for providing better prediction performance. Early work in this direction was done by Piech [23] who proposed a student modelling technique using a Recurrent Neural Network (RNN) to predict student performance. Their work is known as Deep Knowledge Tracing (DKT). We will extend this framework by incorporating mechanisms that handle item difficulty and the student's *ability profile* which we define as:

Definition 1.3.1 ***Ability profile.*** *A student ability profile is defined as a category of students having similar abilities. The student's ability is evaluated from historical performance according to the relative strengths and weaknesses of the skills in her previous practices.*

The intuition behind the ability profile is that students accumulate general knowledge from solving various problems that involve different skills set, and they can transfer and utilize that learning when faced with new, yet similar problems in their future practices. Moreover, problem solving requires not only the mastery of particular skill set in the problem, but also the ability to integrate and apply these mastered skills properly with deep understanding of the problem [11]. Student's temporal ability profile is designed to serve as a knowledge integration indicator, and transfer mechanism across skills and along practice. Identifying their ability at each time interval will help to consider their transferred learning in the assessment of knowledge mastery.

The other contribution we bring to deep learning based student models is on taking into account item difficulty. The current state-of-the-art models do not explicitly address item difficulty. Yet, research in the fields of psychometrics and EDM [24, for eg.] clearly reveal that even if items have the single common skill, their mean success rate may be very different. Accurate prediction of item outcome must therefore take item difficulty into account and this is one of the goals of this thesis.

1.4 Research Questions

This dissertation focuses on two tasks: using machine learning techniques to refine a Q-matrix and using deep learning techniques to assess students' knowledge state and predict their performance that can be used for providing adaptive techniques to personalize learning environments. Most of my work was conducted on the datasets from ASSISTments and Cognitive Tutor[24]. The main research questions investigated are described below.

1.4.1 Can we validate Q-matrices efficiently with a multi-label classifier and empirical data?

There are numerous algorithms and tools to help an expert map exercises and tasks to underlying skills. While the last decade has witnessed a wealth of data driven approaches aiming to refine expert-defined mappings of tasks to skill, most algorithms are working at the level of individual item-skill mapping [8, 10, 25]. Can we train a multi-label classifier that maps an item to all skills simultaneously instead of one skill at a time? Is that more efficient than state-of-the-art Q-matrix validation methods?

1.4.2 Can we develop a student model over the deep learning framework that accounts for both item difficulty and learning transfer?

Classic student modelling methods such as IRT (Item Response Theory in Section 2.3.1.1) have the advantage of taking item difficulty into account. Moreover, given that IRT relies on a single skill, learning transfer naturally occurs between items (success over past items will be reflected in the chances of success over incoming items). However, IRT lacks the advantage of contemporary student modelling techniques such Bayesian Knowledge Tracing (BKT) [18, 19, 26, 27, 28] and the more recent Deep Knowledge Tracing (DKT) [23], namely the ability to model sequential data and to avoid the assumption that no knowledge improvement occurs during student's learning. That assumption was reasonable in a testing context, but it is unrealistic in learning environments.

The challenge is therefore to develop a student model that has the advantage of modeling learning transfer across skills and take item difficulty into account. We will address these requirements within the latest framework, the deep-learning approach. The notion of temporal student profile will address the learning transfer challenge, and a set of features will allow item difficulty to be integrated into the DKT approach.

The proposed approach will be compared with the state-of-the-art student modelling techniques. We want to evaluate if the proposed models improve knowledge tracing and provide better interpretation on student's ability evolution.

1.5 Contributions

My dissertation works are contributed in student modelling of Educational Data Mining (EDM) and Intelligent Tutoring System (ITS) research with an interdisciplinary approach (combining learning analytic methods from computer science and cognitive science) to tackle a topic which is ultimately focused on monitoring student learning progress for improving their learning gain.

It contains contributions to both Q-matrix refinement (achieved by using ensemble learning technique based on multi-label classification) and knowledge tracing (achieved by using deep learning based learner models). According to experimental results, it improves the student model performance for probabilistic prediction that the next answer will be correct. Student performance prediction is applied to help the educators and learners for improving their learning and teaching process effectively and efficiently.Hopefully, my dissertation can spark more endeavors in EDM and ITS.

As a part of my doctoral program, I contributed to a number of publications, which are directly related to this thesis,

1. Sein Minn, Michel C. Desmarais, Shunkai Fu, "Refinement of a Q-matrix with an ensemble technique based on multi-label classification algorithms", *EC-TEL 2016 — European Conference on Technology enhanced learning* , Lyon, France,September 2016, pp. 165-–178.

2. Sein Minn, Feida Zhu, Michel C. Desmarais, “Improving Knowledge Tracing Model by integrating problem difficulty”, *IEEE International Conference on Data Mining, Ph.D. Forum* , Singapore, Nov 2018, pp. 1505–1056.

3. Sein Minn, Michel C. Desmarais, Yi Yu, Feida Zhu, Jill Jen Vie “Deep Knowledge Tracing and Dynamic Student Classification for Knowledge Tracing”, *IEEE International Conference on Data Mining* , Singapore, Nov 2018, pp. 1182–1187.

4. Sein Minn, Michel C. Desmarais, Feida Zhu, “Dynamic Student Classification on Memory Networks for Knowledge Tracing”, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Macau, April, 2019, pp. 163–174.

CHAPTER 2 BACKGROUND

In this chapter, we will review state-of-the-art models for both of Q-matrix refinement and knowledge tracing problems, and first discuss the nature of the data to analyze.

2.1 Dynamic and Static Performance Data

Q-matrix definition and refinement, and Knowledge Tracing, are the fundamental building blocks of skills modeling and assessment. We must know the skills behind tasks/items in order to perform Knowledge Tracing. The general research framework around these topics is to analyze student performance data, or item outcomes. Outcome is generally represented as a two valued variable, pass or fail, but it may also be continuous or nominal (fair/good/excellent), though we will only encounter binary data in this dissertation.

This data comes in two forms. The first one is static data. Typical examples are snapshot test results over a specific course content. We refer to this data as *static* because we assume no learning occurs during the testing, and therefore the student's knowledge state does not change.

The static data we will analyze comes in the form of a binary matrix, which is called *item response matrix*. It represents rows as students and columns as items respectively.

The second kind of data is *dynamic*, because we drop the assumption that no learning can occur as data is collected. Such assumption would be unrealistic for tutoring systems, since the whole purpose is for the student to learn as they use the system. In such cases, students sometimes try the same question several times, possibly for long time periods. This data can also contain information about hints that are asked by the students, or prompted by the system, as the same item can be tried many times.

The system records the full behavior of students throughout the whole learning phase. Dynamic data necessarily involves the notion of a sequence and is generally time-stamped. However, Q-matrix is identical in both of static and dynamic data. Only profile matrix will be changed in dynamic data because of student's learning.

The work on Q-matrix induction and refinement is conducted over static data in our work. This simplifies the analysis, since otherwise we would need to conduct both Q-matrix analysis and Knowledge Tracing simultaneously, since performance data becomes ambiguous without both information. For eg., item success can be attributed to the mastery of a skill, or to some other missing skills (which are not defined in that item).

The work on Knowledge Tracing will be conducted on dynamic data, mostly because intelligent tutoring systems are the environments that can benefit from a skills modeling component. And as mentioned, they are typically used for periods of a few hours, up to many weeks in a row, and learning is a strong factor in this situation.

2.2 Q-matrix Refinement

Given a Q-matrix obtained from an expert, refinement refers to the task of validating the corresponding item to skills mapping. This is in contrast to Q-matrix induction, where the Q-matrix is entirely derived from data.

Note that Q-matrices derived from data often yield models that are better at predicting student performance from past observations [10]. However, they suffer the issue of interpretability [29]. Skills in such Q-matrices are unlabeled and essentially defined by the items they are linked to. That leads to the issue of interpretation. Moreover, teachers often have predefined sets of skills they aim to teach their students, and it is a more natural task for them to look for test items that will exercise these skills, than to start with a data driven Q-matrix obtained from an algorithm, and figure out which skills might be linked to what competences or knowledge they wish to teach and assess .

2.2.1 Q-matrix

Three matrices are involved in a Q-matrix refinement problem. In the conjunctive model of Q-matrices, they represent the following boolean matrix product:

$$R = A \odot Q$$

where R is the observed response matrix, A is the student profile matrix, and Q is the Q-matrix. They are explained below.

The observed response matrix represents the outcome data. For example, the following matrix is the outcome data of 4 students (respondents) and 9 questions (items).

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}$$

A success is assigned 1 and failure 0 in this matrix.

The Q-matrix, represented by Q, represents the item to skills mapping as shown below for 9 items and 3 latent skills:

$$Q = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

As mentioned, this is a conjunctive matrix (see below) and therefore the first item in this example require skills 1 and 2.

Finally, the student profile matrix, A, which is unknown in real scenario, represents what is often termed the cognitive diagnostic. For the 4 students above, it might look like:

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

For example, the first student only possess skill 2 in this student profile matrix,

In some models, values of both A and R could be a real value between the interval $[0, 1]$. If a standard dot product was used, such real values could be found in A and the product $R = AQ$ might even yield values beyond the range $[0, 1]$. In such cases, rounding to binary values $\{0, 1\}$ is often done.

Recall that a Q-matrix is a mapping between items/tasks/exercices and the underlying skills necessary to their successful completion. These skills can involve factual knowledge, deep understanding of abstract concepts, general problem solving abilities, practice at recognizing patterns and situations, etc. A domain expert defines such mapping and can be a teacher, who may be interested in a specific subset of all possible skills. This subset will generally omit the basic level skills such as reading for a history test, or addition and multiplication for a calculus test.

For example, skills might be a subset that more closely relates to a given topical or pedagogical perspective at the expense of alternative perspectives. A tutor may not care much about general problem solving abilities that require months to acquire, and focus on factual knowledge and rules that are easier to teach and assess, even though both the problem solving skills and factual knowledge are involved in the training and assessment material.

Whatever the motivation is for defining the skills behind the successful completion of tasks, a first point to emphasize is that for the same tasks, one skill definition may be considered appropriate for one context whereas another will be required for another context. A second point to emphasize is that the definition of skills behind tasks, or the converse, the definition of tasks for a given set of skills, are non-trivial and error-prone processes.

Therefore, tools to help a tutor, or a designer of a learning environment, validate a given mapping of skills to tasks would be highly valuable. Let us refer to this endeavor as the problem of *Q-matrix refinement*, where the Q-matrix represents the mapping of tasks to skills.

2.2.2 Diagnostic Classification Models

Q-matrices come in many flavours and have a long history within the psychometrics field. We review this work here and the variety of Q-matrices we find, but note that in this dissertation, only the *conjunctive* type of Q-matrices considered, or *non-compensatory*.

We refer to models that combine a student profile matrix and a Q-matrix as Diagnostic Classification Models (DCM) [5]. Many models come from psychometrics. Compared to the classic Item Response Theory with continuous latent attributes, which is covered later, DCM represent discrete latent attributes. These attribute patterns are binary vectors with 1 indicating mastery of that latent attributes and 0 otherwise. These patterns provide the feedback to teachers to help with designing remedial instructions. They are alternatively called as:

- Restricted Latent Class Models [30]
- Latent Response Models [31]
- Multiple Classification Latent Class Models [32]
- Cognitive Diagnosis Models [6]
- Cognitive Psychometric Models [33]
- Structured Item Response Models [34]
- Structured Located Latent Class Models [35]

These models are somewhat similar. We will use the term Diagnostic Classification Models (DCMs) to embrace the whole family. Generally, we can assume DCMs belong to Latent Variable Models (LVM). For LVM, there are two type of variables, the manifest variable (observed variable) and latent variable (unobserved variable).

In DCMs, only models with manifest variables and latent discrete variables are considered, that is, statistically they belong to Latent Class Analysis (LCA). A variety of DCMs has been proposed over the past decade. Depending on the nature of the models, DCMs can be classified into two categories,

- *compensatory*
- *non-compensatory*

Non-compensatory DCM models presume that each item requires a particular set of skills and lacking any one of them would lead the student to a failure. The model is called deterministic-input noisy-and (DINA) model [9]. It is a well-known non-compensatory model. By contrast, the noisy-or-gate (DINO) [6] is at the other extreme of this requirement: any skill is sufficient for success. The DINA model is also considered a *conjunctive* model, whereas DINO is considered a *disjunctive* one. Other models where each skill marked as relevant to a task in a Q-matrix and contributes to the chances is simply considered *compensatory.* A matrix factorization model such as proposed in [10] is one such model. Regardless of their formulation, all of the above models share the general idea of relating each item to single or multiple latent attributes.

Note that the DINA model relies on a conjunctive Q-matrix and will be used as a basis for the Q-refinement studies in this dissertation.

2.2.2.1 Compensatory models

An example of a compensatory model is found in a real study to assess the prevalence of pathological gambling traits [5]. Respondents who agreed with the following assertion:

"Gambling got me into trouble over my finance situation."

may involve the presence of two attributes:

Attribute 1: breaks the law such as forgery, fraud, theft, or embezzlement to finance gambling.

Attribute 2: depends on money provided by others to relieve a desiderata situation caused by gambling.

Any attribute may trigger a positive answer to the assertion, but not necessarily. However, the presence of both attributes is more likely to trigger a positive answer than any single attribute alone.

A similar argument can be made for two skills linked to an item. Mastery of any of the two skills (traits) may lead to succeed at the item, but the mastery of both will make the success more likely.

DINO: The deterministic input, noisy-or-gate (DINO) model is a well-known *disjunctive* model in skills modeling [6]. The DINO model is a full compensatory DCM that has two parameters at the item level, slip and guess. Respondents have a high probability of providing a correct answer with any one of the required skills instead of all of the required skills.

Upon the given response matrix (which represents respondents provide a correct response to items or not is represented with binary value), we are going to assess consisting of a domain of skills or attributes behind those items. An assessment consisting of I items consider measuring a domain of K attributes or skills.

Let $Y_{ij}, i = 1, 2, ..., I, j = 1, 2, ..., J$, be a binary 0/1 response for item i by respondent j with 1 representing the respondent providing a correct response to the item and 0 otherwise.

The attribute pattern for respondent j, α_j is a vector of length K with binary 0/1 elements with 1 meaning the respondent has mastered the attribute and 0 otherwise. For a test requiring K attributes, respondents can be classified into one of the 2^K possible attribute patterns.

$$P(X_{ij} = 1|\xi_{ij}) = (1 - s_i)^{\xi_{ij}} g_i^{1-\xi_{ij}}$$

where

$$\xi_{ij} = 1 - \prod_{i=1}^{I} (1 - \alpha_{jk})^{q_{ik}}$$

$$s_i = P(X_{ij} = 0|\xi_{ij} = 1)$$

$$g_i = P(X_{ij} = 1|\xi_{ij} = 0)$$

j : the respondent

i : the item

X_{ij} : the response outcome of j to i

α : the attribute pattern for that respondent

- s_i: slip (the chances that a student that masters any required skills has a wrong answer to the item i).
- g_i: guess (the chances that a student that masters none of the required skills has a correct answer to the item i).

Based on the formula above, item i is correct with two possible probabilities. If a respondent has mastered none of the required attributes, they are still likely to provide a correct answer via guessing, so the correct response probability in this case is g_i. When a respondent has mastered at least one of the required attributes, the correct response probability is $1 - s_i$. The DINO model can be estimated using Markov chain Monte Carlo (MCMC) [6] or as a constrained log-linear model with latent classes.

2.2.2.2 Non-compensatory models

Non-compensatory DCMs require all skills for a success to an item. For instance, to solve the math problem of

$$5 \times 3 - 9 =?$$

the respondent requires all of these two elementary math skills:

- multiplication
- subtraction

DINA: The most popular DCM non-compensatory model is DINA, the noisy-and-gate. Similar to DINO model, DINA model also takes into account the possibility that a respondent with all required skills misses an item and possibility through careless errors (slip parameter) and the possibility that respondent who lack at least one of the required skills gives a correct response by guessing (guess parameter).

For the DINA model,

$$P(X_{ij} = 1|\xi_{ij}) = (1 - s_i)^{\xi_{ij}} g_i^{1-\xi_{ij}}$$

where:

$$\xi_{ij} = \prod_{i=1}^{I} \alpha_{jk}^{q_{ik}}$$

$$s_i = P(X_{ij} = 0|\xi_{ij} = 1)$$

$$g_i = P(X_{ij} = 1 | \xi_{ij} = 0)$$

where:

- j: the respondent
- i: the item
- X_{ij}: the response outcome of j to i
- α: the attribute pattern class that the respondent belongs to
- s_i: slip (the chances that a student that masters all required skills has a wrong answer to the item i).
- g_i: guess (the chances that a student that does not master all required skills has a correct answer to the item i).

Response outcome is binary: $\{0, 1\}$. When $\xi_{ij} = 1$, the respondent j has mastered all the required skills and 0 otherwise.

The synthetic data used in this study is generated with the DINA model.

2.2.3 Q-matrix Refinement Techniques

We review three Q-matrix refinement methods that are prevalent and often used as state-of-the-art comparison.

2.2.3.1 MinRSS

MinRSS [25] relies on the notion of an ideal response pattern and aims to minimize the distance between real response pattern and ideal response pattern.

For any given Q-matrix, there is an ideal response pattern (ideal response vector) for each student profile pattern (profile vector). An ideal response vector corresponds to the expected success or failure to an item according to the skills the student masters, and the skills required by this item.

In a conjunctive Q-matrix and assuming a required skill vector q and a mastered skill vector v, the ideal response vector, r, is the expected outcome of:

$$r = \|\neg(p \wedge \neg q)\| > 0$$

To take a few examples, we would have:

p	q		Ideal response vector
$(0,0,1)$	$(1,1,1)$	$\rightarrow$	1
$(1,0,1)$	$(1,1,0)$	$\rightarrow$	0
$(1,1,1)$	$(1,1,1)$	$\rightarrow$	1
$(0,0,1)$	$(0,0,0)$	$\rightarrow$	1

The set of ideal response patterns corresponds to the expected outcome values for all combinations of profile patterns, under the assumption that there are no slip and guess factors. For example, for a 4-students and 3-skills profile matrix, the set of possible combinations of skills (profile patterns) is $2^{4\times3}$, and there will be a corresponding ideal response pattern for each. Note that two different profile patterns may map to the same ideal response pattern (binary vector of a student's responses to items): unrequired skills will lead to the same ideal response pattern, regardless of whether the student masters them or not. Therefore, the total number of ideal response patterns may be much lower than the number of profile patterns. Of course, real data contains guesses and slips, or noise of different kind, and the problem is how to measure the difference between the real pattern and ideal pattern. The most common metric for binary data is Hamming distance:

$$d_h(r,\eta) = \sum_{i=1}^{J} |r_i - \eta_i| \tag{2.1}$$

where:

i: the item

r: the real response vector

η: the ideal response vector

Chiu [25] considered a more refined metric. The idea is that if an item has a smaller variance (or entropy), then it should be given higher weight with the formula:

$$d_{\omega h}(r,\eta) = \sum_{i=1}^{J} \frac{1}{\bar{p_i}(1-\bar{p_i})} |r_i - \eta_i| \tag{2.2}$$

where:

r: the real response vector

η: the ideal response vector

$\bar{p_i}$: is the proportion of correct answers of item i

Equipped with this metric, we can find the most approximate ideal response matrix and then find the correspondent profile matrix A. With these results, a powerful method was proposed to update the Q-matrix [7].

First, a squared sum of errors for each item i can be computed by

$$RSS_i = \sum_{j=1}^{N} (r_{ij} - \eta_{ij})^2$$

N: the number of respondents

j: the respondent

i: the item

Then, the item with the highest RSS is chosen and its corresponding q-vector is updated. All other possible q-vectors are tested to calculate their RSS and the q-vector giving the lowest RSS is chosen to replace the original q-vector. The method is named MinRSS because aims to minimize this value. The Q-matrix is thus changed and the whole process will be repeated, except that the previously changed q-vector would be left out of the candidates pool for the next round. The whole procedure terminates when the RSS for each item no longer changes.

This method has a high consistency quality as shown by Wang [36]. That is, it has good performance under different underlying conjunctive models.

2.2.3.2 MaxDiff

The second well-known Q-matrix refinement technique is MaxDiff. It rests under the DINA model. For every item j, there are two model parameters, slip s_j and guess g_j. de la Torre et al [8] proposed that a correctly specified q-vector for item j should maximize the difference of probabilities of correct response between examinees who have all the required attributes and those who do not.

Denote the 2^K binary vectors defined by the K number of skills as $\alpha_l, l = 0, 1, .., 2^K - 1$,and let α_0 correspond to the null vector $(0, 0, ..., 0)'$. Then q_j is the correct q-vector if:

$$q_j = \arg\max_{\alpha_l} [P(X_j = 1 | \xi_{ll'} = 1) - P(X_j = 1 | \xi_{ll'} = 0)] = \arg\max_{\alpha_l} [\delta_{jl}]$$

where $\xi_{ll'} = \prod_{k=1}^{K} \alpha_{l'k}^{\alpha_{lk}}$.

Since $P(X_j = 1|\xi_{ll'} = 1) = 1 - s_j$ and $P(X_j = 1|\xi_{ll'} = 0) = g_j$, we get:

$$q_j = \arg\max_{\alpha_l}[1 - (s_j + g_j)]$$

Maximizing this difference is equivalent to minimizing the sum of the slip s_j and guess g_j parameters.

A general approach is to test all q-vectors to find the maximum δ_{jl}, but that is computationally expensive. [8] proposed a greedy algorithm that adds skills into a q-vector sequentially.

1. δ_{jl} is calculated for all q-vectors which contains only one skill.
2. Choose the one with biggest δ_{jl}.
3. δ_{jl} is calculated for all q-vectors which contains two skills including the previously chosen one.
4. Choose the largest q-vector δ_{jl} again.
5. Repeat until adding skills no longer increases δ_{jl}

However, this algorithm requires knowing s_j and g_j in advance. For real data, they are estimated with the EM (Expectation Maximization) algorithm [9].

2.2.3.3 Matrix Factorization

Matrix Factorization (MF) is a commonly used technique in Data Mining. An influential paper on the special case of Non-negative matrix factorization (NMF) in pattern recognition [37] popularized the approach. It was introduced for Q-matrix refinement by Desmarais [38]. Numerous applications of MF and NMF later followed in the EDM field [10, 39, 40].

Boolean Matrix Factorization (BMF), in particular, gained attention over the years [41, 42]. It exerts stronger restrictions on the factor matrix that each entry needs to be binary.

Another method is the Alternate Least Squares (ALS) algorithm. The algorithm starts with an initial Q-matrix, Q_0, which is the target matrix to refine. A least-squares estimate of the skills matrix $\hat{S}_0$ can is obtained with:

$$\hat{S}_0 = (Q_0^{\mathrm{T}}\, Q_0)^{-1}\, Q_0^{\mathrm{T}}\, R \tag{2.3}$$

A new estimate of the Q-matrix, $\hat{Q}_1$, is again computed with the least-squares estimate:

$$\hat{Q}_1 = R\,\hat{S}_0^{\mathrm{T}}\,(\hat{S}_0\,\hat{S}_0^{\mathrm{T}})^{-1} \tag{2.4}$$

The process cycle through equations (2.3) and (2.4) to estimate in turn $\hat{S}_1$, $\hat{Q}_2$, etc. The values converges to obtain a factorization that closely approximate R.

All these matrix factorization techniques aim to minimize the loss function

$$\mathcal{L}(Q) = ||R - AQ^{\mathrm{T}}||$$

where Q is a real matrix, positive matrix and binary matrix respectively for ALS and NMF. For BMF, the matrix dot product is replaced by $\odot$ which is the binary product. There are several methods to implement this factorization.

2.3 Knowledge Tracing

Once we have the mapping of skills to items, we can tackle the problem of modeling the Knowledge Tracing problem which constitutes another fundamental building block of student modeling.

Knowledge Tracing is the assessment of students' knowledge and the prediction of whether students may or may not answer a problem correctly based on past item test outcomes. KT leverages machine learning and data mining techniques to provide better assessment, enabling supportive learning feedback and adaptive instructions. By tracing the knowledge state of the students, we can optimize instruction for individualization and personalization. Recent state-of-the-art methods are either based on modelling of temporal latent variables or factor analysis with temporal features.

Early work dates back to the late 1970s and a wide array of Artificial Intelligence and Knowledge Representation techniques have been explored, of which we can mention rule-based and Bayesian representation of student knowledge and misconceptions, skills modelling with logistic regression in Item Response Theory, Bayesian student modelling methods, case-based reasoning, and, more recently reinforcement learning and deep learning [43, 44, 45]. One can even argue that most of the main techniques found in Artificial Intelligence and Data Mining have found their way into the field of ITS, and in particular for the problem of knowledge tracing, which aims to model the student's state of mastery of conceptual or procedural knowledge from observed performance on tasks [18]. Recurrent Neural Networks (RNNs) models, used in the Deep Knowledge Tracing (DKT) approach [23], are a more recent example. They show better predictive performance than the preceding approaches, in part because they are known to preserve past information in sequential data, such as student outcome traces. We will address in this thesis how to improve their capacity to model skill profiles during the long-term learning process and improve their predicting performance.

In this section, we review five of the best known state-of-the-art student modelling methods for knowledge tracing and estimating student's performance (see [46] for a review). They are chosen either because of their predominance in psychometrics (IRT) or Educational Data Mining (BKT), or because they are best performers (PFA, DKT, DKVMN). All of these models use a Q-matrix to construct a cognitive model based upon these observed behaviours and to apply the models to make predictions.

2.3.1 Early Knowledge Assessment and Knowledge Tracing Models

This section presents the fundamental theory and models which we use as baseline comparisons in our experiments.

2.3.1.1 Item Response Theory (IRT)

Item Response Theory (IRT) is a prominent approach that has been studied in the field of psychometrics for over decades and formed the basis of the first personalized assessment environments, Computer Adaptive Tests [46].

The main idea behind IRT is that the probability of success on a task (item) increases as a function of the level of mastery of a single latent, unobserved skill behind all tasks. It assumes the student knowledge state is static, such as in exam test data. Student knowledge state is represented by her proficiency when completing an assessment. The static knowledge state assumption implies that proficiency. In addition to the static knowledge assumption, in its original form, IRT models a single skill and assumes the test items are *unidimensional* in that respect [47, 48, 49, 50].

In assessments such as standardized tests, learners are usually modelled by one static latent variable – Rasch model [51], which makes it unfit for fine-grained cognitive diagnosis. However, it enjoys a strong theoretical background both in terms of being grounded in psychometric measurement, and a sound mathematical framework with proven algorithms [52].

IRT takes dichotomous item response outcomes and assigns student i with a proficiency θ_i, which is not changing during the exam. Each item j has its own difficulty β_j. The main idea of IRT is estimating a probability that student i answers item j correctly by using student ability and item difficulty,

$$\begin{aligned} P_{ij} &= f(\theta_i - \beta_j) \qquad (2.5) \\ &= \frac{1}{1+e^{\theta_i - \beta_j}} \qquad (2.6) \end{aligned}$$

where f is the logistic function, θ is the student proficiency on the topic tested, and β_j is the difficulty of item j.

There are variants called multidimensional-IRT that can handle two and more dimensions [53, 54] but their complexity is much greater, and they have not yet been used widely in personalized learning environment [46]. Often, a single and different IRT model is used for each skill.

An important consideration is that IRT is not considered a *Knowledge Tracing* approach to the extent that it makes the assumption that the student does not learn during the process of testing. It is considered a *Knowledge Assessment* approach and each new item tested helps bring information to refine the estimate of the knowledge state, θ. We will see later that other approaches explicitly model the learning process and are therefore named *Knowledge Tracing* approaches.

Note that when IRT or similar knowledge assessment models are used in the context of a learning environment, where the knowledge state is *expected* to change, the assessment is often done on the first attempt to items, in the context of an exercise where multiple attempts are allowed. Alternatively, multiple trials can also be counted as a different items and an index t will be used on β ($\beta_{j,t}$) that corresponds to the trial.

IRT*: Recently, Wilson [47] proposed an IRT* (Bayesian extensions of IRT model) model that outperforms state-of-the-art knowledge tracing models, it uses Bayesian approach and regularizes on $\log P_{ij}$ by imposing independent standard normal prior distributions over each θ_i and β_j. It needs to maximize on log posterior probability of $\{\theta_i, \beta_j\}$ given the response data $\{r(i,j,t) \in D\}$, where response $r \in \{0,1\}$ and t is the time of each attempt.

$$\begin{aligned} \log P(\{\theta_i\},\{\beta_j\}|D) = \sum_{(i,j,r,t)\in D} & r \log f(\theta_i - \beta_j) + (1-r) \\ & \log(1 - f(\theta_i - \beta_j)) - \frac{1}{2}\sum_i \theta_i^2 - \frac{1}{2}\sum_j \beta_j^2 + C. \end{aligned} \qquad (2.7)$$

IRT* leverages the information of both items and students that directly interact in the system. Maximum a posteriori (MAP) estimates of θ_i and β_j are computed by Newton-Raphson method. We will refer this model as IRT*.

2.3.1.2 Bayesian Knowledge Tracing (BKT)

In an interactive learning environment, the assumption of a static student knowledge state is no longer tenable and models that drop this assumption have emerged in the late 1990s along with the term “knowledge tracing”, as opposed to “knowledge assessment”, as discussed earlier.

Bayesian Knowledge-Tracing (BKT) is the earliest approach to model a learner’s changing knowledge state and is arguably the first model to relax the assumption on static knowledge states. It was introduced for knowledge tracing within a learning environment [18].

In its original form, BKT assumes a single skill is tested per item. Questions are tagged with only one knowledge component (KC) or skill. Based on a sequence of outcomes, the learner’s state of skill mastery is inferred at each time step. This approach to modelling is particularly relevant for tutors that use exercises and scaffolding as the main vehicle for learning and that monitor fine-grained skill mastery to decide on the next step.

The BKT model can be considered as a Markov Model to the extent that the probabilities above depend either on fixed parameters and on the state in the previous time step, $t-1$. BKT models can also be considered simple Dynamic BNs, which are a special class of BNs for representing temporal information. Figure 2.1 illustrates a sample structure of such network for a sequence of two observations of outcomes, incorrect and correct, and the inferred mastery states, from “not learned to “learned”.

The standard BKT model is comprised of 4 parameters which are typically learned from the data while building a model for each skill. The model’s inferred probability mainly depends on the parameters which are used to predict how a student masters a skill given that student’s chronological sequence of incorrect and correct attempts to questions of that skill thus far.

BKT takes four probabilities:

- $P(L_0)$: the probability a student knows the skill before attempting the first problem;
- $P(T)$: the probability a student, who does not currently know the skill, will know it after the next practice opportunity;
- $P(G)$: the probability a student guesses a question and gets a correct answer despite not knowing the skill (*guess*); and

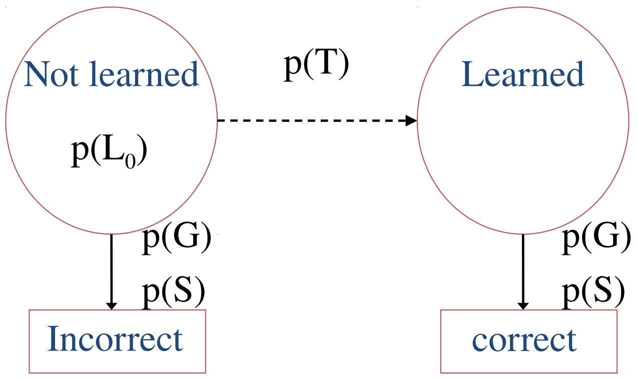


Figure 2.1 Standard BKT structure example.

- $P(S)$: the probability a student answers a question incorrectly despite knowing the skill (*slip*).

In a typical learning environment with BKT, the estimate of student mastery of a skill is continually updated every time student gives a response to an item. The probability that a student knows the skill at time t is:

$$P(L_t|Correct) = \frac{P(L_{t-1})(1 - P(S))}{P(L_{t-1})(1 - P(S)) + (1 - P(L_{t-1}))P(G)} \tag{2.8}$$

$$P(L_t|Incorrect) = \frac{P(L_{t-1})P(S)}{P(L_{t-1})P(S) + (1 - P(L_{t-1}))(1 - P(G))} \tag{2.9}$$

$$P(L_t) = P(L_t|Action) + (1 - P(L_t|Action))P(T) \tag{2.10}$$

First, the probability that a student knows the skill before the response is updated with $Action$ by using the equations 2.8 or 2.9 according to the response evidence. Then, the system estimates the possibility that the student learns the skill during the problem step by using equation 2.10. With Bayesian knowledge tracing algorithm used in cognitive tutoring system, the student will be instructed to practise similar problems with a particular skill, when the system does not recognize that the student has sufficient knowledge about that skill (e.g. the probability that the student knows the skill is less than 95%) [55].

Several extensions of BKT have been introduced, such as the contextualization of estimates of guessing and slipping parameters [55], estimates of the probability of transition from the use of help features [19], and estimates of the initial probability that the student knows the skill [26]. Item difficulty has also been integrated in this model [27].

2.3.1.3 Performance Factor Analysis (PFA)

Performance Factor Analysis (PFA) was adapted from Learning Factor Analysis (LFA, [56]) to allow the creation of a "model overlay" that traces predictions for individual students with individual skills so as to provide the adaptive instruction to automatically remediate current performance [57].

LFA accumulates learning for student i using one or more skills k.

$$\log P_{ij} = \theta_i + \sum_{k \in KCs} (\beta_k + \gamma_k n_{ik}), \tag{2.11}$$

where

- θ_k the ability for student i
- β_k the bias for the skill k
- γ_k the bias for prior attempt to skill k
- n_{ik} the number of attempts of student i on skill k

This model is an extension of the Rasch model (unidimensional IRT) which has an equivalent form to Equation 2.11 with γ_k set to 0 and only a single β_k value.

[57] adapted the LFA model with sensitivity to the indicator of student learning performance. The PFA model allows conjunction by summing the contributions from all skills needed in a performance. This kind of "compensatory" model of multi-skill learning allows the lack of one KC to compensate for the presence of another in addition to showing conjunctive effects.

PFA also relaxes the static knowledge assumption and models multiple skills simultaneously [57]. Its basic structure is:

$$\log P_{ij} = \sum_{k \in KCs} (\beta_k + \gamma_k s_{ik} + \rho_k f_{ik}), \tag{2.12}$$

where

- β_k : the bias for the skill k
- γ_k : the bias for success attempt to skill k

- ρ_k : the bias for failure attempt to skill k
- s_{ik} : the number of successful attempts of student i on skill k
- f_{ik} : the number of failure attempts of student i on skill k.

It does not consider student proficiency θ, because it assumes that θ cannot be estimated ahead of time in adaptive situations [57].

2.3.2 Neural Networks based Knowledge Tracing Models

In the previous section, we covered the foundations and state-of-the-art of student skills modeling, but omitted the developments in neural networks based knowledge tracing that occurred within the last five years or so. Our contribution falls within this line of work which we review in the current section.

2.3.2.1 Neural Networks

Basics concepts

A neural network (NN) is a network of artificial neurons, or nodes, for solving and prediction of complex real-world problems. It is inspired by biological neurons of human brain and the connections between each neuron. Mcculloch [58] initially developed the concept of a computational neuron, that allows taking multiple input values x_i and provides an output value $\hat{y}$ with the values of (0 or 1) for representing whether the neuron is quiescent or excited respectively.

An early version of NN is the perceptron, introduced by Rosenblatt [59]. It is the simplest processing unit that makes decisions by weighing up incoming evidences. This process is

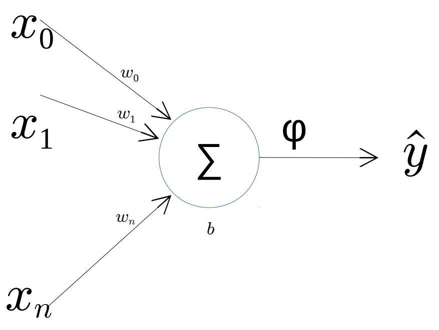


Figure 2.2 Diagram of a perceptron neuron

performed by applying weights, w_n, and bias, b, to an input vector, and an activation function φ that breaks the linearity of the computations and allows the learning of more complex patterns. See figure 2.2. Many different activation functions are proposed to construct perceptrons in today's architectures, for example the sigmoid, hyperbolic tangent, and softmax functions, and the rectified linear unit function (ReLU) that is widely used in deep learning architectures.

The perceptron initially performs linear combination of the input values $x_0, .., x_n$ along with weights $w_0, .., w_n$ and bias b, then utilizes the activation function φ to obtain the output value $\hat{y}$ by following equation:

$$\hat{y}_t = \varphi(\sum_{n=1}^{N} x_n.w_n + b). \tag{2.13}$$

For the task of multinomial classification (e.g classifying an image of digit numbers, 0 to 9), we can apply a perceptron for each class, as illustrated in Fig 2.3. The prediction of a class corresponds to the output from each perceptron. In this case, there is a standard equation to calculate the probability for each of K outcomes by using *softmax* function:

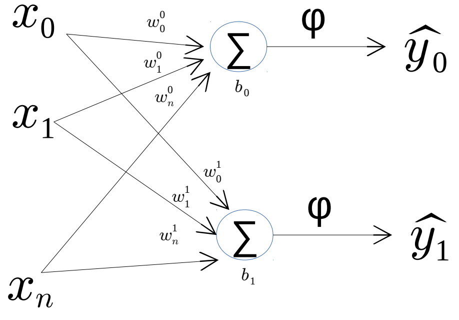


Figure 2.3 Diagram of a multi-class perceptron

$$P(\hat{y}_i) = \frac{e^{\hat{y}_i}}{\sum_i^K e^{\hat{y}_i}} \tag{2.14}$$

Single-class perceptrons or multi-class perceptrons have a single decision layer to yield final prediction which limits the range of problems those they can solve. Combining multiple perceptrons in a neural network provides promising result for solving more complex problems. The output of each perceptron serves as the input to another by stacking multiple perceptrons one after one another. The outputs from hidden layer will server as inputs to the next layer (See in Fig 2.4). We distinguish between three different type of layers: the input layer, the hidden layer(s) and output layer.

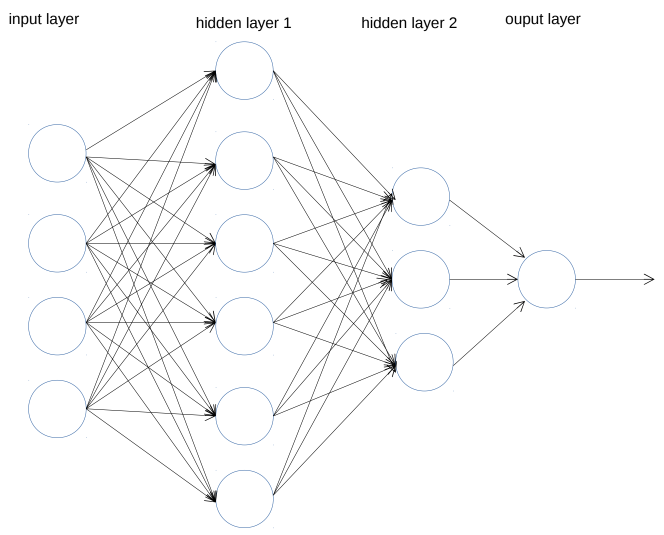


Figure 2.4 A multi-layer neural network.

Akin to the perceptron, an activation is applied at each layer. The most common activation functions are:

- sigmoid function

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.15}$$

- Rectified Linear Units (ReLU)

$$f(x) = max(0, x) \tag{2.16}$$

- hyperbolic tangent (tanh)

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.17}$$

Training

During the training phase of neural networks, weight parameters are learned by being iteratively modified such that the model fits well with data for a given task. The metric determines how the model performs effectively on that task, is defined by the loss function. The loss function provides a single scalar value that represents how far is the model's predictions from the actual observed values in the test or training set. There are two common loss functions based on the types of problem, those that rely on cross-entropy for multi-class classification tasks, and the mean-squared error for regression tasks.

Deep neural networks

If the network contains multiple hidden layers, it is referred as a deep neural network (DNN). We can distinguish two types of DNN according to the direction of connections:

- Feedforward networks: the flow of information is in a single direction.
- Feedback (recurrent) networks: the signal is allowed in both directions, such as within a layer or in a previous layer.

Feedforward networks are widely used because of their stability, but we will review later the Recurrent Neural Network (RNN) architecture that has the power to maintain a longer-term memory of input patterns. They are used with sequential data in problems that can be framed as "what will happen in the next given condition". RNNs represent connections between nodes from a directed graph along a temporal sequence to capture temporal dynamic behaviour. For example LSTM (long short term memory) remembers the things that have happened in the past and find the patterns across time for making next prediction sense.

Yet another common type of DNN architecture is the Convolutional Neural Networks. They are often applied in problems of prediction like image classification, but their use now extends to language models and various types of classification tasks. They are made up of three basic layers called convolution layers, pooling layers, and fully connected layers. It usually processes a grid of values such as 2D matrix of an image and outputs a result.

Deep learning has effectively had an impact on a wide range of fields like natural language processing (such as machine translation, language modelling, question answering [60, 61, 62, 63]), image classification and computer vision (such as image captioning [64, 65]). While it is beyond the scope of our work to include a comprehensive foundation in deep learning, we refer interested readers to the introductory textbooks [66, 67]. More recently some researchers have also started to apply CNNs to problems with sequence of values and gotten some promising results [68].

2.3.2.2 Deep Knowledge Tracing (DKT)

Deep learning has only relatively recently been used for Knowledge Tracing, but as it did in many domains before, their performance challenges the state-of-the-art.

Piech [23] introduced Deep Knowledge Tracing (DKT) at the NIPS 2015 conference. Akin to BKT, it uses data in which skills are tried and the performance outcome to predict future sequence attempts. It encodes skill and student response attempt in a one-hot feature input vector as input for each time t. The output layer provides the predicted probability that the student would answer that particular next problem correctly at time $t + 1$.

DKT uses Recurrent Neural Networks (RNNs) to represent the latent knowledge space of students along with the number of practices dynamically. The increase in student's knowledge through an assignment can be inferred by utilizing the history of student's previous performance.

We will not describe in full details DKT here, but will cover it in Chapter 4 when we introduce our approach that extends this architecture.

2.3.2.3 Dynamic Key-Value Memory Network (DKVMN)

We end this chapter by mentioning another DNN architecture that is named Memory Networks [69] that integrates a long-term read-write memory component. This architecture addresses the shortcoming of RNN to memorize large amount of important facts from the past. It was originally applied to the question-answering task in natural language, but has recently been applied to the task of assessing a student's mastery of concepts with a specific architecture named Dynamic Key-Value Memory Network (DKVMN, [70]). Assessment of mastery level on particular skill are stored in memory slots and controlled by read and write operations through additional attention mechanisms.

DKVMN incorporates a matrix called key memory, which stores the concept representations and the other dynamic matrix called value memory, which stores and updates the data about student's understanding (concept state) of each concept. The static and dynamic matrices respectively represent immutable and mutable objects. After the training, the keys will be created. Once key memory has been created, it is fixed and immutable during testing. However the knowledge state of the students will be updated through value memory across time during both of training and testing phases.

Akin to the DKT architecture, we postpone its detailed description to Chapter 5 when we introduce an extension to this architecture.

CHAPTER 3 Q-MATRIX REFINEMENT

In this chapter, we describe our first contribution published at the EC-TEL conference [71], an ensemble learning technique based on multi-label classification for refining Q-matrices. It builds upon the principle of using an ensemble technique to combine refinements from different algorithms [40], but instead of working at the level of the whole Q-matrix, it works at the item level and uses the multi-label classification technique to refine the mappings of skills.

We describe the method, setup, analysis, and results of a performance assessment of the Q-matrix refinement approach.

3.1 Refinement of Q-matrix with ensemble learning

Experts define the mapping of item to skills based on their own judgement. This is error prone considering that skills may not be perfectly aligned with some items, may be ill defined, or may simply involve counter intuitive skills that are not defined in the skill set. Wrongling defined mapping will result in incorrect cognitive diagnostic.

The Q-matrix refinement consists in starting with an expert-given Q-matrix to identify potential flaws based on student performance data. This can be achived with a number of techniques that work directly with the data, as we will see later, or it can use the output of each technique and hope to extract better information than any individual one. This is the approach adopted here.

The three Q-matrix refinement techniques described in chapter 2: MinRSS, MaxDiff, and Matrix Factorization are substantially different methods to refine a Q-matrix. In that respect, their respective outcome may be complementary, and we can hypothesize that they can be combined to provide a more reliable output than any single one. The ensemble learning approach does exactly this. It aims to combine algorithms that each make individual predictions and decide from the different predictions which is the most accurate.

3.1.1 Ensemble learning for Q-matrix refinement

Ensemble learning is a family of methods that rely on multiple learning algorithms to obtain better predictive performance. Like all ensemble learning approaches, the main idea of using ensemble techniques for Q-matrix refinement is to optimize the choice of outcome from different Q-matrix refinement algorithms.

However, to train an ensemble learning algorithm, we need labeled data that specifies which skills are mapped to which items. [40] used synthetic data for that purpose, for which the ground truth is known. A large number of Q-matrices are created and synthetic student performance data is generated to simulate the responses of hypothetical students with the given Q-matrices.

The process is summarized in figure 3.1 and consists in the following steps:

(1) Start with the QM that we wish to validate and refine.

(2) Generate all permutations of that matrix for producing syntheic Q-matrices (or a sample of, if the matrix is large). This consitutes the ground truth from which labels are obtained for conducting the supervised training.

(3) Generate synthetic performance data from a permutated QM. The DINA model serves as the generative process and a set of random student skill mastery profiles is also generated to produce the synthetic data. This data set is considered *labeled* because the ground truth QM is known.

(4) Introduce a single perturbation in the ground truth QM of (2).

(5) The perturbated QM is fed to the data driven refinement algorithms. Each algorithm outputs the cells that it considers incorrect. The output of this step therefore is an estimate of the validity of each cell of the perturbated QM.

(6) The output of (5) and the true value of each QM cell from (3) is given to the ensemble algorithm (Multi-label classifier), possibly along with other features such as the number of positive values of the line and column of the Q-matrix, and measure the likeliness of the cell to be wrongly classified (see [40] for details).

(7) Finally, outputs from three data driven methods is fed into Multi-label classifier as features and make the prediction of labels. Those predicted labels are converted into final refined Q-matrices as output. a test set (holdout) of the ground truth is used in the evaluation process.

More details on the generation of the synthetic data are provided in section 3.2.2

This process allows supervised learning of the ensemble learning algorithms. The aim is to identify the algorithms that are most effective in a given context by using the synthetic data to train the ensemble algorithm.

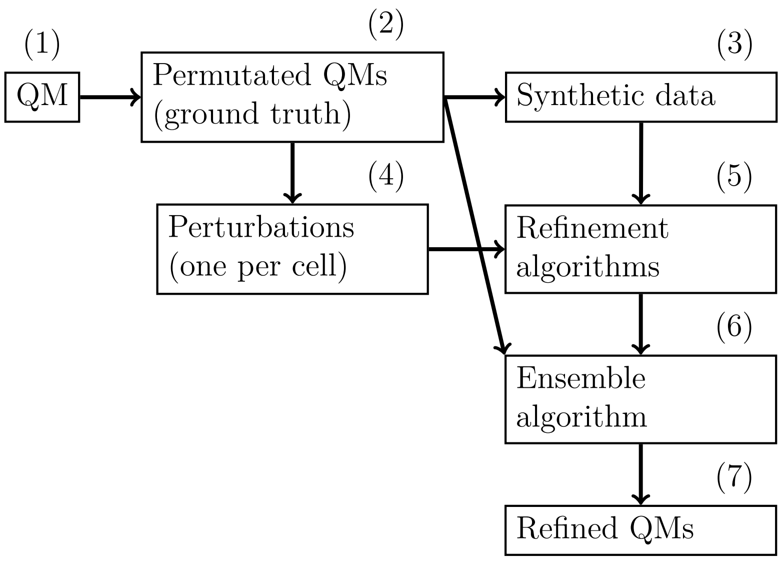


Figure 3.1 Refinement Procedure of each Q-Matrix QM_i

We follow a similar process, but apply ensemble learning based on multi-label classification over each proposed q-vectors and predicts more reliable q-vectors for Q-matrices. Two techniques are proposed based on two multi-label classification methods:

- Binary Relevance method (BR)
- RAndom k-labELsets method (RAkEL)

They are covered later in this chapter, section 3.1.3.

The motivation for the multi-label approach is that it can train the ensemble technique at the item level instead of each matrix cell in isolation. It seems intuitively appealing that while each item is different and independent of the others. When it comes to which skills it involves, there is some level of dependency in each item. For example, there should never be an item with no skills. This can easily be learned when an algorithm has the full set of skills to consider, but it is not so when the item-skill mapping is given in isolation of the other skills involved for that item. While this example is over simplistic, it highlights the general idea. Therefore an analysis at the level of the item would yield more accurate results.

Figure 3.2 shows the general refinement process. First, and for comparison purpose, we perform refinement with the three data driven methods, MinRss, MaxDiff, ALSC that were investigated in [40]. Then, after a data transformation, we classify the likely refinements. Details are described below.

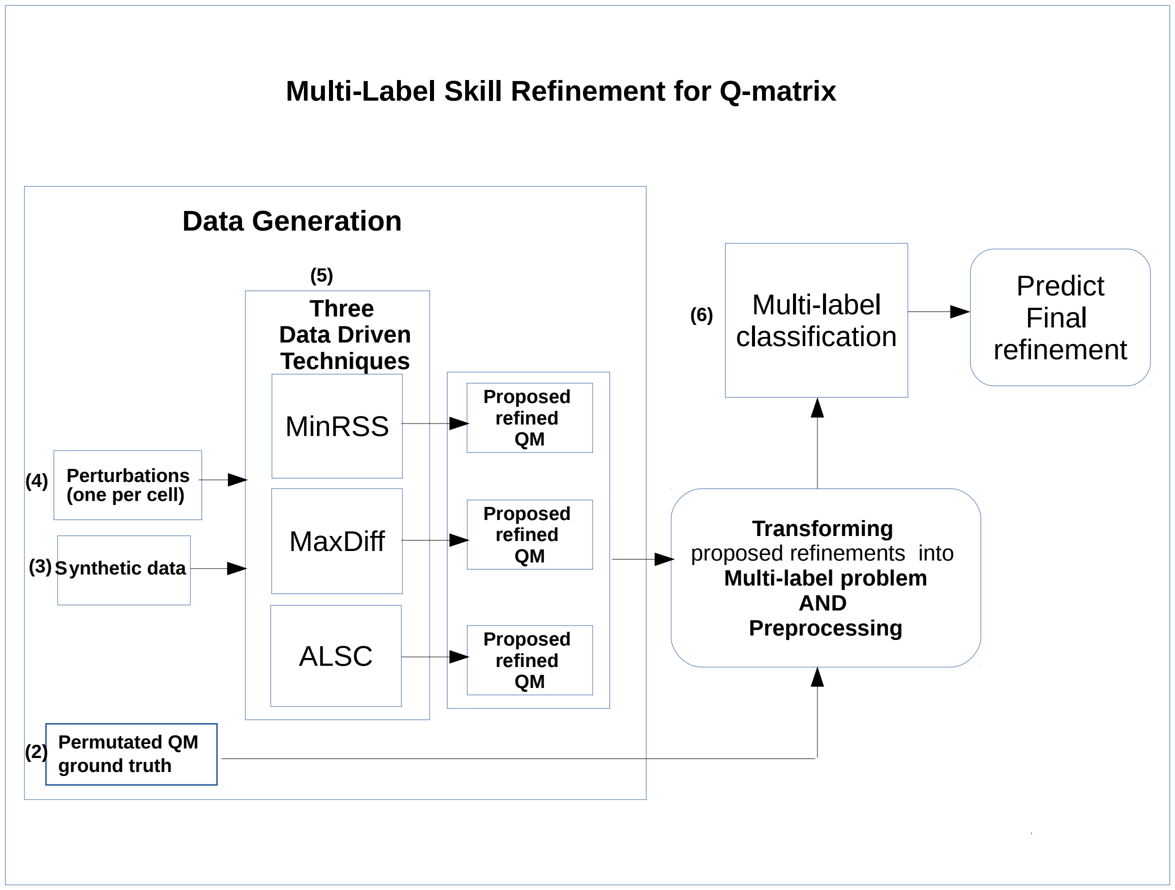

Multi-Label Skill Refinement for Q-matrix
Data Generation
(5)
Three Data Driven Techniques
MinRSS
MaxDiff
ALSC
Proposed refined QM
Proposed refined QM
Proposed refined QM
(4) Perturbations (one per cell)
(3) Synthetic data
(2) Permutated QM ground truth
(6) Multi-label classification
Predict Final refinement
Transforming proposed refinements into Multi-label problem AND Preprocessing

Figure 3.2 Refinement Procedure of each Q-Matrix QM_i

3.1.2 Data transformation and filtering

After applying the three refinement methods over a Q-matrix, we get three proposed Q-matrices. They are transformed to a format suitable to multi-label classification as shown in Table 3.1. Each line is a record for a single item to skills mapping. The left side represents proposed q-vectors from three data driven techniques. They serve as features in classification. The right column contains the true q-vectors. A comparison of the input data of the ensemble algorithms that rely on cell-wise data and our method which relies on item-wise data is shown in figure 3.3.

Table 3.1 Example of the data used for the training of multi-label classification. 9 features of X come from refined q-vectors [3 skills:$S1, S2, S3$] of 3 q-matrix refinement methods: MinRSS, MaxDiff and ALSC (Left) and Real Values or Labels L represent the target q-vector from the ground truth q-matrix

Predicted Outputs skills S_n									Real Values (ground truth)		
MinRSS			MaxDiff			ALSC					
S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3
X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	L_1	L_2	L_3
1	1	0	1	1	0	1	1	0	1	1	0
0	1	0	0	1	1	0	1	1	0	1	1
1	1	1	1	0	1	1	1	1	1	0	1
1	0	0	1	0	0	1	1	1	1	0	0
1	0	1	1	0	1	1	0	1	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...

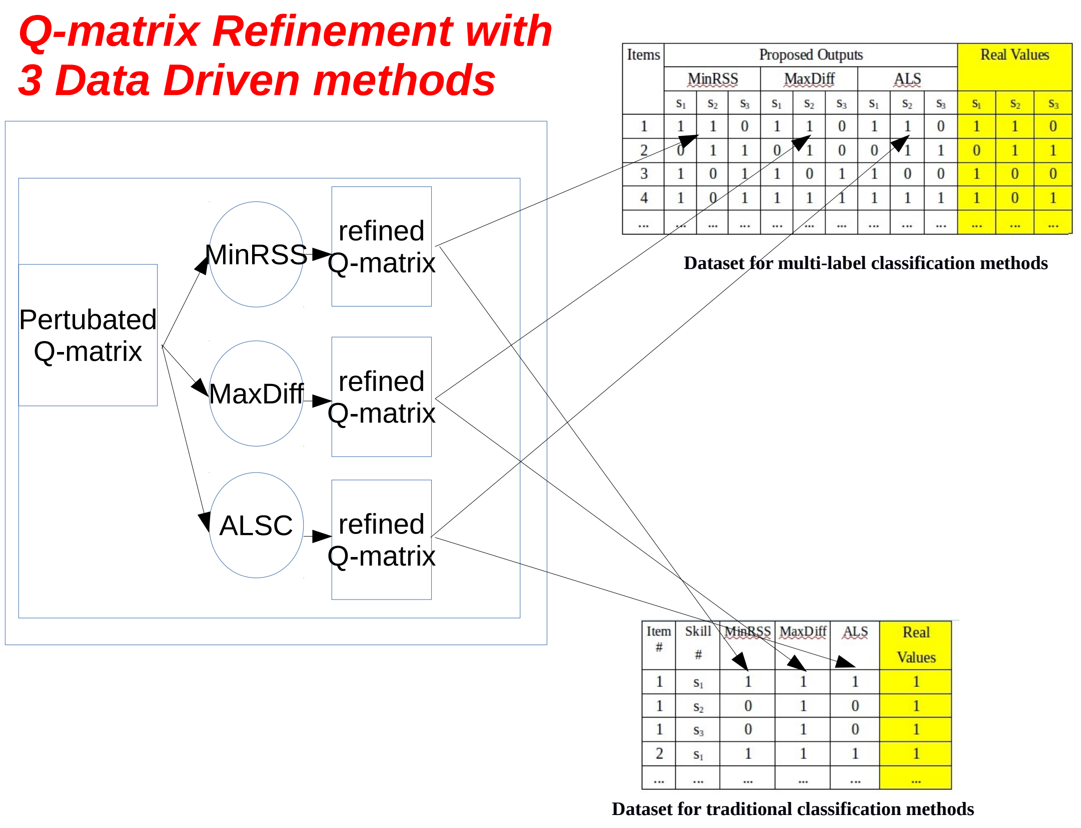
Filtering The filtering process is based on the occurences of identical q-vectors in table 3.1. Let us introduce some notation to explain this process. This an important step in the process to improve the performance and reduce the computational complexity at the same time.

Assume we have a set of skills, $\{S\}$. All possible q-vectors can be represented as a binary vector of length $|\{S\}|$ where $S_i = 1$ if the skill is present, and 0 otherwise. For example, the first line of table 3.1 is a repetition of the q-vector $(1, 1, 0)$, one for each method (MinRSS, MaxDiff, ALSC), and one for the real value.

Defining the set of q-vectors as capital letters class labels:

$$\mathcal{S} = \{\text{q-vectors}\} = \{A, B, C, ..\}$$

Then a line instance in table 3.1 will be considered *all-equal* if all of the individual q-vectors



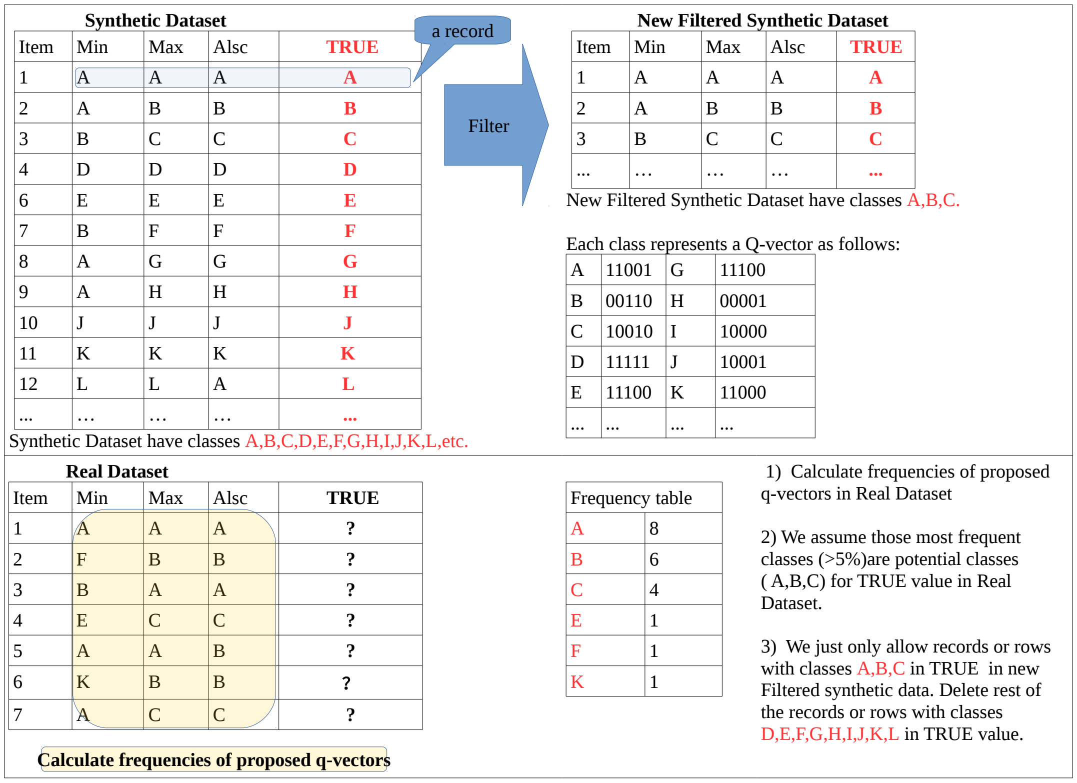
Items	Proposed Outputs									Real Values		
	MinRSS			MaxDiff			ALS					
	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3
1	1	1	0	1	1	0	1	1	0	1	1	0
2	0	1	1	0	1	0	0	1	1	0	1	1
3	1	0	1	1	0	1	1	0	0	1	0	0
4	1	0	1	1	1	1	1	1	1	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...

Dataset for multi-label classification methods

Item #	Skill #	MinRSS	MaxDiff	ALS	Real Values
1	s_1	1	1	1	1
1	s_2	0	1	0	1
1	s_3	0	1	0	1
2	s_1	1	1	1	1
...	...	...	...	...	...

Dataset for traditional classification methods

Figure 3.3 Comparison of datasets for traditional classification methods and multi-label classification methods



Synthetic Dataset

Item	Min	Max	Alsc	TRUE
1	A	A	A	**A**
2	A	B	B	**B**
3	B	C	C	**C**
4	D	D	D	**D**
6	E	E	E	**E**
7	B	F	F	**F**
8	A	G	G	**G**
9	A	H	H	**H**
10	J	J	J	**J**
11	K	K	K	**K**
12	L	L	A	**L**
...	…	…	…	**...**

Synthetic Dataset have classes A,B,C,D,E,F,G,H,I,J,K,L,etc.

New Filtered Synthetic Dataset

Item	Min	Max	Alsc	TRUE
1	A	A	A	**A**
2	A	B	B	**B**
3	B	C	C	**C**
...	…	…	…	**...**

New Filtered Synthetic Dataset have classes A,B,C.

Each class represents a Q-vector as follows:

A	11001	G	11100
B	00110	H	00001
C	10010	I	10000
D	11111	J	10001
E	11100	K	11000
...	...	...	...

Real Dataset

Item	Min	Max	Alsc	TRUE
1	A	A	A	**?**
2	F	B	B	**?**
3	B	A	A	**?**
4	E	C	C	**?**
5	A	A	B	**?**
6	K	B	B	**?**
7	A	C	C	**?**

Calculate frequencies of proposed q-vectors

Frequency table	
A	8
B	6
C	4
E	1
F	1
K	1

1) Calculate frequencies of proposed q-vectors in Real Dataset

2) We assume those most frequent classes (>5%)are potential classes (A,B,C) for TRUE value in Real Dataset.

3) We just only allow records or rows with classes A,B,C in TRUE in new Filtered synthetic data. Delete rest of the records or rows with classes D,E,F,G,H,I,J,K,L in TRUE value.

Figure 3.4 Example of Data Filtering Process

have the same class label for the three methods, and for the real value (ground truth). This is the case for the first line in table 3.1, but not for the second because MinRSS yields a different q-vector.

The filtering process is defined as yielding the subset $\mathcal{S}' \subseteq \mathcal{S}$:

$$\mathcal{S}' = \{\text{q-vector} \in \mathcal{S} \text{ and } \textit{all-equal}(\text{q-vector})) \text{ and } \text{freq(q-vector)} > n\}$$

where *all-equal*(q-vector) meets the *all-equal* requirement above and freq(q-vector) is the number of occurrences of *all-equal*(q-vector) in a synthetic data set. If that frequency is above a fixed threshold, n, it is kept in the filtering process, and rejected otherwise. n is set to the first 5% most frequent in our experiments.

Figure 3.4 schematically illustrates this data filtering process for a 5 skills QM. The filtered out classes will never be suggested as refinements for real Q-matrix in classification.

The training of the multi-label algorithm is based on $\mathcal{S}'$. Comparison of results are shown in Figure 3.11 and discussed later.

3.1.3 Proposed Multi-label Skills Refinement Algorithms

The multi-label algorithm is significantly more complex to solve than single-label (two-class or multi-class) problems. Only a few studies on multi-label learning are reported in the literature, mainly on text categorization, scene classification, and in bioinformatics.

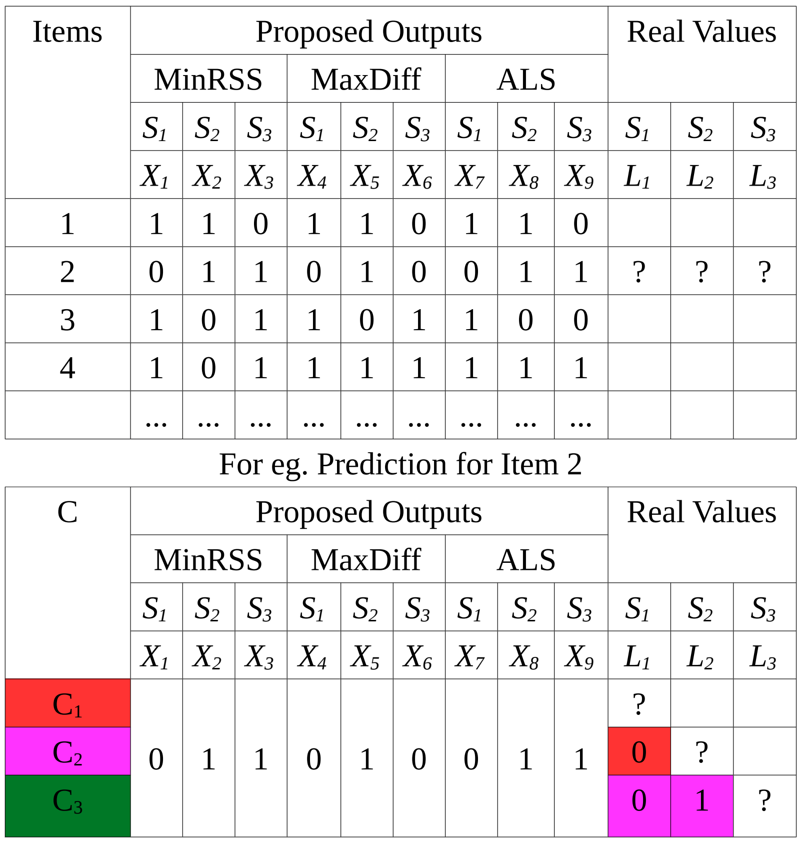
As mentioned, multi-label classification aims to predict a whole vector of labels at once, namely the item skills set in our case. In this study, we use two multi-label classification methods: binary relevance method (Classifier chain method) [72] by using Naive Bayes classifier, and RAndom k-labELsets(Ensemble method) [73] by using the J48 decision tree algorithm.

3.1.3.1 Binary Relevance method with Naive Bayes

The binary relevance method (BR) converts the multi-label problem into several binary classification problems [72]. This method involves Q binary classifiers linked along a chain.

Assume a set of classifiers, $\{C\}$ and a set of labels, $\{L\}$, which in our case correspond to the skills set. This method trains $|L|$ number of binary classifiers $C_1, ..., C_{|L|}$. Each classifier C_j is responsible for predicting the 0/1 association for each corresponding label $j \in L$. Any binary classifier can be used and in our experiments we chose Naive Bayes.

BR creates a linked chain of classifiers, each classifier taking as input features the output of the



Items	Proposed Outputs									Real Values		
	MinRSS			MaxDiff			ALS					
	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	L_1	L_2	L_3
1	1	1	0	1	1	0	1	1	0			
2	0	1	1	0	1	0	0	1	1	?	?	?
3	1	0	1	1	0	1	1	0	0			
4	1	0	1	1	1	1	1	1	1			
	...	...	...	...	...	...	...	...	...			

For eg. Prediction for Item 2

C	Proposed Outputs									Real Values		
	MinRSS			MaxDiff			ALS					
	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	L_1	L_2	L_3
C_1										?		
C_2	0	1	1	0	1	0	0	1	1	0	?	
C_3										0	1	?

Figure 3.5 An example of the classification process of Binary Relevance method

last classifier, in addition to the original feature set $\{X\}$, to predict the next label. In other words, the classifier C_i for label L_i in the chain considers the predicted values $l_1, l_2, ..., l_{i-1}$ from the previous classifiers as additional attributes. The feature set is thus extended with the class values (labels) of all previous classifiers in the chain. At classification time, the process starts at C_1, and chains the classifiers such that for C_i, it computes:

$$P(l_i) = \arg\max_{l_i} P(l_i|x, l_1, l_2, ..., l_{i-1}) \tag{3.1}$$

Figure 3.5 shows how the BR method works in its classification. Classifier C_1 is trained to predict L_1 with the feature set $\{X\}$. Classifier C_2 is trained to predict L_1 with the same feature set, plus the L_1 prediction which becomes a new feature for the next classification, and so on.

The chaining of classifiers has the advantage of allowing the classifiers to take into account interactions, or correlations between the features. The order of the chain is based on the same order given by the expert.

3.1.3.2 RAndom k-labELsets with J48

The second multi-label classifier proposed is RAndom k-labELsets (RAkEL).

The ensemble methods for multi-label learning are developed on top of the common problem transformation or algorithm adaptation methods. The most well-known problem transformation algorithm are RAkEL [73]. RAkEL constructs each base classifier by considering a small random subset of labels. It decomposes all labels L with size k into m random subsets of labels set and trains a label power-set (LP) classifier using each set of labels. Let the term L^k denote the set of all distinct k-labelsets of L. The size of L^k is given by the binomial coefficient:$|L^k| = \binom{i}{k}$.

Given a size of labelsets k over set of i labels and a number of desired classifiers $m \leq |L^k|$, RAkEL initially selects m k-labelsets $R_j, j = 1...m$ from the set L^k via random sampling without replacement. Then RAkEL learns multi-label classifiers $C_j, j = 1...m$ using LP. For the prediction, each classifier provides binary predictions for each label set. Subsequently, It determines the final set of labels for a given example by using a simple voting process. It calculates the mean of these predictions for labels from each classifier C and outputs a final positive decision if it is greater than a 0.5 threshold. This intuitive threshold corresponds to the majority voting rule for the fusion of classifier decisions. It has been used for deriving a final decision in the problem transformation methods.

Items	Proposed Outputs									Real Values		
	MinRSS			MaxDiff			ALS					
	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	L_1	L_2	L_3
1	1	1	0	1	1	0	1	1	0			
2	0	1	1	0	1	0	0	1	1	?	?	?
3	1	0	1	1	0	1	1	0	0			
	...	...	...	...	...	...	...	...	...			

For e.g. Prediction for Item 2

C Subset size=2	Proposed Outputs									Real Values		
	MinRSS			MaxDiff			ALS					
	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3	S_1	S_2	S_3
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	L_1	L_2	L_3
$C_1\{L_1,L_2\}$										0	1	
$C_2\{L_2,L_3\}$	0	1	1	0	1	0	0	1	1		1	1
$C_3\{L_1,L_3\}$										0		1
Voting										0	1	1

Figure 3.6 An example of the classification process of Random k-Labelsets method

The classification process of RAkEL is given in Figure 3.6, with user-specified parameters: k=2 (size of labelsets). In this way, the proposed algorithm aims to take into account label correlations using single-label classifiers that are applied on sub tasks with a manageable number of labels and adequate number of records per label. In this experiment we use the J48 classifier, an optimized implementation of the C4.5 or improved version of the C4.5. J48 constructs a decision tree as an output.

In Figure 3.6, all possible combination of labels are made over three labels $\{L_1, L_2, L_3\} \in L$. Those subsets are named as $\{S_1, S_2\}$,$\{S_2, S_3\}$ and $\{S_1, S_3\}$ in our case. Then it builds the classifiers $\{C_1, C_2, C_3\}$ over these three subsets. Each classifier predicts the labels in each subsets independently. After getting the results from each classifier, a simple voting process is performed according to the votes or results form classifiers.

3.2 Datasets

3.2.1 Real data

For the sake of comparison, we use the dataset from Tatsuoka [12] that was also used in [40]. It is a well-known data set in fraction algebra and consists of 3 expert-driven Q-matrices and one data driven Q-matrix with an SVD approach. They allow us to analyze different Q-matrices over the same performance data. Table 3.2 provides the basic information and source of each dataset and Table 3.3 contains the details of each Q-matrix.

All Q-matrices contain 11 items that are common to all Q-matrices. The same performance data of 536 respondants can therefore be used for refining every one of them.

Table 3.2 Q-matrix for validation & explanation of category

Q-Matrices	Number of			Description
	Skills	Items	Cases	
QM1	3	11	536	Expert driven from [13]
QM2	5	11	536	Expert driven from [8]
QM3	3	11	536	Expert driven from [74]
QM4	3	11	536	Data driven, SVD based

It can be seen that each item in QM1 is a binary unit vector (“one-hot”), implying that skills are exclusive. This is typical of cases where items are broken down in single topics. QM3 also has the particularity of having the first skill required by all items. This skill can represent a general ability that spans across specific skills and is more representative of a compensatory type of Q-matrix, even though it is formally a conjunctive Q-matrix. We would expect methods such as matrix factorization to perform better on such matrices because the underlying semantics is that skills can take real values that are additively combined to compute the chances of success.

Table 3.3 Q-matrices 1, 2, 3, and 4

	skills of													
	QM 1			QM 2					QM 3			QM 4		
Item	1	2	3	1	2	3	4	5	1	2	3	1	2	3
1	0	1	0	1	1	1	1	0	1	1	0	1	1	0
2	0	0	1	1	1	1	1	1	1	0	1	1	0	1
3	0	0	1	0	0	1	0	0	1	0	1	0	1	0
4	1	0	0	1	1	1	1	0	1	0	0	1	0	0
5	0	1	0	1	1	1	1	0	1	1	0	1	0	0
6	0	1	0	1	1	0	0	0	1	1	0	0	0	1
7	0	0	1	1	0	1	1	1	1	0	1	1	0	1
8	0	0	1	1	0	1	0	0	1	0	1	0	1	1
9	1	0	0	1	0	1	1	0	1	0	0	1	0	0
10	1	0	0	1	1	1	1	0	1	0	0	1	0	1
11	0	1	0	1	1	1	1	0	1	1	0	1	0	0

We refined real Q-matrix with $[11 \times 3]$ or $[11 \times 5]$ dimension, that is, 11 items and 3 skills or 5 skills respectively, then transform outputs into datasets for multi-label classification with following steps:

1. For each Q-matrix of $[11 \times 3]$ with one cell perturbation, we have 33 pertubated Q-matrices and for each Q-matrix of $[11 \times 5]$ with one cell perturbation, we have 55 pertubated Q-matrices .

2. Feed pertubated Q-matrices and response matrices to MinRSS, MaxDiff and ALSC method and then produce refined Q-matrices from each methods.

3. Finally, transform refined Q-matrices from three data driven methods into different formats of datasets for multi-label classification as in Figure 3.3

3.2.2 Synthetic data

We use synthetic data generated from 1000 permuted matrices for training and use real data for testing. The procedure for data generation of training and testing is shown in Figure 3.7. The general idea is to introduce a perturbation in a Q-matrix and to run the refinement algorithms on the perturbated matrix to validate whether the perturbation is identified and false alarms are introduced by multi-label classifier.

Given the Q-matrices, we generate the synthetic data that provides the ground truth to do the training. Pertubation are introduced to make the data closer to real data and we use the original ratio of 0/1 in the perturbated matrix to create the 1000 permutations. In this experiment, we explored only a single cell perturbations for training model and test with both synthetic and real data (where real data was perturbated ranging from 1 to 10 cells). Next, we follow the approach in Figure 3.2, multi-label classification algorithms are used for predicting all skills of an item at once.

Synthetic data (outputs from refinement on permutated Q-matrices with three data driven methods) is used for training of learning the reasoning rules of classification in different simulated scenarios and test with both synthetic and real data. [40].

The synthetic data generation process initially starts from Q-matrix with $[11 \times 3]$ or $[11 \times 5]$ dimension, that is, 11 items and 3 skills or 5 skills respectively, then applies following steps:

1. Randomly permute the Q-matrix and iterate up to 1000 times for obtaining 1000 new Q-matrices as in Figure 3.7.
2. For each new Q-matrix, produce a response matrix with 400 respondents by using DINA model with $slip = guess = 0.2$ and uniform profile matrix .
3. For each Q-matrix of $[11 \times 3]$ with one cell perturbation, we have 33 pertubated Q-matrices and for each Q-matrix of $[11 \times 5]$ with one cell perturbation, we have 55 pertubated Q-matrices .
4. Feed pertubated Q-matrices and response matrices to MinRSS, MaxDiff and ALSC method and then produce refined Q-matrices from each methods.
5. Finally, transform refined Q-matrices from three data driven methods into different formats of datasets for multi-label classification as in Figure 3.3

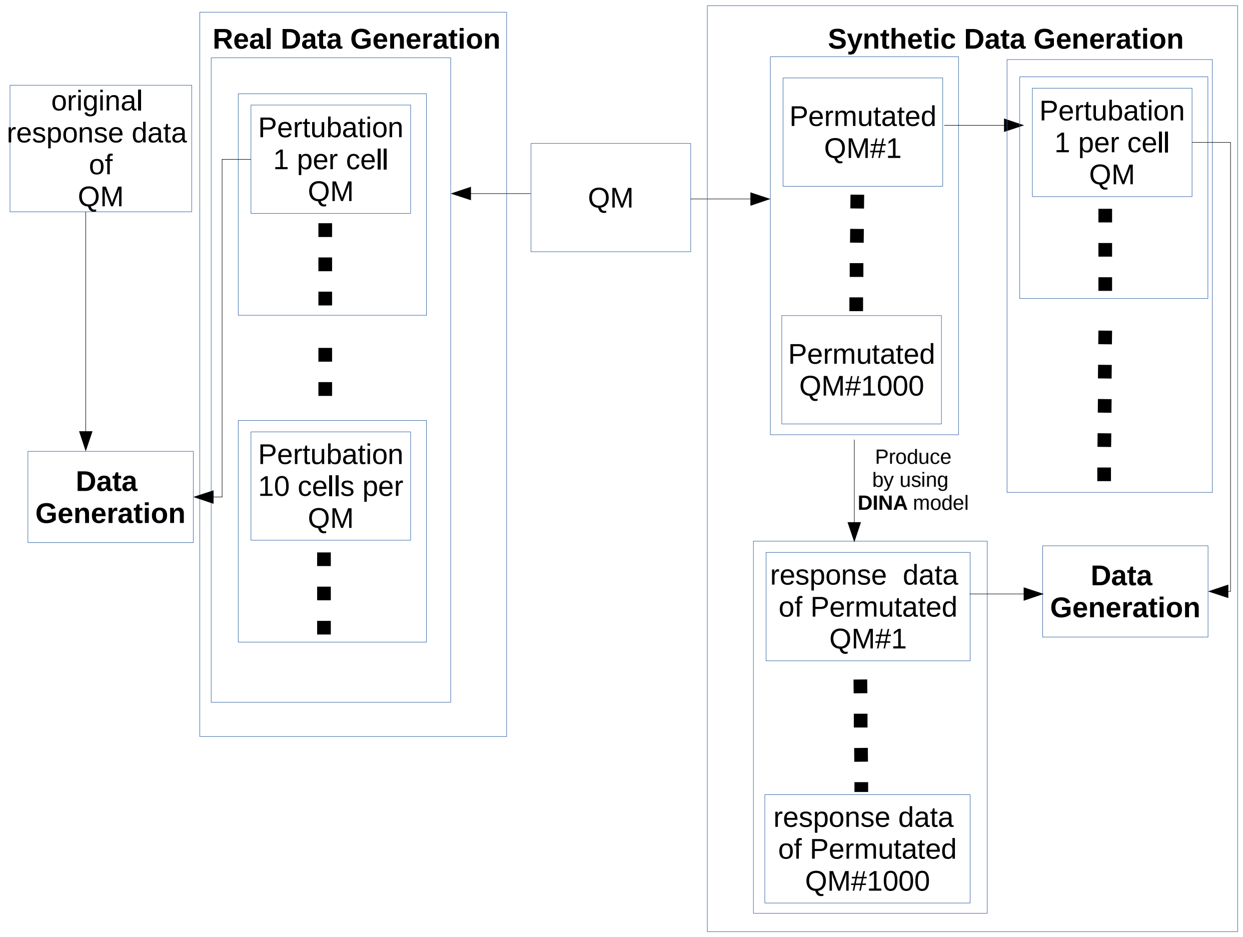

Real Data Generation
Synthetic Data Generation
original response data of QM
Data Generation
Pertubation 1 per cell QM
Pertubation 10 cells per QM
QM
Permutated QM#1
Permutated QM#1000
Pertubation 1 per cell QM
Produce by using DINA model
response data of Permutated QM#1
response data of Permutated QM#1000
Data Generation

Figure 3.7 Data generation procedure of synthetic and real data from each Q-Matrix QM_i

3.3 Error metrics

Multi-label classifiers output, for each data record, a set of class labels that aim to correspond to the true set of labels. In the context of Q-matrices, records are q-vectors (items or lines in the Q-matrix) and labels are skills. However, previous Q-matrix refinement techniques were assessed at individual cell levels. We will use metrics at both item and cell levels to account for the different evaluation perspectives.

- Hamming Loss: a standard measure for multi-label classification.
- Cell Wise Accuracy: errors at the individual cell level.
- Vector-Wise Accuracy: errors at the q-vector level.

Let us first make a few definitions to describe the metrics in more details.

Let D be the class-labels domain of a multi-label evaluation data set. In the context of this experiment, labels are skills. Thus, for QM2, $D_{qm_2} = \{s_1, s_2, s_3, s_4, s_5\}$, and $D = \{s_1, s_2, s_3\}$ for all other QMs. Let $L_i \subseteq D$ correspond to the skills required by item i (L_i is a q-vector, or a line in the Q-matrix). For example, if item 1 of QM2 has the q-vector $(0, 1, 0, 1, 0)$, then the corresponding L_i is $L_1 = \{s_2, s_4\}$.

A multi-label classifier's task is to predict, for each q-vector, the subset $Z_i \subseteq D$ that corresponds to the true labels L_i. Let H be a multi-label classifier and $Z_i = H(\text{QM}, i)$ be the set of labels predicted for item i of Q-matrix QM.

Given the above definitions, the metrics for evaluating the predictions are defined below.

Hamming Loss: a measure of how many times an instance label set is misclassified, i.e. a label not belonging to the instance is predicted or a label belonging to the instance is not predicted. The performance is perfect when $HammingLoss = 0$; the smaller the value of $HammingLoss$, the better the performance:

$$HammingLoss = \frac{1}{m} \sum_{i=1}^{m} \frac{|Z_i \Delta L_i|}{|D|} \tag{3.2}$$

where m is the number of rows of the Q-matrix in the evaluation, and Δ stands for the symmetric difference between two label sets.

Cell Wise Accuracy: Cell wise accuracy is calculated based on the average differences of the actual and the predicted labels over the evaluation data set. The performance is perfect when *CW-Accuracy* $= 1$; the bigger the value,the better the performance:

$$CW\text{-}Accuracy = \frac{1}{m}\sum_{i=1}^{m}\frac{2|L_i \cap Z_i|}{|Z_i| + |L_i|} \tag{3.3}$$

Vector-Wise Accuracy: Vector-wise accuracy is based on whole item vector prediction. Any cell wise difference in an item vector is considered a loss over the full vector. *VW-Accuracy* is defined as follows:

$$VW\text{-}Accuracy = \frac{1}{m}\sum_{i=1}^{m} I(Z_i = L_i) \tag{3.4}$$

where I is identicality of the vectors

3.4 Empirical Evaluation

We compare our proposed methods with three data driven techniques [7, 8, 9, 10], and ensemble techniques applied with traditional classifiers (Classification Tree, Boosted Classification Tree, Regression Tree, Boosted Regression Tree) proposed in [40] for both synthetic and real data. Our experiment relies on the CDM [74] and NPCD package for R, which provided both the code for three basic data driven techniques and the data, `rpart` packages for traditional classification and `mulan` [75] for multi-label classification.

The experimental results are reported in Tables 3.4, 3.5, and 3.6 for synthetic data, and in Figures 3.8, 3.9, and 3.10 for real data.

A **logit scale** is used when measures are on the $[0, 1]$ scale and the results are close to 1. This measures allows better comparison of values, in particular for the graphical representations, where, for example, a difference between 0.990 and 0.995 is often difficult to discriminate due to the graphic's scale, yet it can represent a 50% reduction in error. The logit can be considered a good estimate of the relative remaining error on a scale of $[0, 1]$ (for e.g., it displays a relative error reduction in vector-wise accuracy from 0.90 to 0.95 as similar to the reduction from 0.99 to 0.995).

We compare our two multi-label based approaches (BR and RAkEL) with three well-known data driven methods, four ensemble methods based on traditional classification (CT, BCT, RT and BRT) with both real and synthetic data. For all methods based on traditional classification, we applied default parameters and experimental settings used in original papers [40, 76]. We also report two baselines: majority vote and random guessing for cell-wise accuracy.

All experiments were done with 10 fold cross validation.

3.4.1 Result for synthetic data

Table 3.4 Logit value of Hamming Loss result of Synthetic data (single perturbation)

	Individual			Ensemble					
				Non multi-label				Multi-label	
QM	MinRSS	MaxDiff	ALSC	CT	BCT	RT	BRT	BR	RAkEL
qm1	-4.74	-3.95	-4.59	-Inf	-Inf	-Inf	-5.2	-5.47	**-5.99**
qm2	-3.88	-3.26	-2.73	-4.41	-4.41	-4.41	-3.72	-4.45	**-5.01**
qm3	-5.44	-4.05	-5.25	-5.88	-6.07	-5.88	-5.95	-6.16	**-8.52**
qm4	-5.08	-3.89	-4.91	-5.18	-5.31	-5.18	-4.98	-5.08	**-5.77**

Table 3.5 Logit value of vector-wise accuracy result of Synthetic data (single perturbation)

	Individual			Ensemble					
				Non multi-label				Multi-label	
QM	MinRSS	MaxDiff	ALSC	CT	BCT	RT	BRT	BR	RAkEL
qm1	3.76	2.88	3.49	Inf	Inf	Inf	4.09	4.39	**4.9**
qm2	2.19	1.58	0.97	2.75	2.75	2.75	2.04	2.81	**3.5**
qm3	4.59	3.08	4.39	4.97	5.11	4.97	4.97	5.49	**8.11**
qm4	4.17	2.81	3.79	4.13	4.21	4.13	3.88	4.01	**4.8**

Table 3.6 Logit value of cell-wise accuracy result of Synthetic data (single perturbation)

	Individual			Ensemble					
				Non multi-label				Multi-label	
QM	MinRSS	MaxDiff	ALSC	CT	BCT	RT	BRT	BR	RAkEL
qm1	4.71	4.12	4.93	Inf	Inf	Inf	5.47	5.65	**6.21**
qm2	4.18	3.70	3.21	4.62	4.62	4.62	3.77	4.8	**5.31**
qm3	5.31	4.03	5.13	5.57	5.84	5.57	5.95	5.95	**8.11**
qm4	4.94	4.00	4.95	4.93	5.08	4.93	5.05	5.08	**5.74**

For synthetic data, a single cell is perturbed. We can see from Tables 3.4, 3.5 and 3.6, that most of multi-label skill refinement methods can recover at least 99% for all Q-matrices. The performance reaches almost 100% in terms of vector-wise accuracy and cell-wise accuracy. The standard deviations of all values are less than 0.01.

Clearly, all methods using multi-label refinement algorithms perform much better than any single method and the results are also as good as or better than those of the single-cell ensemble technique based on the decision tree method reported in [40] and [76].

Table 3.7 Logit value of cell-wise accuracy result of synthetic data (single perturbation)

QM	Majority Vote	Random Guessing	RAkEL
qm1	3.44	2.76	**6.21**
qm2	3.94	3.28	**5.31**
qm3	3.44	2.76	**8.11**
qm4	3.44	2.76	**5.74**

The comparison with the majority vote and random guessing shows that all methods are substantially better.

3.4.2 Result for real data

For real data, multiple perturbations are introduced and the results are shown in Figures 3.8, 3.9 and 3.10. They show the trends as a function of the number of perturbations. The black lines show the results of the three individual refinement algorithms, and the colored lines show the ensemble techniques refinement results.

As expected, the performance of all methods decline with the number of perturbations. However, and contrary to synthetic data, the multi-label methods do not systematically outperform the other methods in QM3 and QM4 because QM3. In QM3, we remove skill 1 because all values in skill 1 is 1 and it is not possible to train a binary classifier for a label with all values are 1. Besides, QM4 is SVD driven Q-matrix instead of expert driven like three other q-matrices.

For QM1 and QM2, the proposed multi-label methods show better performance in general than three data driven methods, and the trend remains same over number of perturbation. However, this is not the case for QM3 and QM4.

For QM4, MaxDiff is the best at one cell perturbation among all other methods. QM4 is the only data driven QM in all tested real Q-matices, which is driven from SVD. Besides, MaxDiff is a method which performs the least number of changes among all methods. Those reasons might explain why MaxDiff does better when few perturbations are introduced, and why this advantage disappears with more perturbations.

RAkEL shows the best performances for expert driven Q-matrices in general, but not that significant for SVD driven Q-matrix. In QM2, RAkEL outperforms 67% better than MaxDiff (best among three data driven algorithms) and 19% better than BRT (best among non multi-label techniques, except BR) in terms of vector-wise accuracy when it reaches to 10 cells perturbation. However, the results for QM3 and QM4 show that the MaxDiff method has better performance in vector-wise accuracy than proposed methods up to 2 or 3 perturbations.

Note that all results are based on the filtering process described in section 3.1.2. This process improves substantially the results as can be seen in the example accuracy difference reported in figure 3.11.

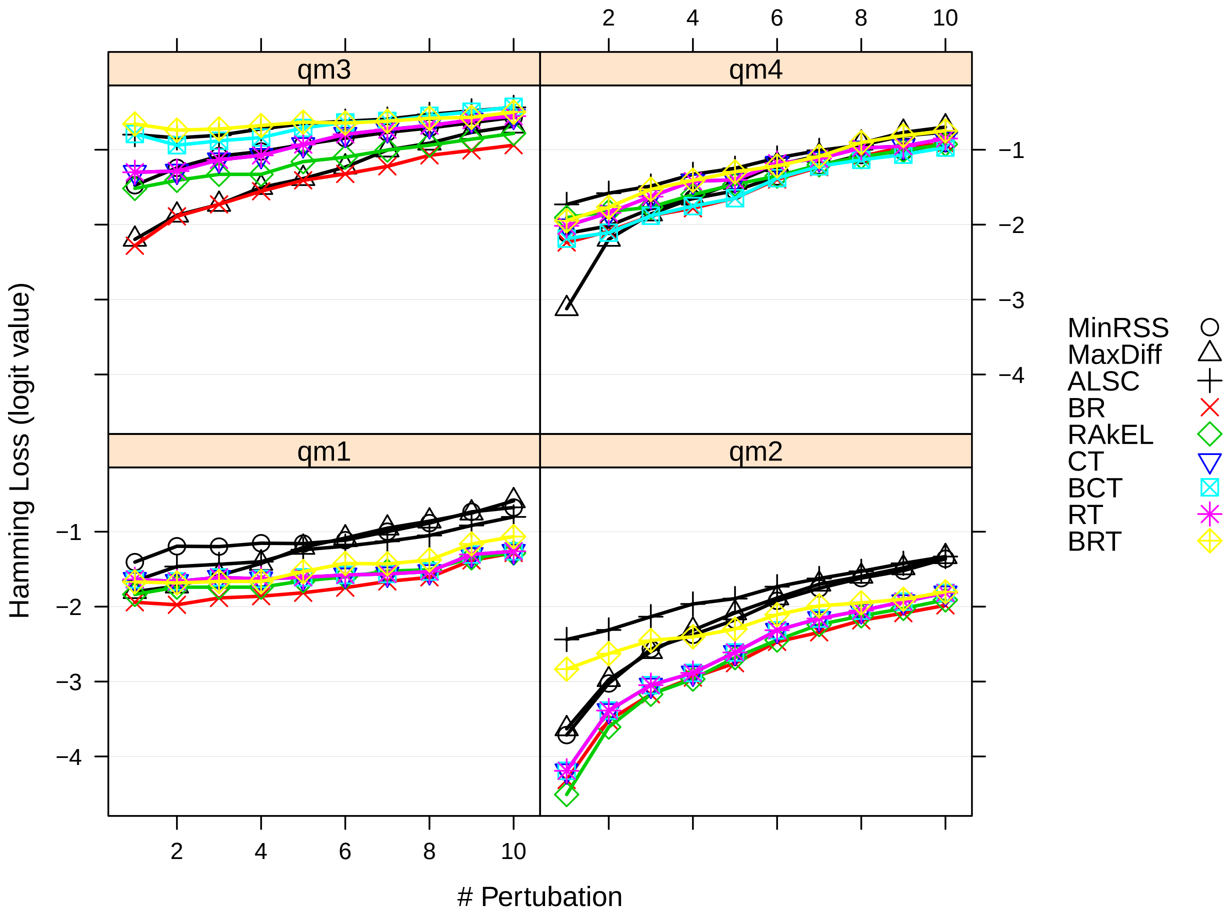


Figure 3.8 Real data: Logit value of hamming loss as a function of the number of perturbations. Black lines are the original single methods. Colored lines are ensemble techniques.

3.4.3 Discussion

In this research, we present a Q-matrix refinement method, called multi-label skill refinement method, that combines three data driven techniques and two multi-label classification techniques to assess the skills required for tasks of students. The driving intuition behind the approach is that instead of using an ensemble approach that works at the level of each

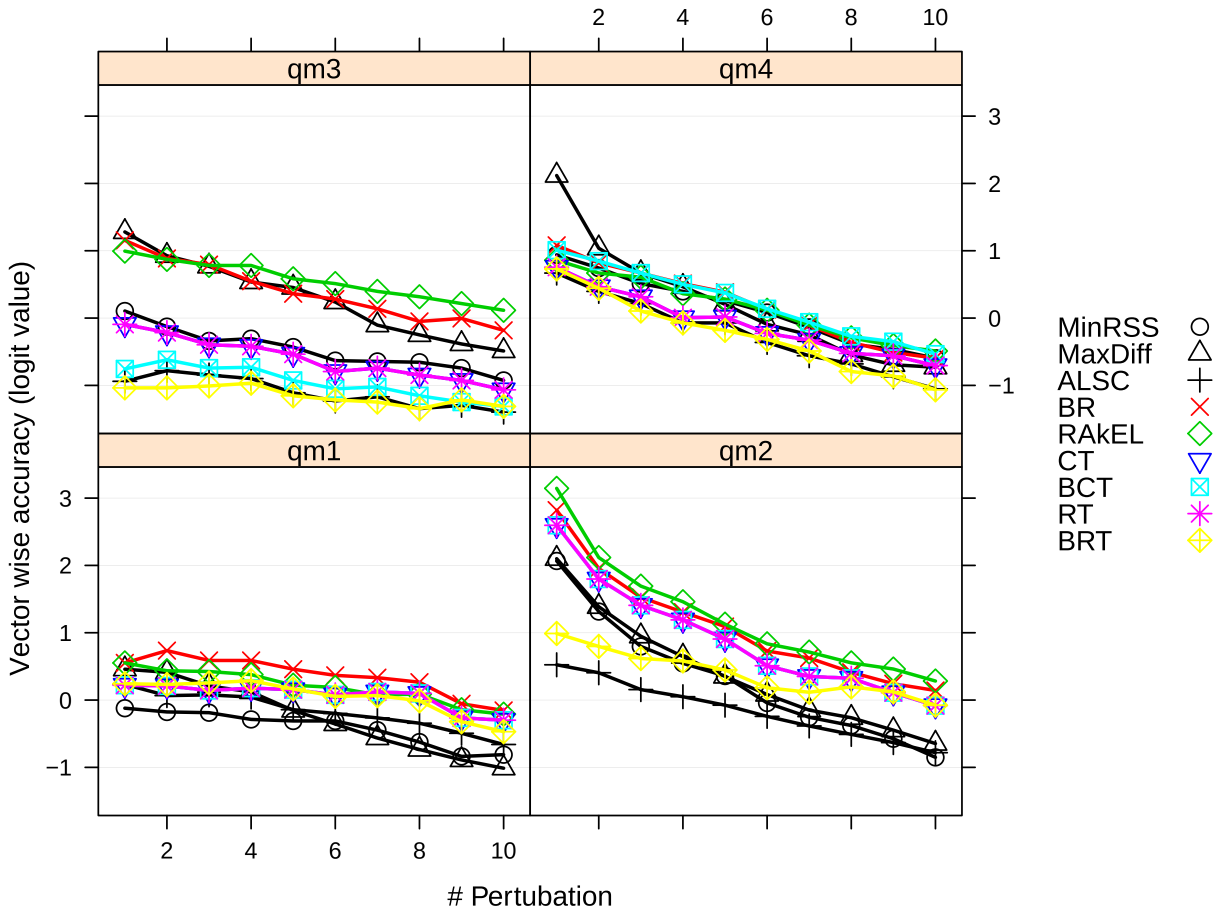


Figure 3.9 Real data: logit value of vector-wise accuracy as a function of the number of perturbations.

individual cell of the Q-matrix, the chosen perspective is to refine the skills mapping at the task/item level.

Results show that this approach improves the performance in general. Experiment with three expert-driven Q-matrices and 1 Q-matrix driven from SVD, shows the proposed approach outperforms three state-of-the-art algorithms for synthetic data. For real data, the results are not as sharp as the synthetic data and, for the data-driven Q-matrix, they are actually worst for the single perturbation case and the MaxDiff. This might be explained by the fact that MaxDiff is very conservative and tends to avoid making any changes to the given Q-matrix. As more perturbations are introduced, this advantage becomes less important.

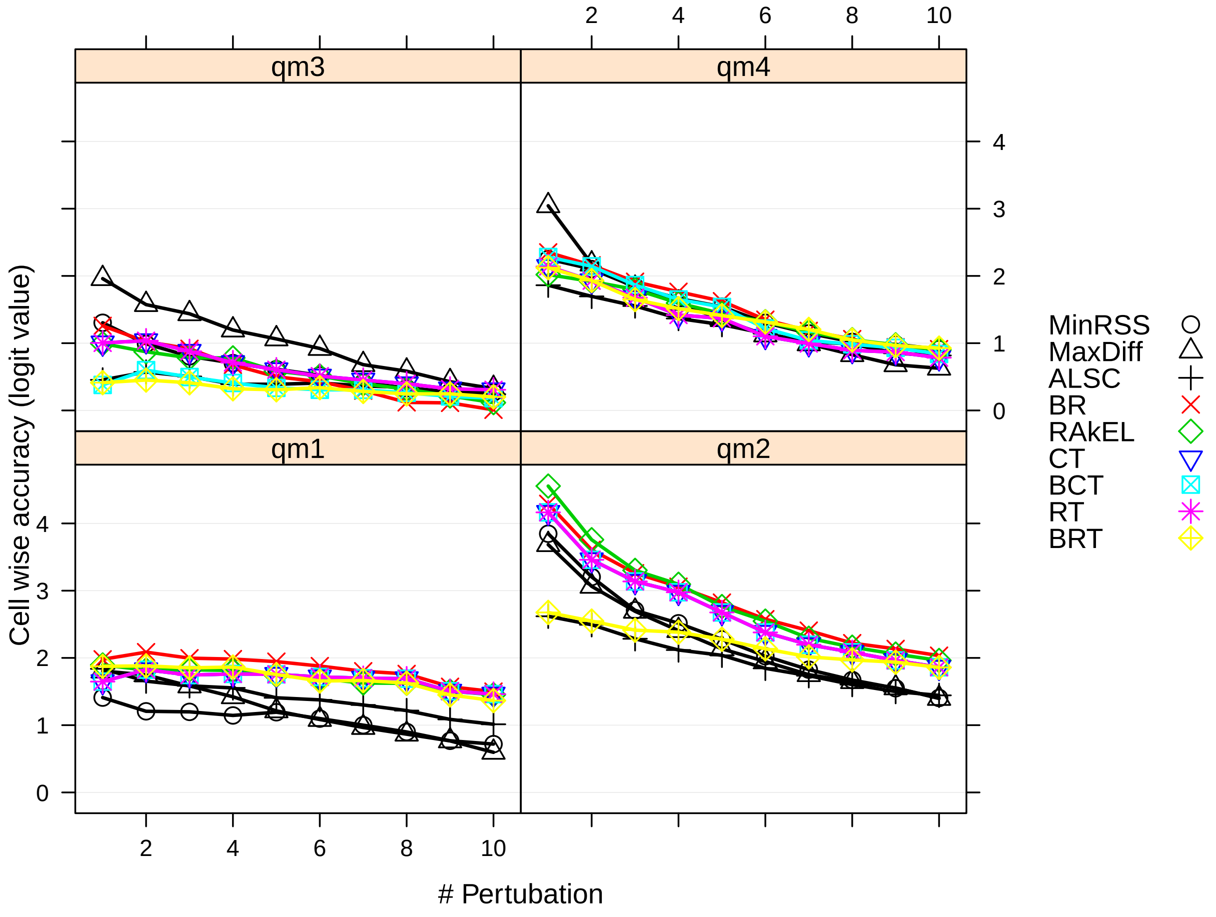


Figure 3.10 Real data: logit value of cell-wise accuracy as a function of the number of perturbations.

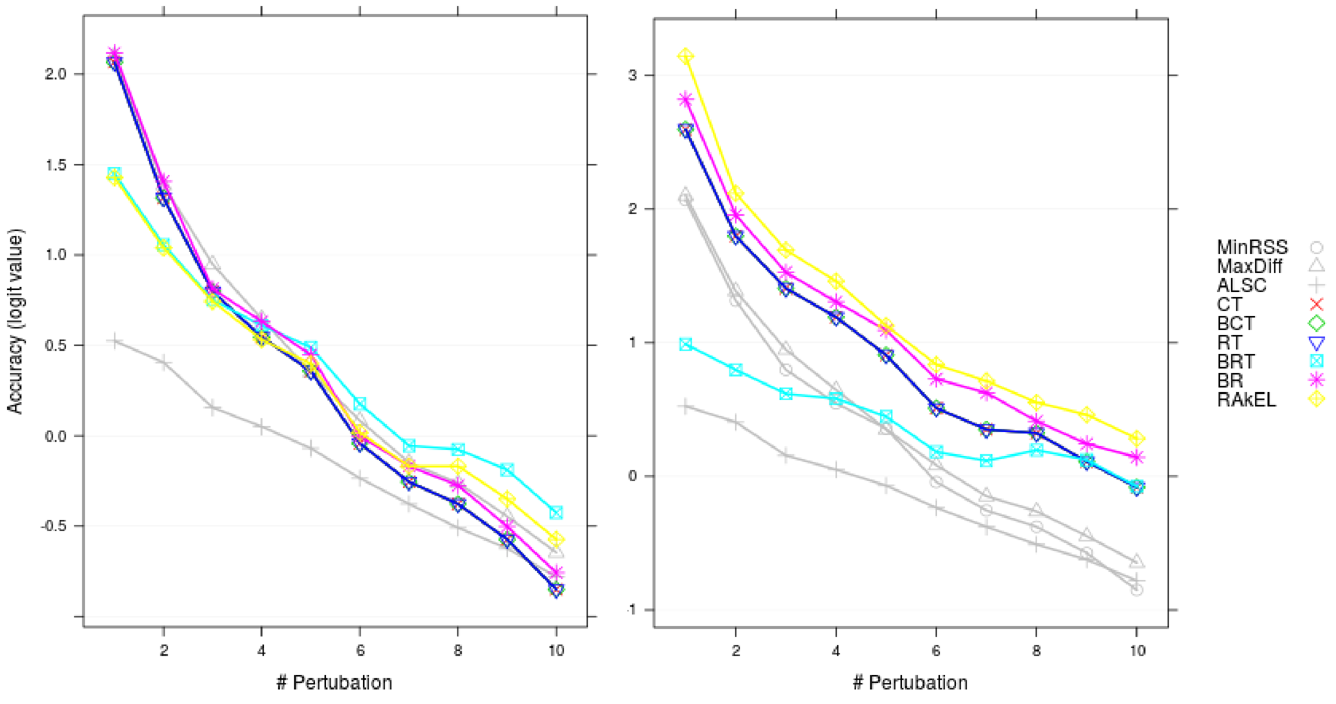


Figure 3.11 Real data QM2: Logit value of vector-wise accuracy with (right) and without (left) data filtering.

CHAPTER 4 KNOWLEDGE TRACING AND LEARNER ABILITY PROFILE WITH AN LSTM ARCHITECTURE

In this chapter, we show how we can leverage a deep-learning architecture, LSTM, to include a temporal ability profile and realize a form of transfer learning. LSTM, or Long Short-Term Memory, is a highly popular architecture that can model sequential data and retain previous information to better predict the next item in the sequence. It was first used for KT by Piech [23] and we extend it to model a student temporal ability profile that represents a more general skill set that spans across individual skills. The temporal ability profile is meant to serve as a learning transfer mechanisms across skills.

The approach is named DKT-DSC and was published at IEEE ICDM (International Conference on Data Mining) [77]. We describe the method, data, setup, analysis, and results of a performance assessment of our novel model.

4.1 Learning transfer and temporal ability profile

A strong limitation of KT models such as BKT is that skills are independent. When a student is tried on a new skill, the learning she has gained on previous skills is not considered relevant to the new skill tried. This is known as the *learning transfer* mechanism that is lacking in BKT and in most KT models.

Intuitively, learning transfer across skills is a naturally occurring phenomena. Most people would agree with the observation that general domain knowledge accumulates through practice and allows the student to solve new problems in the same domain with greater success, even if that problem involves new and specific skills.

Learning transfer implies that students are able to transfer their acquired skills to new situations and across problems involving different skills set. Models such as Learning Factor Analysis (LFA—[56]) and Performance Factor Analysis (PFA—[57]) aim to capture this learning transfer phenomena by including a factor that represents the learning accumulated on all KCs through practice and include this factor as a predictor of success in further practice. These models have been shown to outperform the standard BKT model in general, and have given rise to further research on the subject [78, 79, 80].

One of the recent contributions in that direction is the work of Nedungadi [81] who propose enhancements to the BKT model (that learns over each skill independently) by clustering students into high, medium and low knowledge. Our work is similar to this approach, as we also use a clustering approach to create learner profiles, and integrate this enhancement into a neural approach, the DKT model (that learns across all skills simultaneously). While [23] argued that DKT has the ability to model some of the learning transfer occuring across skills, our proposed approach includes this student profile feature explicitly with the aim that it will improve the modeling of learning transfer and that it will get reflected in better performance prediction.

4.2 Long Short-Term Memory (LSTM)

Akin to DKT, the proposed approach relies on the LSTM architecture. We describe LSTM in this chapter along with the standard DKT. We later describe the student profile modeling proposed.

Deep learning methods are capable of recognizing non linearity and complex patterns of data in time series forecasting. As described in Chapter 2, Neural Networks (NNs) are a highly popular and useful approach to machine learning. However, a serious limitation of feed-forward neural network is that each prediction is treated independently from the previous. Thus, there is no concept of maintaining a current state that can serve as a memory of past inputs. Recurrent Neural Networks (RNNs) address this issue by basing each prediction not only on its current input but also on the hidden/latent state from the previous prediction, as depicted in Figure 4.1. Long-short Term Memory Networks (LSTMs) is a variant of RNNs that addresses this issue through gate units, effectively allowing the model to exercise forgetting or remembering on certain elements of the information as it is sequentially learned through the network. LSTM was the de facto standard for processing sequential problem.

The Long Short-Term Memory (LSTM) architecture was originally proposed by Hochreiter [82], later refined by many researchers and applied in a large variety of problems. The term “Long Short-Term Memory” comes from the intuition of recurrent neural networks with weights for remembering long-term memory, those weights are changing slowly during training for encoding knowledge about the complex data across time. Short-term memory existed in the form of ephemeral activations, by passing from each node to subsequent nodes [83]. LSTM has been used in many application domains such as natural language processing, handwriting recognition, speech recognition, time-series prediction [84, 85].

Hochreiter[82] introduced the original LSTM model in order to overcome the problem of vanishing gradients. A standard recurrent neural network with a hidden layer is resembled as LSTM. Instead of using ordinary nodes of the hidden layer, the LSTM has memory cells. LSTM contains a set of memory cells where the data is captured and stored. A composite

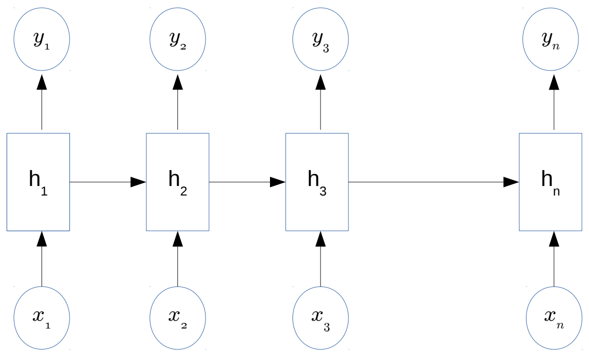


Figure 4.1 Architecture of an “unfolded”, or “unrollded” Recurrent Neural Networks (RNNs), where h_n represents the hidden state at time n.

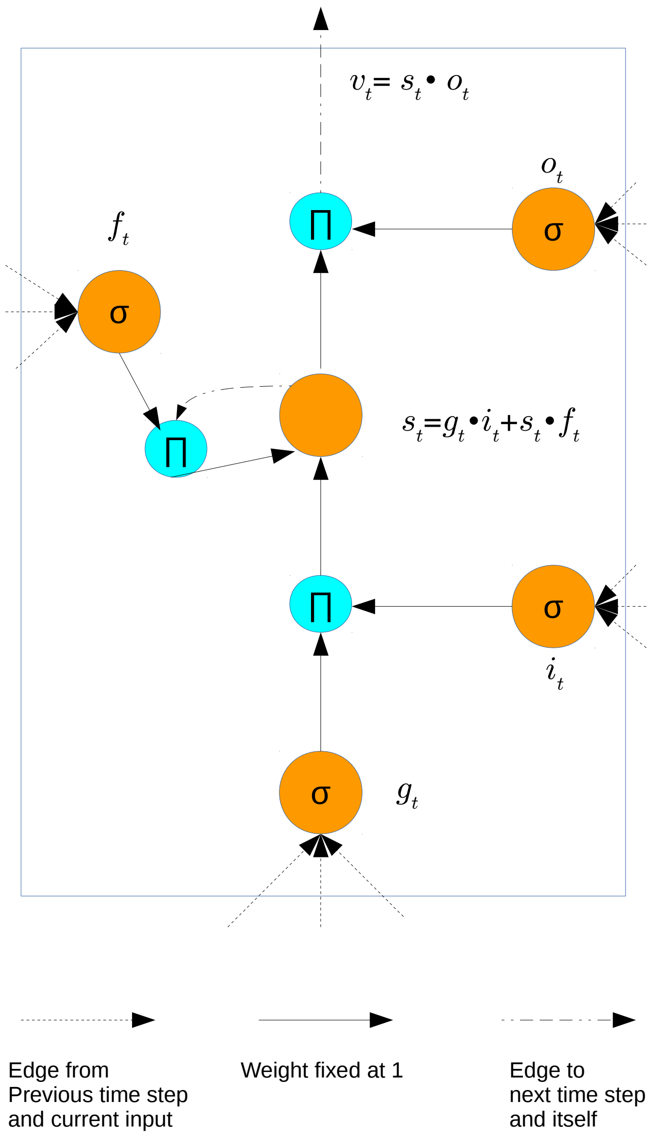
unit built with simple nodes with a particular connectivity pattern and the multiplicative nodes is called a memory cell [83].

These memory cells resemble a transport line that connects one to another by conveying data from past and gathering them for the present one. A self-connected recurrent node with fixed weight ensures to pass the gradient across many time steps without vanishing or exploding in each memory cell. The edges that connect adjacent time steps are called recurrent edges. A self-connection from a node to itself may also form a cycle. LSTM has the ability to add or remove of information to the internal state of the memory cell, which is controlled by structures called gates. By using some gates in each memory cell, data can be disposed, filtered, or added to the next memory cells. Gates are important and significant features of the LSTM approach.

The Sigmoid function σ of a gate provides numbers between of 0 and 1 which represents the amount of data need to be filtered in each cell. More specifically, an estimated value of 1 indicates that “let everything pass through”; and an estimated value of 0 implies that "let nothing pass through".

All elements included in LSTM cell are described in [83] as follows:

In the original paper and in subsequent works [1, 82, 83, 86], the input node is annotated with g, not to be confused with g as a gate. Seeking comprehensibility, we adhere to the notation of [86], we use i, f, s, and o to represent respectively the input gate, forget gate,



$v_t = s_t \bullet o_t$
o_t
f_t
σ
∏
$s_t = g_t \bullet i_t + s_t \bullet f_t$
i_t
g_t
Edge from Previous time step and current input
Weight fixed at 1
Edge to next time step and itself

Figure 4.2 a LSTM memory cell with a forget gate as described by Gers [1]

internal state and output gate.

Input node: a node g that implements standard activation with the input layer x_t at the current time step and along recurrent edges from the previous time step h_{t-1} of the hidden layer. Typically, the weighted sum input is run through a tanh activation function, although in the original LSTM paper, the activation function is a sigmoid.

Input Gate: that determines which input data need to be stored in the cell and chooses which values will be modified and make a vector of new values that could be updated to the internal state of the memory cell. It takes sigmoid activation from the current data point x_t as well as from the previous time step of the hidden layer. Gate value is used to multiply with the value of another node for decision making of data flow. In this sense, if the gate value is 0, then flow from the other node is cut off. If the value of the gate is 1, all flow is passed through. The value of the input gate multiplies the value of the input node.

Internal State: a node with linear activation is located at the heart of each memory cell, which is referred in the original paper as the "internal state" of the memory cell. The internal state has a self-connected recurrent edge with fixed unit weight. This edge is often called the constant error carousel, that spans next time steps with constant weight, error can flow across time steps without exploding or vanishing. In vector notation, the internal state is updated with:

$$s_t = g_t \odot i_t + s_{t-1} \tag{4.1}$$

where $\odot$ is element wise multiplication.

Forget Gate: decides what information is throw away from the cell state with sigmoid layer. This variant was introduced by Gers [1]. It provides a method that the network can learn to flush the contents of the internal state of the memory cells. The equation for calculating the internal state on the passing forward with forget gates, is

$$s_t = g_t \odot i_t + f_t \odot s_{t-1} \tag{4.2}$$

In the original LSTM version by Hochreiter [82] uses Eq. (4.1) to compute s_t.

Output Gate: provides the output value that will be based on the internal state of the cell along with filtered and current updated data. The output value produced by a memory cell comes from the multiplication of the value of the internal state and the value of the output gate. It is required that the internal state first be run through a tanh activation function. This gives the output of each cell the same dynamic range as an ordinary tanh hidden unit.

The computation in the LSTM model formally proceeds according to the following calculations, which are performed at each time step. These equations give the full algorithm for a LSTM with forget gates:

$$g_t = \tanh(W_{gx}x_t + W_{gh}h_{t-1} + b_g) \tag{4.3}$$

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \tag{4.4}$$

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \tag{4.5}$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \tag{4.6}$$

$$s_t = g_t \odot i_t + f_t \odot s_{t-1} \tag{4.7}$$

$$h_t = \tanh(s_t) \odot o_t \tag{4.8}$$

where W_g, W_f, W_i, W_o are weights of associated inputs, and b_g, b_i, b_f, b_o are bias, and h_{t-1} is the values output by each memory cell from the previous time step $t-1$ in the hidden layer of the LSTM, while the value at time step t is the vector h_t. The simple LSTM without forget gates performs calculations by setting $f_t = 1$ for all t. We use the function tanh for the input node g following the state-of-the-art design of [86]. However, in the original LSTM paper, the activation function for input node g is the sigmoid funtion σ.

In the forward pass, the LSTM can learn when to let activation into the internal state. Once the value is 0, no activation can propagate to the input gate and no value will be produced from output gate. When both gates are closed, the activation is trapped in the memory cell without any growing or any shrinking of the knowledge. During the backward pass, the gradient is also propagated back across many time steps without any exploding or vanishing in the constant error carousel. However, the LSTM has shown a promising ability to learn long-range dependencies as compared to simple RNNs. The output from each memory cell flows to the input node and all gates of each memory cell in the adjacent time step. It is common to include multiple layers of memory cells [86].

4.2.0.1 Deep Knowledge Tracing (DKT)

The DKT model is based on Long Short-Term Memory (LSTM). This model compactly encodes the historical information from previous time steps and determines what and how much information to remember by using input gate, forget gate and output gate of LSTM. Then, it makes a prediction for the current time step by taking recent information into account.

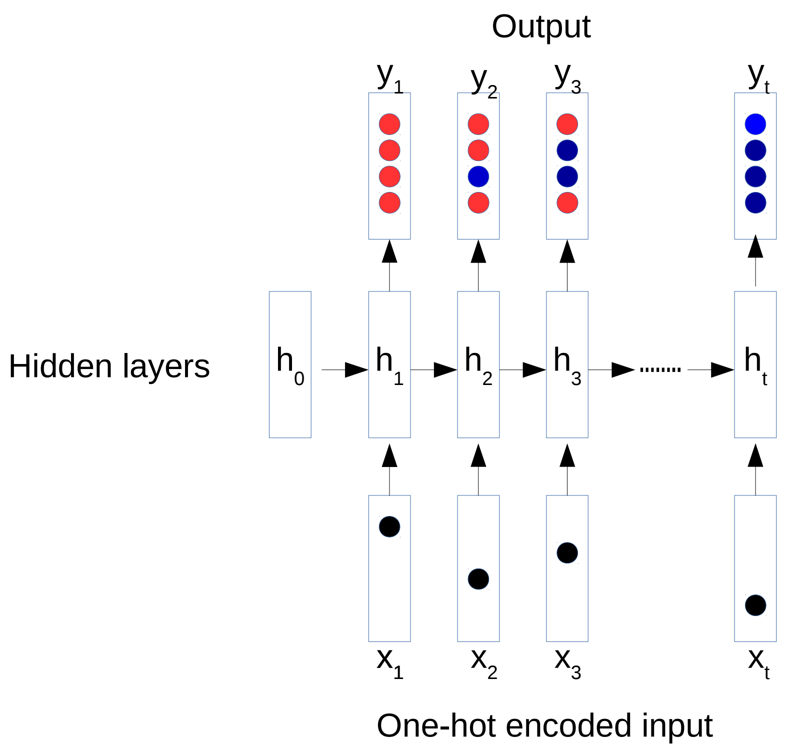


Figure 4.3 DKT unfolded architecture.

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 student's knowledge state from data. It is defined by the following equations:

$$h_t = \tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h), \tag{4.9}$$

$$y_t = \sigma(W_{yh}h_t + b_y). \tag{4.10}$$

In DKT, both tanh and the sigmoid function are applied element wise and parameterized by an input weight matrix W_{hx}, recurrent weight matrix W_{hh}, initial state h_{10}, and output weight matrix W_{yh}. Biases for latent and output units are represented by b_h and b_y. Where x_t is a one-hot encoded vector of the student interaction $x_t = \{s_t, r_t\}$ that represents the combination of which skill s_t was practiced with student response r_t, so $x_t \in \{0, 1\}^{2M}$ according to number M of unique skills. The output y_t is a vector of number of skills, where each value represents the predicted probability that the student would answer that particular problem with associated skill correctly. Thus the prediction of r_{t+1} of next problem associated with skill s_{t+1} can be retrieved from vector of y_t.

4.3 Deep Knowledge Tracing and Dynamic Student Classification (DKT-DSC)

We propose a knowledge tracing model: Deep Knowledge Tracing and Dynamic Student Classification (DKT-DSC) for Knowledge tracing problem and extended form DKT. However, it can capture temporal student's ability profile at each time interval in student's long-term learning process. The aim is that this mechanism will realize what is known as *transfer learning*, the learning that occurs across skills.

We describe the method, setup, analysis,and results of a performance prediction. The code of the model and experiments is publicly available[1].

4.3.1 Student's ability profile

The proposed model rests on the notion of an *ability profile*. The ability profile captures the student's acquired learning pattern, as represented by a skill cluster. It divides students into distinct groups with similar ability profiles, according to their previous performance on various contents in a learning system. This approach has been explored in several works in the field of education [87, 88] for providing more adaptive instructions to each group of students with similar ability. But it had not been defined and tested with a deep learning architecture.

Integrating this mechanism into DKT should allow *transfer learning*. The learning acquired through practice and shared among a cluster of users should in effect realize transfer of learning between skills and help better assess the student knowledge state and predict her performance. Transfer learning implies that practice over items that involves a set of skills that are not deemed necessary to other tasks do affect the chances of success over these tasks, even though the involve a different set of skills.

While it is claimed that some transfer learning occurs in DKT through the mechanism of LSTM [23], it might not be a strong means of transferring learning over a long sequence and across an array of different skills. And while IRT naturally does transfer learning, it does so by having a single skill which compromises a fine-grain assessment. Fine-grain skills

[1]DKT-DSC: https://github.com/simon-tan/DKT-DSC

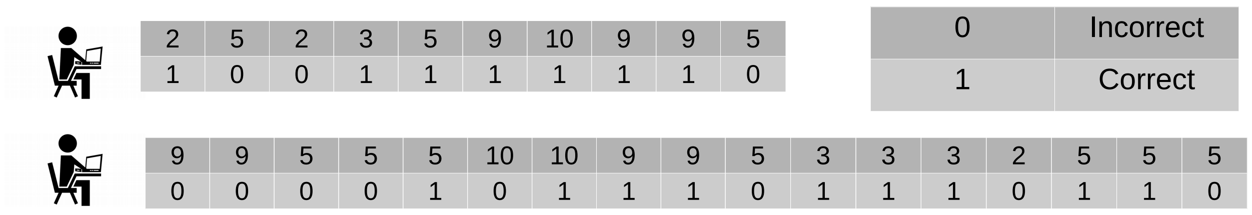
assessment and transfer learning has recently been the topic of research in a factorization architecture [89], and we introduce an analog idea to the DKT model.

Dynamic assessment of student ability profile at each time interval is performed by clustering on the performance of their previous attempts of learning history before the start of next time interval. As the student completes new tasks, her profile will evolve and be used as a feature to better predict task/item attempt outcome. This process is described in more details later in this section (4.3.2). First, we provide some details about the log data and its transformation.

4.3.1.1 Segmenting student attempt sequences

In the current study, we segment student attempt sequences into short “time intervals”. This segmentation will be usefule for the purpose of creating ability profiles as we will see later.

These intervals are not necessarily proportional to actual time lapses. They are proportional to the number of attempts and are chronologically ordered. Where students received the problems in different order of skills and problems.



2	5	2	3	5	9	10	9	9	5
1	0	0	1	1	1	1	1	1	0

0	Incorrect
1	Correct

9	9	5	5	5	10	10	9	9	5	3	3	3	2	5	5	5
0	0	0	0	1	0	1	1	1	0	1	1	1	0	1	1	0

Figure 4.4 Example of students’ attempt sequence with skills and responses (0 represents incorrect and 1 represents correct) .

Segmentation of student response sequences into time intervals serves two purposes:

- To reduce computational burden and memory space allocation for learning throughout a long sequence.
- To re-assess a student’ ability profile after each time interval and assign her into a group which she belongs to for the next time interval dynamically.

A segment represents a time interval in which that student answered a fixed number of problems. That number is set to 6 in the current study, but it can be adapted based on the length of the sequences. While the number of attempts made by each student varies based on the number of questions they answered during their interaction with system, sequences must be of fixed length. When a student stops interacting with the system, a special code (-1) indicates a missing value and it is repeated till the end of the sequence.

Fig. 4.5 illustrates an example of dividing a 24-attempts response sequence of a student into 5 segments (time intervals).

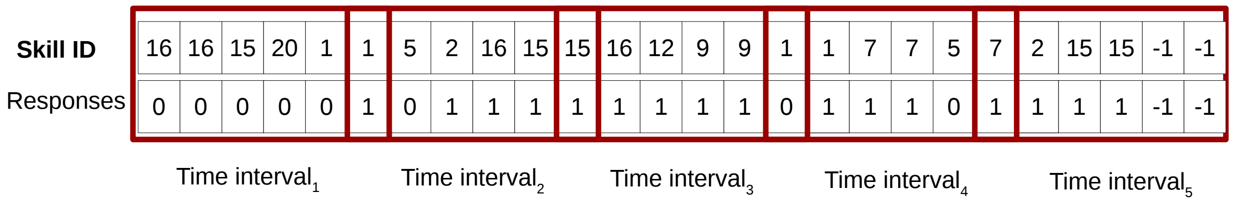


Figure 4.5 Segment of a student's sequence of attempts at different problems involving the corresponding skills.

Time interval: Time interval is a segment containing a number of student's attempts to answer questions in the system. In this perspective, a tick of time is a single attempt to a question or problem.

4.3.1.2 Clustering of skill vectors

Student are grouped according to their ability profile: skills or knowledge they acquired. Data for assessing student's ability profile is available from previous attempts on test items or exercises corresponding to a specific skill.

The ability profile ab is encoded as a cluster ID and computed from the performance vector (with Equation 4.12) of length equal to the number of skills, and updated after each time interval by using all previous attempts on each skill. The success rates on each skill from past attempts data are transformed into a performance vector for clustering student i at time interval 1:z as follows (for brevity we omit indexing all terms by i in equation 4.11):

$$R(x_j)_{1:z} = \sum_{t=1}^{z} \frac{(x_{jt})}{|N_{jt}|}, \tag{4.11}$$

$$d^i_{1:z} = (R(x_1)_{1:z}, R(x_2)_{1:z}, ..., R(x_n)_{1:z}), \tag{4.12}$$

where

- x_{jt}: the attempts of skill x_j being correctly answered at time interval t; successful attempts are set to 1 and 0 otherwise;
- $|N_{jt}|$: the total number of practices of skill x_j up to time interval z;
- n: the total number of skills;
- $R(x_j)_{1:z}$ represents ratios of skill x_j being correctly answered from time interval 1 to current time interval z by student i. This is computed for all skills $(x_1, x_2, .., x_n)$;
- $d^i_{1:z}$ represents a *performance vector* of student i on all skills from time interval 1 until z.

Each student may have a different number of total time intervals in the lifetime of their interactions with the system (see Figure. 4.7).

If a student's time interval has no attempt between interval $0 : z$, we apply 0.5 for success rate, $R(x_j)_{1:z}$.

4.3.1.3 Clustering Students

Assigning students into a group with similar ability at each time interval is performed by k-means clustering [90, 91] on their performance vectors, d_* for detecting student profile at each time interval.

Clusters are computed from a training sample before the knowledge tracing phase. In this study, we set the number of clusters to 8. While the assignment of student to a cluster changes during the knowledge tracing phase, cluster are fixed once determined during the training phase. Figure 4.6 illustrates the general process which is further detailed below.

k-Means clustering: The k-means algorithm is used in partitioning a given set of observations into a predefined number of K clusters. The algorithm as described by Macqueen [90] starts with a random set of K centroids (μ). During each update step, all observations x are assigned to their nearest centroids. Afterwards, the centroids are repositioned by calculating the mean of the assigned observations to the respective centroids. If multiple centroids have the same distance to the observation, a random assignment would be performed. The update process is repeated until the set of centroids converges to stationary values. Learning of the centroids is only done by using performance vectors d_* from students in training dataset. Once it has been learned students in both training and testing datasets are assigned into different cluster according to $d_{1:z-1}$ at each time interval z.

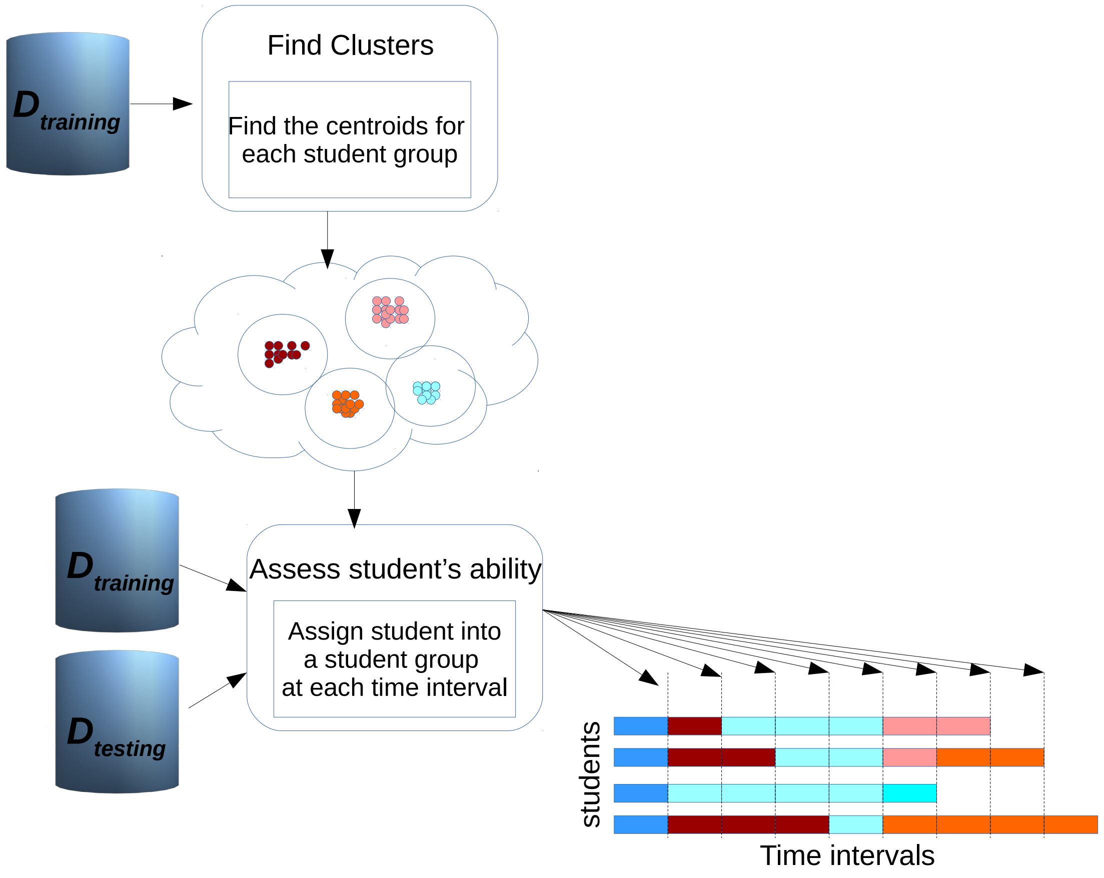


Figure 4.6 Clustering students at each time interval.

After learning the centroids of all K clusters, each student i at each time interval z is assigned to the nearest cluster C_c by the following equation:

$$Cluster(Stu_z^i) = \arg\min_c ||d_{1:z-1}^i - \mu_c||^2$$

where centroids μ_c is the mean of points for cluster C_c, and performance vector $d_{1:z-1}^i$ is the average performance data of student i from time interval 1 to $z-1$.

$$ab_z^i = Cluster(Stu_z^i) \tag{4.13}$$

Figure 4.7 illustrates the data of 33 students' ability profiles, one per line, based on their previous performance and the evolution over time intervals. Dark blue (-1) means students have no more attempts until the end of the period. Default group $ab_{\text{default}} = 1$ is for the first

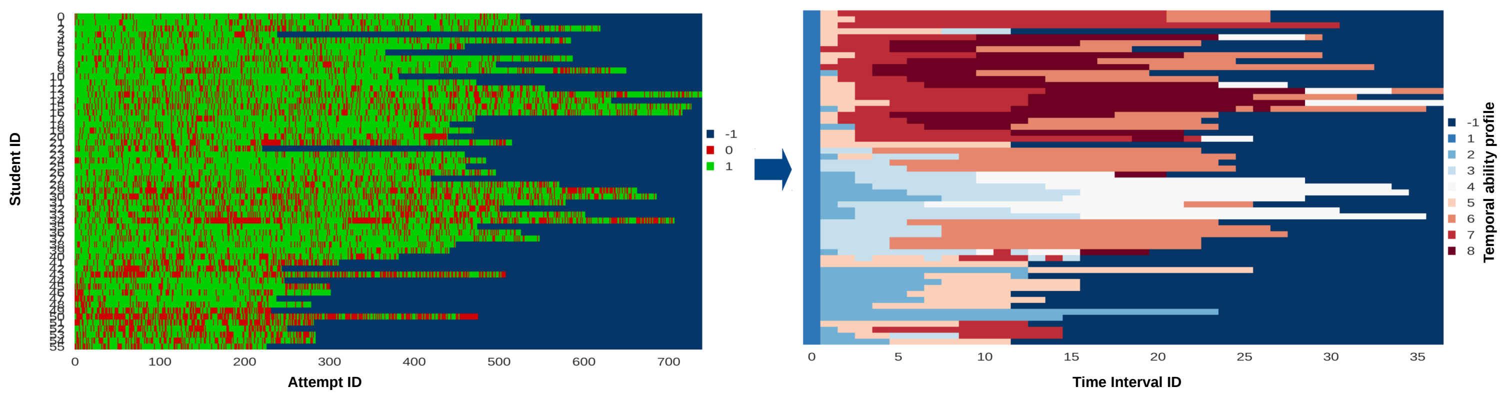


Figure 4.7 Evolution of students' ability profile over each time interval (each time interval contains 20 attempts) throughout their interactions. The graphic on the left shows the success (1), failures (0), or no attempt (-1), per student, and the graphic on the right shows the corresponding cluster at each corresponding time interval.

time interval of every student and the rest of the groups $(2, .., 8)$ are assigned by the k-means clustering method at each time interval z by using previous performance data $d^i_{1:z-1}$.

4.3.2 Architecture of DKT-DSC

The DKT-DSC model is described in more details in this section.

Recall that DKT-DSC incorporates student's ability profile to the DKT for better individualization of the system, by assigning a student into a group of students with similar ability dynamically. It relaxes the assumption that all students have the similar ability and that students' ability is consistent over time.

In the standard DKT, x_t is a one-hot vector (binary unit vector) of the student interaction tuple $x_t = \{s_t, r_t\}$ that represents the combination of the skills s_t practiced, and of r_t which indicates the response. DKT-DSC adds to the input $x_t = \{s_t, r_t\}$ the temporal ability profile ab_z also represented one-hot vector that indicates the cluster ID $Cluster(stu^i_z)$, the temporal ability profile ab_z of student i at current time interval z (for brevity we omit indexing all terms by student i from here on).

The output y_t is a vector of length equals to the number of skills. The predicted probability that the student would correctly answer that particular problem with a given ability profile at time $t+1$ can be obtained from $p(s_{t+1}) \in y_t$. The output y_t of both DKT and DKT-DSC is the same.

Figure 4.8 illustrates how the DKT-DSC model has been adapted by incorporating student's ability profile at each time interval (each segment) to improve individualization in knowledge

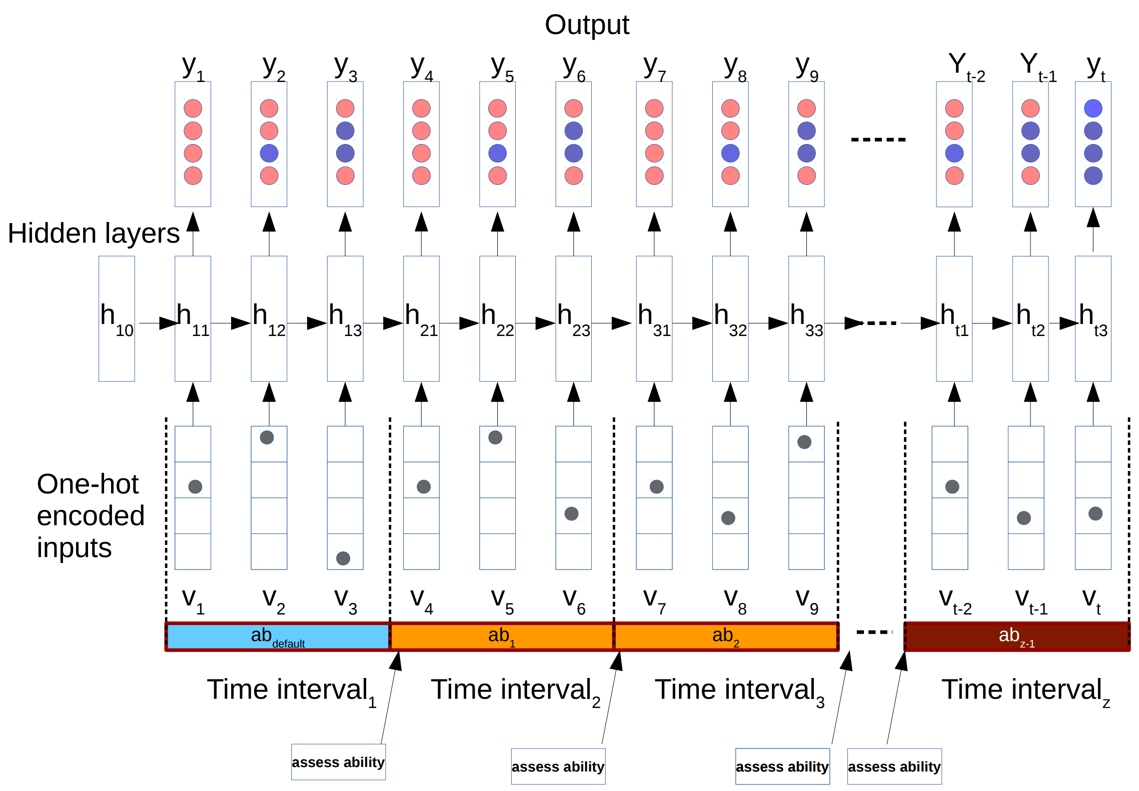


Figure 4.8 DKT-DSC prediction in each time interval (each segment) is associated with a temporal ability profile $ab_{(.)}$ at each time interval throughout interactions of a student with the system.

tracing. The colour at each time interval in the input layer represents which group a student belongs to at that time interval according to her ability profile.

By adding this cluster ID ab_z (ability profile) of what group the student belongs to, we ensure that these high-level skill profiles are available to the model for making its predictions throughout the long term interaction with the tutor.

DKT-DSC also applies same equations as DKT as follows:

$$h_t = \tanh(W_{hx}[x_t, ab_z] + W_{hh}h_{t-1} + b_h), \tag{4.14}$$

$$y_t = \sigma(W_{yh}h_t + b_y). \tag{4.15}$$

Both tanh and the sigmoid function are applied element wise and parameterized by an input

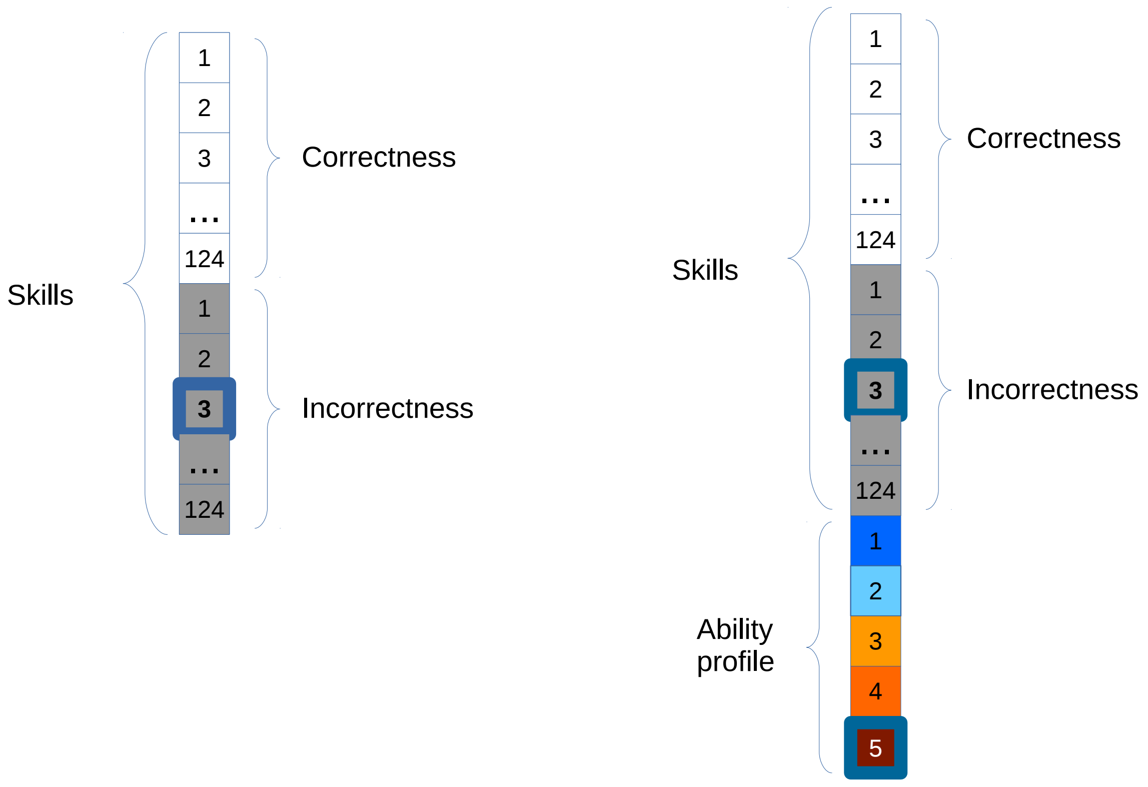


Figure 4.9 Difference between input vectors for DKT (*left*) and DKT-DSC(*right*).

weight matrix W_{hx}, recurrent weight matrix W_{hh}, and output weight matrix W_{yh}. Biases for latent and output units are represented by b_h and b_y. The input $[x_t, ab_z]$ includes concatenation of two one hot encoded vectors: $x_t = (s_t, r_t)$ including skill s_t and response r_t, and ab_z, the temporal ability profile at time interval z. The output y_t is same as in DKT as well, where each value represents the predicted probability that the student would answer that particular problem with associated skill correctly. Thus the prediction of r_{t+1} of next problem associated with skill s_{t+1} can be retrieved from vector of y_t. Without applying students's temporal ability profile, the DKT-DSC model represents the standard DKT model, therefore the DKT-DSC model can be thought of as integrating student temporal ability profile information to DKT.

4.4 Experimental Study

The DKT-DSC architecture is evaluated by comparing its capacity to predict the next item outcome with alternative models. This is the choice of comparison method most studies have used. We complement this evaluation with a more detailed analysis of the temporal ability profile at the end of the section.

4.4.1 Datasets

Validation is conducted over four public datasets from two distinct tutoring scenarios in which students interact with a computer-based learning system in the educational settings: 1) ASSISTments [2]: an online tutoring system that was first created in 2004 which engages middle and high-school students with scaffolded hints in their math problem. If students working on ASSISTments answer a problem correctly, they are given a new problem. If they answer it incorrectly, they are provided with a small tutoring session where they must answer a few questions that break the problem down into steps. Datasets are as follows: ASSISTments 2009-2010 (skill builder), ASSISTments 2012-2013, ASSISTments 2014-2015. 2) Cognitive Tutor, and Algebra 2005–2006 [92] [3] is a development dataset released in KDD Cup 2010 competition from Carnegie Learning of PSLC DataShop.

Table 4.1 Overview of datasets

Dataset	Number of				Description
	Skills	Problems	Students	Records	
Cognitive Tutor	437	15663	574	808,775	KDD Cup 2010 [92]
ASSISTments	123	13002	4,163	278,607	2009-2010 [93]
	198	41918	28,834	2,506,769	2012-2013 [21]
	100	NA	19,840	683,801	2014-2015 [94]

4.4.1.1 ASSISTments

The ASSISTment system[4] [5] [6] is an online tutoring system that was first created in 2004 which engages middle and high-school students with scaffolded hints in their math problem. It consists of hundreds of items generated from a number of different templates, all pertaining

[2]https://sites.google.com/site/assistmentsdata/

[3]https://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp

[4]https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010

[5]https://sites.google.com/site/assistmentsdata/home/2012-13-school-data-with-affect

[6]https://sites.google.com/site/assistmentsdata/home/2015-assistments-skill-builder-data

to the same skill or skill grouping. Students are marked as having completed the problem set when they answer three items correctly in a row without asking for help. If students working on ASSISTments answer a problem correctly, they are given a new problem. If they answer it incorrectly, they are provided with a small tutoring session where they must answer a few questions that break the problem down into steps.

ASSISTments 2009-2010: This came with two different datasets: *skill builder* dataset associated with formative assessment where a student works until a mastery threshold is reached [93] and the *non skill builder* dataset associated with summative assessment where a student has a fixed number of problems.

ASSISTments 2012-2013: This is a large dataset with 2.5 million records during the 2012 and 2013 school years. This dataset was initially produced with the purpose of analyzing student behaviour with a new additional factor, affect [21].

ASSISTments 2014-2015: This dataset contains records that represent a sequence of attempts to a related set of problems in a mastery learning problem set [94]. It contains no individual problem identifier and therefore problem difficulty cannot be computed from this data set.

4.4.1.2 Cognitive Tutor

The PSLC DataShop released several datasets derived from Carnegie Learning's Cognitive Tutor. Algebra 2005-2006 [92] is a development dataset released in KDD Cup 2010 competition[7]. Although both ASSISTments and Cognitive Algebra Tutor are the systems used for practising and learning mathematics skills for solving problems, KDD Cup dataset is actually rather different from ASSISTments.

The Cognitive Tutor uses Knowledge Tracing to determine when a student has mastered a skill. A problem in the tutor can also consist of questions of differing skill. However, once a student has mastered a skill, as determined by KT, the student no longer needs to answer questions of that skill within a problem but must answer the other questions which are associated with the unmastered skill(s).

It divides lessons into multiple pieces called Units. A skill that appears in one unit is treated as another separate skill when appearing in a different Unit. It is built around scaffolded,

[7]`https://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp`

formative assessment, where each step a student takes to answer a problem is counted as a different activity where each step potentially assessing proficiency of different skills.

4.4.1.3 First correct attempt

For all datasets, only first correct attempts to original problems are considered in our experiment. This is a standard practice in the field. We remove data with missing values for skills and problems with duplicate records. To the best of our knowledge, these are the largest publicly available knowledge tracing datasets.

4.4.2 Methodology

We compare next problem student performance prediction of our model with state-of-the-art models mentioned in section 2.3.1: Bayesian extensions of IRT* [47], BKT [18], PFA [57], DKT [23]. But we do not compare with other variant models, because those are more or less similar and do not show significant difference in performance. For IRT*(Bayesian extensions of IRT), we apply the code from Knewton [47] and the code for DKT is from WPI [94]. For DKT, we use the same setting of parameters as DKT-DSC and also apply segmentation for a fair comparison. Predicted sequences of student performance by each model are tabulated and evaluated in terms of Area Under the Curve (AUC) and Root mean squared error (RMSE). AUC provides a robust metric where the value to predict is binary, as it is the case of our datasets. An AUC of 0.50 represents the score achieved by random guess. We set AUC of 0.61 by BKT as a baseline in our experiment.

Table 4.2 Hyperparameters of DKT-DSC

Name	Default Value
Number of training steps	20
Learning rate	0.001
Dropout rate	0.6
Epochs	100
Batch size	32
Lambda for l2 loss	0.3
Number of Hidden layers	1
Epsilon value for Adam Optimizer	0.1
Optimizer	Adam Optimizer

Table 3. hyperparameters across experiments

In our experiment, 5 fold cross-validations are used to make predictions on both datasets. Each fold involves randomly splitting each dataset into 80% training data and 20% test data at the student level. So both training and test datasets contain response records from different students. Finding the centroids of clusters is performed only using data from students in the training dataset. We use EM to train BKT and the limit of iterations is set to 200. We learn models for each skill and make predictions separately, then the results for each skill are averaged. For DKT and DKT-DSC, we set the number of epochs to 100. All these models are trained and tested on the same sets of data. Next response of a student is predicted by using current and previous response sequence in chronological order.

Forty iterations were made in the training stage. DKT-DSC and DKT share the same loss function, with 200 fully-connected hidden nodes for each hidden layer.For speeding up the training process, mini-batch stochastic gradient descent is used to minimize the loss function. The batch size for our implementation is 32. We train the model with a learning rate of 0.01 and dropout is also applied for avoiding overfitting [95]. The standard DKT applies one-hot encoding vector, we apply one-hot encoded vectors in both of DKT and DKT-DSC for fairness of comparison. Specifically speaking, DKT-DSC without using student's temporal ability profile represents standard DKT.

4.4.3 Results

[8]

Results of the comparison are reported in this section. Recall that the comparison is done over four other models and Table 4.4 summarizes the characteristics of each model DKT-DSC is compared to.

Results are shown in Tables 4.3 (a) and (b). The datasets are labeled by their source and year in the case of ASSISTments. For eg. ASSISTments09 refers to *Assistments 2019-2010* (page 70).

Consistent with previous studies, DKT outperforms BKT by a large margin, while DKT-DSC outperforms DKT by a smaller margin and mostly on AUC results. However, a notable result is that, in three of the four data sets, IRT* comes out as the top predictor. Given that IRT* with a single latent skill, this is an unexpected result that shows the importance of comparing with classic approaches.

There are differences in the results of original DKT and DKT reported in this analysis. In original DKT [23], the author utilized the data with duplicate records. We removed duplicate records and only take first-correct attempts into consideration in our data. More detailed

[8]Note that the results are different than the published version [77]. A bug was found and corrected. While the scale of the differences is lower, the general trends are identical. The major difference is that DKT-DSC performed better than IRT* in the published results, whereas it does not in the corrected results.

Table 4.3 Experimental results of DKT-DSC with other models

(a) AUC

Datasets	Model				
	BKT	DKT	IRT*	PFA	DKT-DSC
Cognitive Tutor	64.2±1.0	78.4±0.6	**81.0±0.2**	76.0 ±0.3	79.2±0.5
ASSISTments09	65.1±1.0	72.1±0.5	**75.0±0.5**	70.0 ±0.2	73.5±0.6
ASSISTments12	62.3±0.0	71.3±0.0	**74.2±0.6**	67.2 ±0.4	72.1±0.1
ASSISTments14	61.01±1.0	70.7±0.4	67.0±0.5	69.1 ±0.1	**71.6±0.2**

(b) RMSE

Datasets	Model				
	BKT	DKT	IRT*	PFA	DKT-DSC
Cognitive Tutor	0.44±0.00	0.38±0.01	**0.37±0.00**	0.39±0.00	**0.37±0.01**
ASSISTments09	0.47±0.01	0.45±0.00	0.44±0.01	0.45±0.01	**0.43±0.00**
ASSISTments12	0.51±0.00	**0.43±0.00**	0.44±0.01	**0.43±0.01**	**0.43±0.00**
ASSISTments14	0.51±0.00	**0.42±0.00**	0.44±0.02	**0.42±0.00**	**0.42±0.00**

Table 4.4 Comparison of different models

	IRT*	PFA	BKT	DKT	DKT-DSC
Use of ability profile	No	No	No	No	Yes
Use of item difficulty	Yes	Yes	No	No	No
Learn on single skill	Yes	No	Yes	Yes	Yes
Learn on multiple skills	No	Yes	No	No	No
Learn on sequence	No	No	Yes	Yes	Yes

analysis on DKT performance with various types of data processing in different deep learning platforms can be referred into the paper [94].

The strong predictive performance of IRT* can be explained by two factors.

(1) IRT* uses item difficulty in its prediction. This is supported by the fact that ASSISTments14 only has information on skills tried and does not allow the computation of item difficulty. For this data set, IRT* is above BKT, but below all other models.

(2) IRT*, being a single skill model, effectively ends up allowing learning transfer across skills, albeit at the cost of fine-grain skill assessment.

The first factor is evidence of the need to take item difficult into account in student models, which is the topic of the next chapter.

4.4.4 Analysis of student ability profiles

In order to gain better insight of the student ability profiles, we analyze in greater details their composition and evolution in this section.

First, we assess the strength of the correlation between ability profiles and observed success rate across time intervals. Results are reported in Table 4.5. We find a moderately high correlation of 0.793 between the centroids of ability profiles.

Table 4.5 Average correlation between Ability profiles and observed success rates across time intervals

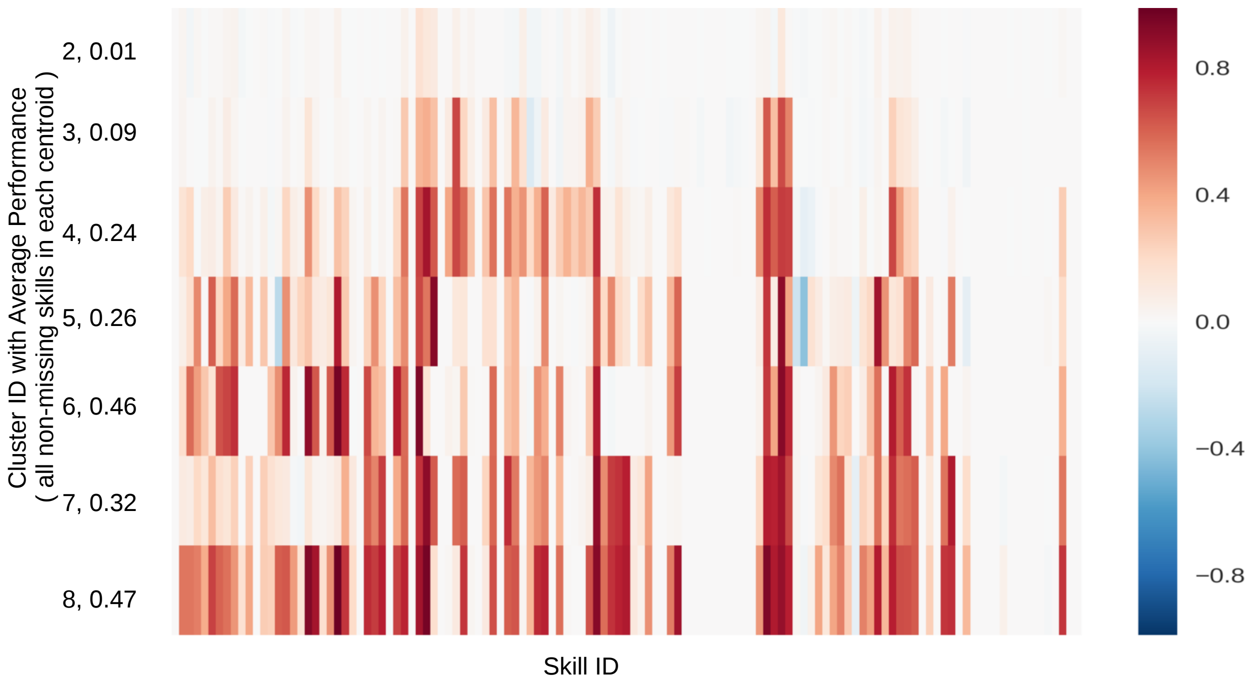
	Ability Profile
Success Rate	0.793

The evolution of ability profiles across time intervals is shown in figure 4.10.

The top graph of the figure shows the breakdown of the clusters per skill. We find 123 skills on the x axis and 7 clusters numbered 2 to 8. The y-axis shows the clusters and the mean success rate (from -1 to 1), and the color represents the skill average success rate of the corresponding cluster. We can see that from cluster 2 to 8, the success rate tends to increase, from an average of 0.01 to 0.47, although this trend is not uniform across skills. However, this is possibly due to the initial starting point at 0.5 rather than to a learning gain as we will see later.

The bottom graph of Figure 4.10 shows the cluster assigned to each student at each time interval, from 0 to 50. An horizontal line represents a sequence cluster assignments per interval, per student. Note that for this graph, if we do not observe any attempt yet for a skill at a time interval, we set the success rate $= 0.5$ for this skill. Note also that all students are forced to start in the same cluster, labeled 1 in 4.10 (bottom graph), for interval 1, since we would not have enough data to do a reliable cluster in any case. Cluster 0 indicates that no more data is available for the last time intervals.

In order to get a further insight of how clusters are formed, we show the evolution of each skill success rate as a function of the time interval in Figure 4.11. The top graph is a replication of top one in Figure 4.10. The bottom graph shows the time interval from 1 to 23 (y-axis) for a single student taken as an example. Before a skill is tested, its success rate is set to 0 by default. Therefore, at sequence 1, only a few skills are tested and the other skills are have a success rate of 0, whereas at sequence 23 over half of the skills are tested. Interestingly, for this student, we find relatively little change of each skill's success rate. In part, this may be



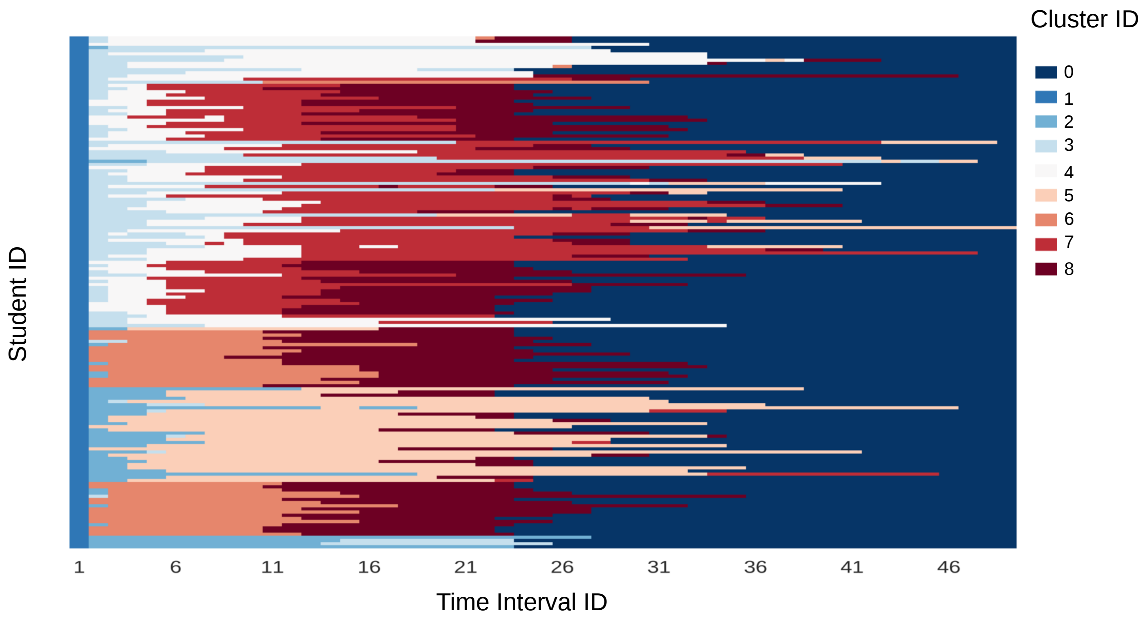
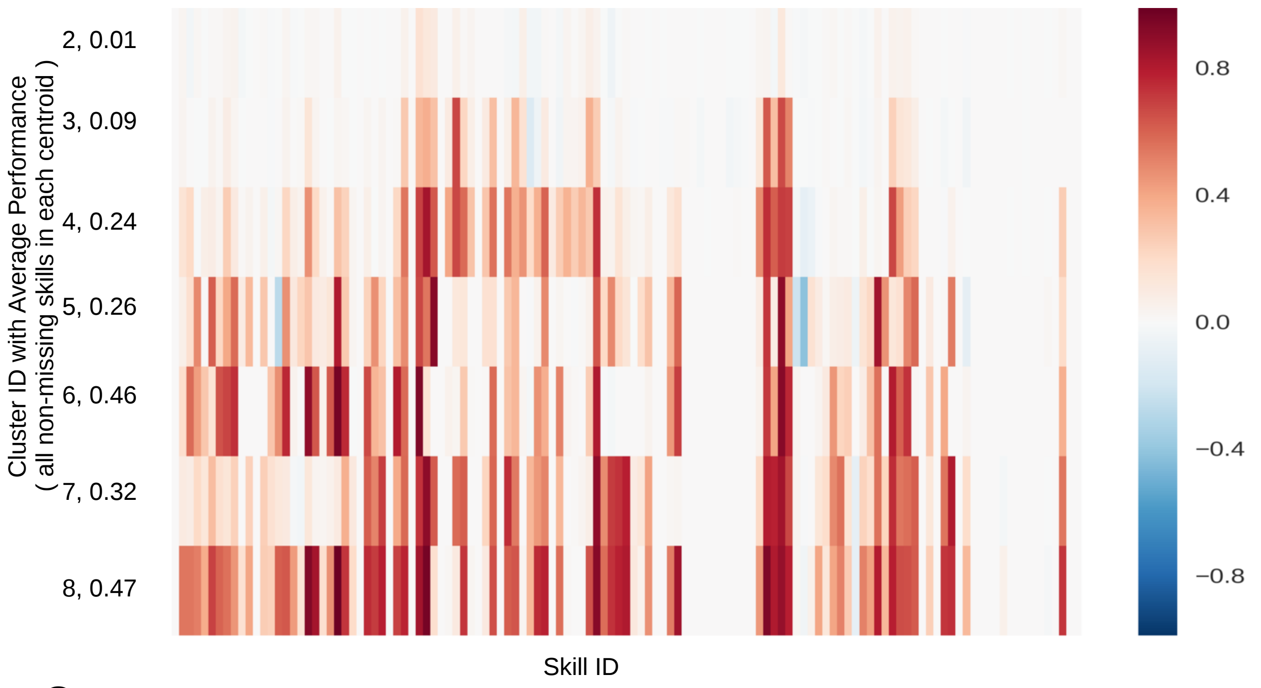


Figure 4.10 Assignment on student ability profile based on her performance: Performance values in each centroids (above) and assigment of student ability profile based on her performance (bottom).

because skills are not retested, but we also see that sometimes they are, and the rate does not necessarily increase.

The rate evolution is analyzed in Figure 4.12. Recall that the average performance of a student a time interval is calculated based on the average value of non-missing skills in the performance vector $d^i_{1:z}$ from Equation 4.12. All students start in cluster 1 by default, and most of the students sequences start in cluster 2 because because of the large number of untested skills. Cluster 2 of Figure 4.12 contains students with both of positive and negative performance up to time interval 5, but it is more likely to have students with lower or negative performance in later time intervals. For the other clusters, we can see the average performance of students that belongs to each group in Figure 4.12.

Finally, we also analyse the ratio of missing skills at each time interval in Figure 4.13. We can see that the ratio of missing skills decreases as students practice more and more skills. Different proportions of missing skills are spread across various clusters.



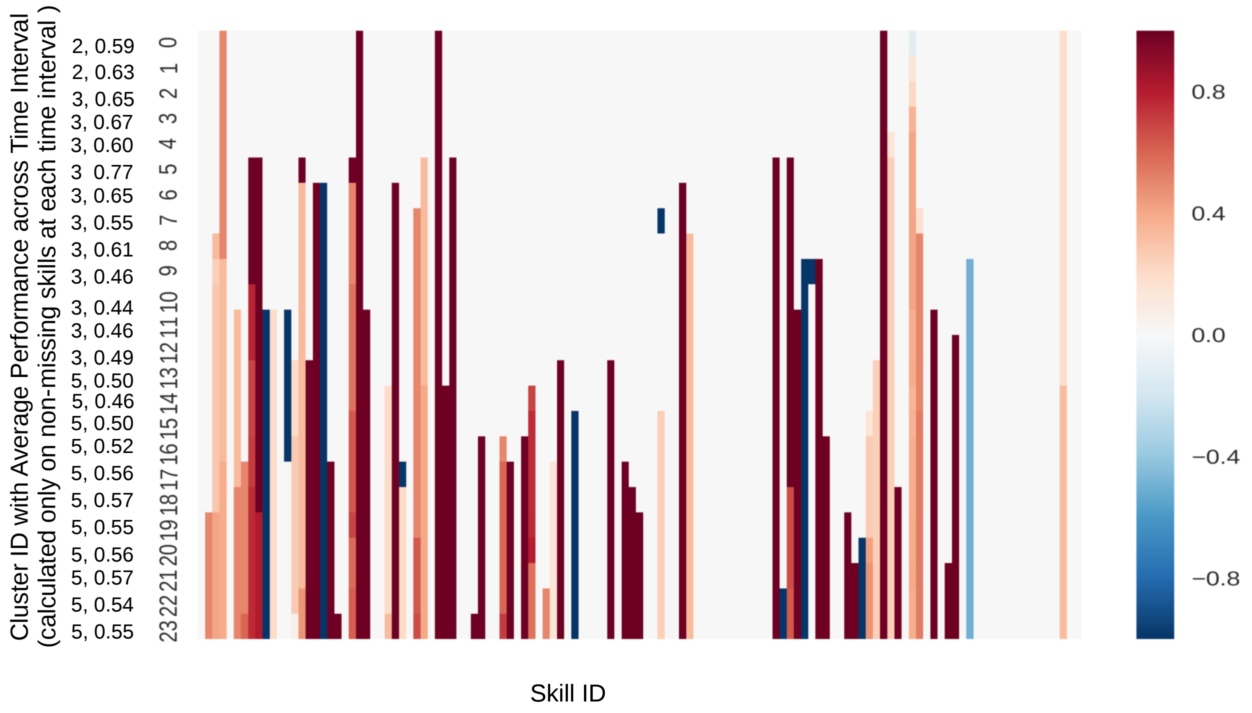


Figure 4.11 Student ability profile based on her performance: Performance values in each centroids (above) and example of a student ability evolution based on her performance on each skill (bottom).

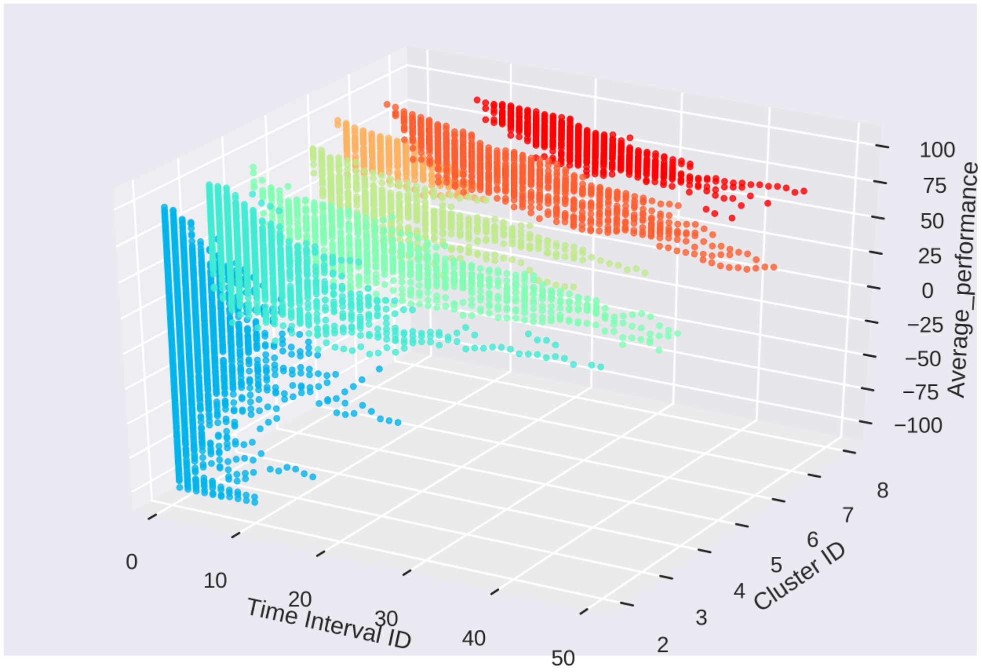


Figure 4.12 Student's average performance over all non-missing skills in each group at time interval respectively

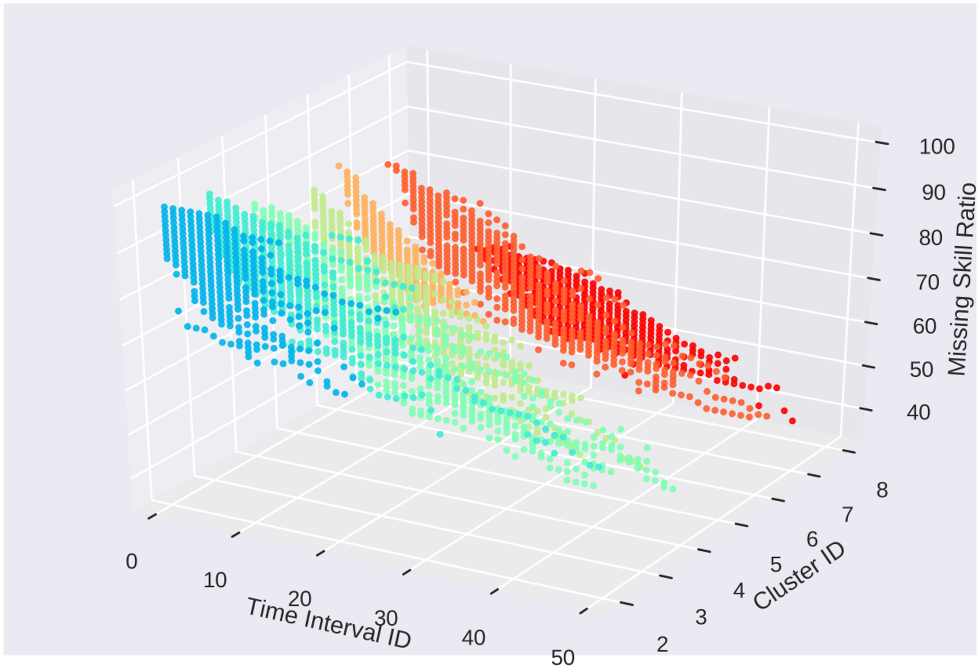


Figure 4.13 Percentage of missing skills associated with each group at time interval respectively

4.5 Discussion

In this study, we proposed a new model, DKT-DSC that assesses student's ability profile at each time interval and dynamically assigns a student into a distinct group of students with the same ability. A student's knowledge is traced based on the group which she belongs to at each time interval. Experiments with four datasets show that the proposed model performs better than the DKT model. DKT assumes all students have the same ability profile and only tracks the improvement of knowledge in a skill sequence without considering difference between ability profiles. In comparison, our model improves over DKT by capturing the student's ability profile over time. Assessing student's ability in this way gives the model critical information in the prediction of student performance in their next time interval and tracing their knowledge where ability profiles of the students evolve dynamically.

This study also confirms an unexpected result that was first observed by Wilson [47]. The IRT* model is a better performer for predicting next item outcome and considered the state-of-the-art student modelling. This result can be explained by the item difficulty factor which is explicitly taken into account by IRT* and not by DKT nor DTK-DSC. This conclusion is further supported by the observation that the advantage of IRT* vanishes for the Assistment-14 data set, for which item difficulty is not accessible because only the skill involved in the exercises is given.

This finding brings us to the next chapter where we elaborate a model based on a different deep learning architecture that takes into account item difficulty in addition to aiming to model learning transfer with the ability profile.

CHAPTER 5 KNOWLEDGE TRACING WITH MEMORY NETWORKS, ITEM DIFFICULTY, AND ABILITY PROFILE

This chapter introduces a novel knowledge tracing method that incorporates long-term ability profile and problem difficulty. This contribution was published in the PA-KDD Conference [96]. The approach builds on previous work that was shown highly effective for Knowledge Tracing, DKVMN [70], and relies on a special kind of RNN called Memory Networks. We first review the principles of Memory Networks. Then the approach is described and evaluated in comparison with comparable approaches.

5.1 Memory Networks

The RNN architecture, and LSTM in particular, has achieved high performance to predict the next output after having read a sequence of inputs [82]. However, memory in RNNs (encoded by weights and hidden states) is typically too small to accurately memorize large amount of knowledge and facts from the past. RNNs are known to have difficulty in memorizing long-term dependencies [69, 97]. Weston [97] proposed memory networks that combine the successful learning strategies developed in RNNs with an explicit external memory module that can be read and written to. These networks are a special kind of RNN. They have been used in domains where a long term memorization is required such as question answering.

Weston [97] describe the structural flow of the original memory networks model, from input to output, as the following steps:

1. Convert input to an internal feature representation.
2. Update memory state given the new input.
3. Produce output features given the new input and the memory state.
4. Decode output features to give final response.

A memory network is formally constructed with a memory module M and four components I, G, O and R [97] as follows:

- Input feature map I: converts the input into the internal feature representation.

- Generalization G: updates old value of memory state with incoming new input. This process allows the network to compress and generalize memories at current stage for future usage.
- Output feature map O: produces a new output in the space of feature representation according to the current memory state and the new input.
- Response R: changes the output into a response or an action as desired.

The model parameters I, G, O and R are not updated after the training phase, but memories will be used for storage and updated in both of training and testing phases.

I **component**: converts the incoming input to internal feature representation with required standard pre-processing: e.g., convert input into one-hot encoded vector, dense feature vector.

G **component**: is responsible to store $I(x)$ into a memory slot of M:

$$M_i = G(M_i, I(x), M), \forall i \in H(.) \tag{5.1}$$

where $H(.)$ is a function to select memory slot for storage of $I(x)$. In its simplest form, $G(x) = I(x)$. Index $H(x)$ will be updated by G while keeping all other parts of the memory untouched. There are variants of G and sophisticated ones could go back and update previous stored memories according to incoming evidence of current input x. G can be defined to operate only on memories that are on a specific topic. When memory becomes full, H performs a procedure of "forgetting" which consists in choosing the memory to replace.

O **component**: computes the output o by calculating and reading from relevant memories k chosen by function $H(.)$ e.g. $|K| = 2$, It is possible to generalize with larger K. For $k = 1$ and $k = 2$ of the highest scoring on relevant memory is retrieved with:

$$o_1 = O_1(x, M) = argmax_{i=1}^{N} s_O(x, M(i)) \tag{5.2}$$

$$o_2 = O_2(x, M) = argmax_{i=1}^{N} s_O([x, M_{o_1}]M(i)) \tag{5.3}$$

where s_O is a function for scoring the match between input $I(x)$ and $M(i)$. The relevant memory $M(i)$ is calculated according to both of incoming input and the first supporting memories. $[x, m_{o_1}, m_{o_2}]$ represents the final output o given to the R component for producing the response.

R **component**: is responsible for producing the final response format based on received output o from the O component. R could be an RNN that is conditioned on the output.

The scoring function s_O is similar to an embedding model;

$$s(x, y) = \Phi_x(x)^T U^T U \Phi_y(y) \tag{5.4}$$

where U is a $N \times d$ matrix where d is the number of input features and N is the embedding dimension. The role of Φ_x and Φ_y is to map input features into the d-dimensional feature space.

G and O components can apply attention mechanisms in read an write operation by focusing both on content and location of the memory [98, 99]. The reading and writing operations are performed through attention weight.

Attention mechanisms can be defined as the function of mapping a query and key-value pairs to an output. The output is calculated from weighted sum of the values according to attention weight based on the query with the corresponding key.

A simple way to compute the attention or correlation weight is using a cosine similarity or an inner product of the input and each memory slot and then a softmax transformation:

$$\text{softmax}(z_i) = e^{z_i} / \sum_j e^{z_j} \tag{5.5}$$

where z_i is each element of the input vector z.

Additionally, there is also a pure content-based memory writer called least recently used access (LRUA) module. It was proposed in [100] to write the key either to the least recently used or to the most recently used memory location.

5.1.1 Dynamic Key-Value Memory Networks (DKVMN)

DKVMN was proposed as an alternative to DKT that is inspired from the memory network architecture [70]. It utilises an external memory neural network module and uses two memory slots called key memory and value memory to encode the knowledge state of students. Assessments of knowledge state on particular skills are stored in memory slots and controlled by read and write operations through additional attention mechanisms. DKVMN calculates attention weight of inputs by using key memory, which is immutable, and reads and writes to the corresponding value memory. DKVMN takes a skill s_t as query, estimate the probability of response $p(r_t|s_t)$ to the problem with associated skill s_t, and then updates the value memory with skill and response $x_t\{s_t, r_t\}$. With the following definitions:

- Key memory matrix M_k (size $N \times d_k$): stores N latent concepts $\{c^1, c^2, ..., c^n\}$ of underlying skills with size d_k.
- Value memory matrix M_t^v (size $N \times d_v$): student's understanding of each concept: concept states $\{c^1_{stu,t}, c^2_{stu,t}, ..., c^n_{stu,t}\}$ at time t stores their values with size d_v.
- d_k: size of the key memory matrix (set at 200 for our experiment)
- d_v: size of the value memory matrix (set at 400 for our experiment)

Note that DKVMN consumes more memory storage than DKT.

The overall model architecture of DKVMN is shown in Figure 5.1. Unlike DKT, DKVMN applies read and write operations to perform knowledge state transitions in memory rather than unstructured state-to-state knowledge transformation in hidden layer of DKT. Knowledge state of a student student is traced by reading and writing to the value memory slots using attention weight computed from input skill and the key memory slots. It comprise three main steps:

(1) Reading: The mastery m_t of s_t is retrieved by weighted sum of values in value memory slots by using attention weight w_t from Eq (5.13):

$$m_t = \sum_{i=1}^{N} (w_t(i) M_t^v(i)) \tag{5.6}$$

(2) Prediction: Given that each skill has its own difficulty, it concatenates the read content m_t and the input skill embedding k_t and then passes it through a fully connected layer with a tanh activation function to get a summary vector f_t, which contains the student's mastery level with the prior difficulty of the skill. The probability of answering the problem with underlying skill $p(s_t)$ is calculated by:

$$f_t = \tanh(W_1^T [m_t, k_t] + b_1) \tag{5.7}$$

where b_1 is the bias vector. Finally, f_t is passed through another fully connected layer with a sigmoid activation to predict the student's performance:

$$p(s_t) = \sigma(W_2^T f_t + b_2) \tag{5.8}$$

Where $\tanh(z_i) = (e^{z_i} - e^{-z_i})/(e^{z_i} + e^{-z_i})$ and $\sigma(z_i) = 1/1 + e^{-z_i}$ and p_t is a scalar that represents the probability of getting correct answer to problem associated with skill s_t.

(3) Writing: After the student answers the problem, the model will update the value memory according to response (r_t) of the student. A joint embedding of $x_t = (s_t, r_t)$ is converted into embedding values v_t and written to the value memory with same attention weight w_t from Eq (5.13) used in read process for updating the knowledge growth v_t of the student after solving this problem. Before writing the student's knowledge growth into the value memory, erasing is performed before adding new information by using erase vector:

$$e_t = \sigma(E^T v_t + b_e), \tag{5.9}$$

where E the transformation matrix is of shape d_v x d_v, e_t is a erase vector with d_v elements that all lie in the range $(0, 1)$. b_e is the bias vector. The vectors of value memory $M^v_{t-1}(i)$ from the previous timestamp are modified as follows:

$$\tilde{M^v_t}(i) = M^v_{t-1}(i)[1 - w_t(i)e_t], \tag{5.10}$$

where 1 is a row-vector of all 1-s. If both the weight at the location and the erase element are 1, the elements of a memory location are reset to zero. No changes are performed in the case of either erase signal or the weight is zero.

After erasing the previous memory, a_t is used to update each memory slots in value memory:

$$a_t = \tanh(D^T v_t + b_a)^T, \tag{5.11}$$

$$M^v_t(i) = M^v_{t-1}(i) + w_t(i)a_t, \tag{5.12}$$

where D is the transformation matrix with shape of $d_v \times d_v$ (see section 5.1.1. b_a is the bias vector. This erase-followed-by-add mechanism allows forgetting and strengthening concept states of student learning process [70] which is not possible in other RNN based models.

The overall model architecture is shown in Figure 5.1.

In addition to these steps, two other processes should be mentioned, Optimization and Attention.

Attention: The attention weight of input skill s_t is computed by utilizing the softmax activation of the inner product between k_t and key memory slot $M^k(i)$:

$$w_t = \text{softmax}(k_t^T M^k(i)) \tag{5.13}$$

where k_t is a continuous scale embedding vector of s_t and softmax is differentiable. Attention weight w_t is used in both reading and writing process and it represents the correlation between a skill and each latent concept.

Optimization: During training, both embedding matrices for s_t and x_t, as well as other parameters and the initial value of M^k and M^v are jointly learned by minimizing a standard cross entropy loss between p_t and the true label r_t cross-entropy loss l between p_t and actual response r_t for as follows:

$$l = \sum_t (r_t \log p_t + (1 - r_t) \log(1 - p_t)), \tag{5.14}$$

In the training process, the initial value of both the key and the value matrices are learned. Each slot of the key memory are embedded with concepts and is fixed in the testing process.

5.2 Dynamic Student Classification on Memory Networks (DSCMN)

We propose a knowledge tracing model called Dynamic Student Classification on Memory Networks (DSCMN) [1] for Knowledge tracing problem that extends DKVMN. However It can:

- capture temporal ability profile at each time interval in student's long-term learning process;
- assess mastery of knowledge state from each of their answers simultaneously;
- utilize problem difficulty in prediction of student performance.

The approach improves the next problem prediction performance. It is an extended version of DKT-DSC and DKVMN. It has a static matrix called *key*, which stores the latent concepts and the other dynamic matrix called value, and stores and updates the mastery levels of corresponding concepts.

Akin to DKVMN, it also comes with N latent concepts $\{c^1, c^2, ..., c^n\}$ that are independent of the skills in the ability profile and explicitly linked to items. These concepts are stored in the key matrix M_k (of size $N \times d_k$) and the student's understanding of each concept: concept states $\{c^1_{stu,t}, c^2_{stu,t}, ..., c^n_{stu,t}\}$ are stored in the value matrix M^v_t (of size $N \times d_v$), which changes over time.

[1]DSCMN: https://github.com/simon-tan/DSCMN

Despite being able to assess the mastery of skills or being more powerful in student performance prediction, each of the above models has deficiencies for dealing with KT task. In both DKT and DKVMN, temporal student's long-term ability profile is ignored when the prediction is performed. Although DKT-DSC assesses student's mastery of skill based on current level of ability profile but it does not take the problem difficulty into account in prediction of student next performance.

To handle all these deficiencies, we proposed a novel model called Dynamic Student Classification on Memory Networks (DSCMN) by leveraging advantageous of DKVMN and DKT-DSC. DSCMN predicts student performance based on both of evaluated temporal student's long-term ability and assessed mastery of skills simultaneously at each time interval. It also takes problem difficulty in student performance prediction too.

5.2.1 Student ability profile

The Student ability profile has been explained at length in chapter 4. In DKVMN, the process is much the same. A vector of length number of skills encodes the student's observed past performance at each time interval, and Euclidian distance is used to associate this vector to the closest ability profile. Ability profiles are learned in advance from the training data using K-means. Figure 5.2, reproduced from from chapter 4, shows the evolution of ability profiles (right) for a sample of students whose trace of successes/failures is shown (left).

The process of defining ability profiles and tracing each student's closest profile (cluster of the K-means) is an independent module in the memory network architecture and serves as separate input to the neural network.

5.2.2 Calculating problem difficulty

The integration of problem difficulty do predict student performance is a distinct feature of DSCMN. Problem difficulty is added to the input of each attempt as a one-hot vector of 10 cells, corresponding to the item difficulty level. Calculation of the difficulty level is explained below. Note that, in this study, we assume each problem is associated with a single skill, but the difficulty is associated with problems, not with skills themselves.

The difficulty of a problem, p_j, is determined on a scale of 1 to 10. This scale is used as an index into the one-hot vector: a value of 1 is set to the cell with the corresponding difficulty level, and 0 for all others. Problem difficulty level is calculated as:

$$\textbf{difficulty level}(p_j) = \begin{cases} \delta(p_j), & \text{if } |N_j| \geq 4 \\ 5, & \text{else} \end{cases} \tag{5.15}$$

where:

$$\delta(p_j) = \text{modulo}_{10}\left(\frac{\sum_i^{|N_j|} O_i(p_j)}{|N_j|} \cdot 10\right) \tag{5.16}$$

and where

- p_j: problem j
- N_j is the set of students who attempted problem p_j
- $O_i(p_j)$ the outcome of the first attempt from student i to problem p_j, 1 if a success, 0 otherwise

$\delta(p_j)$ is a function that maps the average success rate of problem p_j onto (10) levels. Unseen problems, those that do not have any record, and problems with less than 4 students ($|N_j| < 4$) in the dataset, will have a difficulty level of 5.

Figure 5.3 shows the distributions of successes and failures over skills as a function of problem difficulty, and along the number of opportunities of practice over that skill. As expected, more failures are found at the higher difficulty levels, and also at early attemps over a given skill. Notice also that failures after a number of attempts on a given skill do occur, but only at the higher problem difficulty levels. These results demonstrate that problem difficulty does play an important role on the chances of success or failures to a problem, even for well practiced skills.

5.2.3 Proposed Student Model: DSCMN

To assess the mastery of skills according to ability profile, DSCMN uses read and write operations into two key and value memory slots as in DKVMN. DSCMN also assesses the mastery of skills using the attention weight computed from input skills and the key memory. In DSCMN, instead of using embedding values, one-hot encoded inputs are directly fed into memory networks. Knowledge state of a student is traced by reading and writing to the value memory slots using attention weight computed from input skill and the key memory slots as follows.

Reading: The mastery m_t of skill s_t is retrieved by weighted sum of values in value memory slots by using attention weight w_t from Eq (5.24):

$$m_t = \sum_{i=1}^{N}(w_t(i)M_t^v(i)) \tag{5.17}$$

Mastery m_t of skill s_t is obtained from the reading process before writing x_t to value memory. The model will write x_t into value memory using equations: erasing according to Eq. (5.19) and adding through Eq. (5.21) after student answered the problem at time t.

Writing: After the student answers the problem, the model will update the value memory according to student response (r_t). A one-hot encoded vector of $x_t = (s_t, r_t)$ concatenating skill id s_t and response r_t is written to the value memory with same attention weight w_t from Eq (5.24) used in read process. Erasing is performed over vector e_t before adding new information:

$$e_t = \sigma(E^T x_t + b_e), \tag{5.18}$$

$$\tilde{M}_t^v(i) = M_{t-1}^v(i)[\mathbf{1} - w_t(i)e_t], \tag{5.19}$$

where $\mathbf{1}$ is a row-vector of all 1s. If both the weight at the location and the erase element are 1, the elements of a memory location are reset to zero. No changes are performed in the case of either an erase signal or if the weight is zero. After erasing previous memory, adding vector a_t is used to update each memory slots in value memory.

$$a_t = \tanh(D^T x_t + b_a)^T, \tag{5.20}$$

$$M_t^v(i) = M_{t-1}^v(i) + w_t(i)a_t, \tag{5.21}$$

Where E and D are the transformation matrix of dimension $d_v \times d_v$. b_e and b_a are bias vectors. This erase-followed-by-add mechanism allows forgetting and strengthening concept states of student learning process [70] which is not feasible in other RNN based models.

Prediction: The probability of answering the problem with underlying skill $p(s_t)$ of student in temporal ability profile ab_z at time interval z is estimated by feeding previous response and mastery of skill in the ability profile of student into hidden layer h and prediction $p(s_t^{ab_z}) \in y_t$ is calculated as follows:

$$h_t = \tanh(W_{hx}[x_{t-1}, m_t, pd_t] + W_{hh}h_{t-1} + b_h), \tag{5.22}$$

$$y_t = \sigma(W_{yh}h_t + b_y), \tag{5.23}$$

Where ab_z is the ability profile of that student at time interval $t \in z$ and the input $[x_{t-1}, m_t, pd_t]$ is concatenation of $x_{t-1} = (s_{t-1}, r_{t-1})$ including previous response r_{t-1} and practiced skill s_{t-1}, and m_t mastery of skill s_t and associated problem difficulty pd_t in ability profile of student i at time interval z. Biases for hidden and output units are represented by b_h and b_y. Both tanh and the sigmoid function are applied element wise and parameterized by an input weight matrix W_{hx}, recurrent weight matrix W_{hh}, and output weight matrix W_{yh}. The probability of $p(s_t^{ab_z})$ is in vector of y_t, which contains predicted probability of next problem with associated skills being answered correctly. DSCMN possesses the ability to assess the mastery of skill based on temporal long-term ability profile and takes problem difficulty in student performance prediction. Prediction is performed by using these factors and stored it in hidden state h_t.

Attention: The attention weight of input skill s_t is computed by utilizing the softmax activation of the inner product between k_t and key memory slot $M^k(i)$:

$$w_t = \text{softmax}(k_t^T M^k(i)) \tag{5.24}$$

Where $k_t = [s_t, ab_z]$ is the concatenation of two one-hot encoded vectors: one represents skill s_t and another indicates ability profile ab_z at time interval z. Attention weight w_t is used in both reading and writing process in later. The softmax function is differentiable.

Optimization: To improve the predictive performance of RNN based models, we trained with the cross-entropy loss l between $p_t = p(s_t^{ab_z})$ and actual response r_t for all RNN based models as follows:

$$l = \sum_t (r_t \log p_t + (1 - r_t) \log(1 - p_t)), \tag{5.25}$$

The initial value of both the key and the value matrices are learned during training process. Each slot of the key memory are embedded with concepts and is fixed in the testing process.

Parameters setting. For both DKVMN and DSCMN, d_v is set at 400 . However, for DKVMN we have $d_k = 400$, whereas for DSCMN this parameter is set to the sum of the number of skills and the number of difficulty levels, $d_k = |s| + |pd| = 123 + 10$.

5.3 Model Comparison

Finally, we summarize the characteristics of all models in this thesis in Table 5.1.

Temporal ability profile used in DKT-DSC and DSCMN is estimated from clusters of students with similar ability profile in each time interval, but student ability of IRT* is the probability of answering the problem correctly according to parameters of that problem .

Unlike the problem difficulty used in DSCMN, the problem difficulty in PFA is learned as the effect of prior successes and prior failures for each skill. The parameters of item (including problem difficulty) of IRT* is estimated from the modelling of student's personal ability. IRT* and PFA do not consider the student's sequence practice order. Differences between each models are graphically illustrated in Figure 5.5.

Models like IRT*, DKT, DKVMN, DKT-DSC and DSCMN are only able to learn problems with a single skill, while only PFA model can consider multiple skills in a problem at the same time. Finally, models like IRT* and PFA that take into account problem difficulty provide stronger performance prediction.

5.4 Datasets

In order to validate the proposed model, we tested it on four public datasets from two distinct tutoring scenarios in which students interact with a computer-based learning system in educational settings.

These datasets are described in section 4.4.1. For convenience they are described again below with emphasis on their major characteristics.

- The ASSISTment system[2] is an online tutoring system that was first created in 2004 which engages middle and high-school students with scaffolded hints in their math problems. If students working on ASSISTments answer a problem correctly, they are

[2]`https://sites.google.com/site/assistmentsdata`

Table 5.1 Comparison of different models

	IRT*	PFA	BKT	DKT	DKVMN	DKT-DSC	DSCMN
Ability profile	No	No	No	No	No	Yes	Yes
Problem difficulty	Yes	Yes	No	No	No	No	Yes
Learn on Single/multiple skills	Single	Multiple	Single	Single	Single	Single	Single
Learn on sequence	No	No	Yes	Yes	Yes	Yes	Yes

given a new problem. If they answer it incorrectly, they are provided with a small tutoring session where they must answer a few questions that break the problem down into steps. Datasets are: ASSISTments 2009-2010 skill builder data set, ASSISTments 2012-2013, ASSISTments 2014-2015.

In all datasets, problems are usually tagged with just one skill, but a rare few may be associated with two or three skills. It typically depends on the structure given by the content creator. Some researchers separate a record with multiple skills into multiple single skill records by duplicating. Wilson [47] claimed that this type of data processing can artificially boost prediction results significantly, because these duplicate rows can be accounted for approximately 25% of the records in the Assistment09 dataset for DKT models. So we removed data with missing values for skills and duplicate records for the fairness of comparison. For all datasets, only first correct attempts to original problems are considered in our experiment.

- KDD Cup: The PSLC DataShop released several data sets derived from Carnegie Learning's Cognitive Tutor. Algebra 2005-2006 [92] is a development dataset released during the KDD Cup 2010 competition[3]. In this dataset, the problems are associated with multiple skills. So we regard a subset of multiple skills as a new skill [94].

Table 5.2 contains descriptive statistics of the data sets. Note that Assistment14 [94] does not have the number of problems because it only contains no item identification, only the skill tested. This implies that item difficulty cannot be taken into account and will affect results for models that take this factor into account.

We evaluate models described above with four datasets from two separate real world tutors. The experimental results show how the models perform across different datasets. Only the first correct attempts to original problems are considered in our experiment.

To the best of our knowledge, these are the largest publicly available knowledge tracing datasets.

5.5 Experimental Study

In this experiment, we set a time interval as 20 attempts by a student. The total number ability profiles in our experiment is set at 8 for DKT-DSC and DSCMN. Five fold cross-validations are used to make predictions on both datasets. Each fold involves randomly splitting each dataset into 80% training data and 20% test data at the student level. Initial

[3]`https://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp`

Table 5.2 Overview of datasets

Dataset	Number of				Description
	Skills	Problems	Students	Records	
Cognitive Tutor	437	15663	574	808,775	KDD Cup 2010 [92]
ASSISTments	123	13002	4,163	278,607	2009-2010 [93]
	198	41918	28,834	2,506,769	2012-2013 [21]
	100	NA	19,840	683,801	2014-2015 [94]

values in both key and value memory of DKVMN and DSCMN are learned in the training process. Each slot of the key memory is used for storage of concepts, initial values in value memory represents concept state prior difficulty and is fixed in the testing process.

We implement all models with Tensorflow and DKT, DKT-DSC and DSCMN share the same structure with 200 fully-connected hidden nodes for LSTM hidden layer. In DKVMN and DSCMN, key memory size is set to 200 and value memory to 400. For speeding up the training process, mini-batch stochastic gradient descent is used to minimize the loss function. The batch size for our implementation is 32. We train the model with a learning rate 0.01 and dropout is also applied for avoiding over-fitting [95]. We set the number of epochs to 100. All these models are trained and tested on the same sets of data. For BKT, we use Expectation Maximization (EM) algorithm and the limit of iterations is set to 200. We learn models for each skill and make predictions separately and the results for each skill are averaged.

Table 5.3 Hyperparameters of DSCMN

Name	Default Value
Number of training steps	20
Learning rate	0.001
Dropout rate	0.6
Epochs	100
Batch size	32
Lambda for l2 loss	0.3
Number of hidden layers	1
Number of hidden nodes	200
Epsilon value for Adam Optimizer	0.1
memory size	200
Value memory size	400
Optimizer	Adam Optimizer

Table 5.4 DSCMN: Comparison of AUC results

Datasets	Model				
	BKT	DKT	DKVMN	DKT-DSC	DSCMN
Cognitive Tutor	64.2±1.0	78.4±0.6	78.0±0.0	79.2± 0.5	**86.0±0.5**
ASSISTments09	65.1±1.0	72.1±0.5	71.0±0.5	73.5±0.6	**81.2±0.4**
ASSISTments12	62.3±0.0	71.3±0.0	70.7±0.1	72.1±0.1	**78.5±0.1**
ASSISTments14	61.01±1.0	70.7±0.4	70.0±0.1	**71.6±0.2**	71.0±0.01

5.5.1 Results

Results are shown in Table 5.4. DSCMN performs significantly better than state-of-the-art models in three out of four datasets. DSCMN model can achieve AUC of 86, with significant improvement with a notable gain of 10% to original DKT and DKVMN, and 8% to DKT-DSC. In ASSISTments09 dataset, DSCMN also achieves about 10% gain with maximum test AUC=81.2 when compared to DKT-DSC with 73.5, while the original DKT has AUC=72.1 and DKVMN has AUC=71.0. On the ASSISTments12 dataset, DSCMN achieved maximum test AUC=78.5, well above all others.

Results differ for the latest ASSISTments14 dataset (which contains more students and less data compared to other three datasets and without problem information). A likely explanation is that the absence of item difficulty information apparently deprives the DSCMN of critical information to boost its predictive performance.

Table 5.5 DSCMN: Comparison of RMSE results

Datasets	Model				
	BKT	DKT	DKVMN	DKT-DSC	DSCMN
Cognitive Tutor	0.44±0.00	0.38±0.01	0.38±0.00	0.37±0.01	**0.35±0.00**
ASSISTments09	0.47±0.01	0.45±0.00	0.45±0.01	0.43±0.00	**0.40±0.00**
ASSISTments12	0.51±0.00	0.43±0.00	0.43±0.00	0.43±0.00	**0.40±0.00**
ASSISTments14	0.51±0.00	**0.42±0.00**	**0.42±0.00**	**0.42±0.00**	**0.42±0.00**

In Table 5.5, when we compare the models in term of RMSE, BKT is minimum 0.44 in Cognitive Tutor and, maximum 0.51 in ASSISTments12 and ASSISTments14 and 0.47 in ASSISTments09. RMSE results in all dataset with maximum of 0.40 for DSCMN while all other models are no more than 0.42 (except DKT in the Cognitive Tutor dataset and DSCMN in ASSISTments14). According to these results, DSCMN shows significantly better performance than DKT-DSC and than other models in Cognitive Tutor, ASSISTments09, ASSISTments12, except for DKT-DSC in ASSISTments14. Here again, the absence of item difficulty can explain the difference of results for the ASSISTments14 data set. This is further investigated in the next section.

5.6 Ablation studies

The results have so far suggest there may be different impacts of each factor on the predictive performance of NN based models, and in particular the impact of item difficulty. This question is further analyzed in this section.

We compare both our LSTM based and Memory Networks based models through an ablation study with following factor combinations:

- DKT-DSC: LSTM based knowledge tracing model with ability profile.
- DKT-PD: LSTM based knowledge tracing model with problem difficulty.
- DKT-DSC-PD: LSTM based knowledge tracing model with ability profile and problem difficulty.
- DSCMN: Memory networks based knowledge tracing model with ability profile and problem difficulty.
- DKVMN-DSC: Memory networks based knowledge tracing model with ability profile.

When prediction for r_{t+1} of next problem associated with skill s_{t+1} is performed in LSTM based models:

- DKT-DSC: refer to Chapter 4 for DKT-DSC.
- DKT-PD: The concatenation of two one-hot encoded vector x_t and pd_{t+1} is fed to LSTM.
- DKT-DSC-PD: The concatenation of three one-hot encoded vector x_t, pd_{t+1} and ab_z is fed to LSTM

When prediction for r_t of problem associated with skill s_t is performed in memory networks based models:

- DSCMN: we refer to Chapter 5 for DSCMN.

- DKVMN-DSC: same as DSCMN, but pd_t is removed from equation 5.22 in the prediction step.

The results are reported in tables 5.6 and 5.7.

Although ability profile provides indication in student ability evolution in student learning process but it just have mostly similar or improvement in student performance prediction around 1 to 2 %. Results from each model with different combination of factors show us which features provide more information in student performance prediction among various datasets. In there, we can also see that the problem difficulty factor is the most influential factor to student performance prediction. When we apply problem difficulty to both of LSTM base model and memory networks based model, it increases around 8 to 10 % respectively. It also reflects that is why IRT* shows better performance in student performance prediction among other models (except neural networks based models with problem difficulty see Tables 4.3a, 4.3b,5.4 and 5.5). For ASSISTments14 dataset, all of the models (including IRT*, see Tables 4.3a and 4.3b) show similar performance, because it does not have problem information in dataset and we cannot calculate problem difficulty for each problem.

5.7 Discussion

In this research, we proposed a novel model that is called DSCMN, which can predict the student performance by gathering information from skills, problems and student: mastery level of skills of a student in various problems at each time step, along with student ability profile at each time interval and take problem difficulty to account in prediction of student performance. DSCMN is extended from DKT-DSC by leveraging of the student's ability profile and considering problem difficulty in student performance prediction. Experiments with four datasets show that the proposed model performs statistically and significantly better in predictive performance than state-of-the-art KT models. In DSCMN, we improve over DKT-DSC by using memory networks in assessment of student skill mastery according to

Table 5.6 AUC results of factors combinations

Datasets	Model				
	DKT-DSC	DKT-PD	DKT-DSC-PD	DSCMN	DKVMN-DSC
Cognitive Tutor	79.2± 0.5	85.5±0.5	85.6±0.5	**86.0±0.5**	79.8 ±0.06
ASSISTments09	73.5±0.6	80.6±0.4	**81.2±0.4**	**81.2±0.4**	72.7±0.06
ASSISTments12	72.1±0.1	78.3 ±0.1	**78.5±0.1**	**78.5±0.1**	72.5±0.01
ASSISTments14	**71.6±0.2**	71.3±0.03	71.4±0.03	71.0±0.01	71.2 ±0.02

Table 5.7 RMSE results of factors combinations

Datasets	Model				
	DKT-DSC	DKT-PD	DKT-DSC-PD	DSCMN	DKVMN-DSC
Cognitive Tutor	0.379±0.002	0.350±0.003	**0.349±0.004**	0.350±0.003	0.375±0.002
ASSISTments09	0.434±0.002	0.410±0.002	0.406±0.001	**0.405±0.002**	0.435±0.004
ASSISTments12	0.427±0.001	0.406±0.001	**0.405±0.001**	**0.405±0.001**	0.425±0.001
ASSISTments14	**0.419±0.000**	**0.419±0.000**	0.420±0.000	**0.419±0.000**	0.424±0.001

their ability profile and utilize the important information of problem (difficulty) in prediction of student performance. Dynamic evaluation of student's ability profile at each time interval and problem difficulty at each time step plays critical roles and helps DSCMN capture more variance in the data, leading to more accurate and personalized performance predictions.

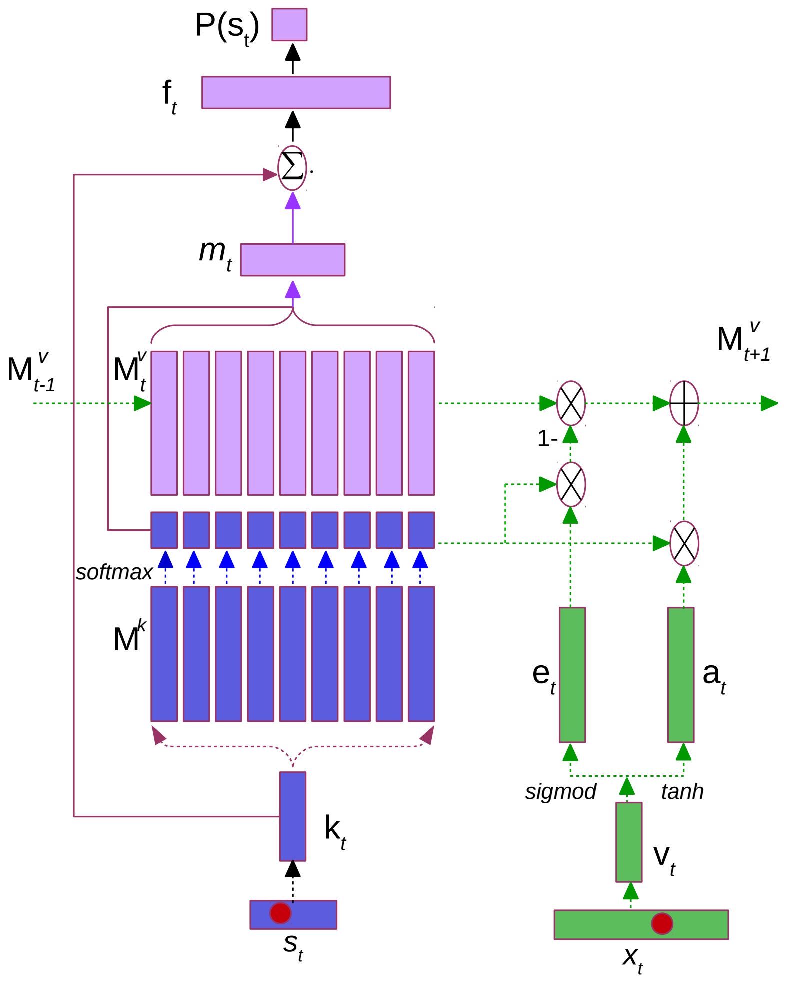


Figure 5.1 DKVMN architecture.

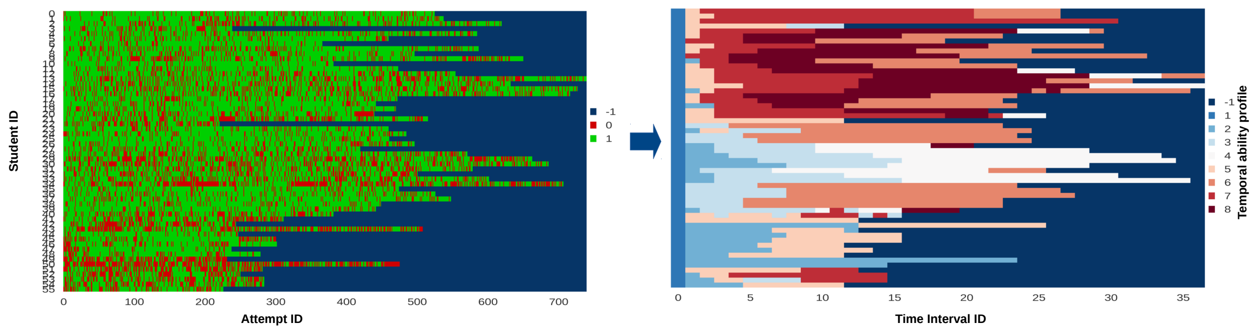


Figure 5.2 Evolution of ability profile in long-term learning process of random 56 students in ASSISTments 2009 dataset

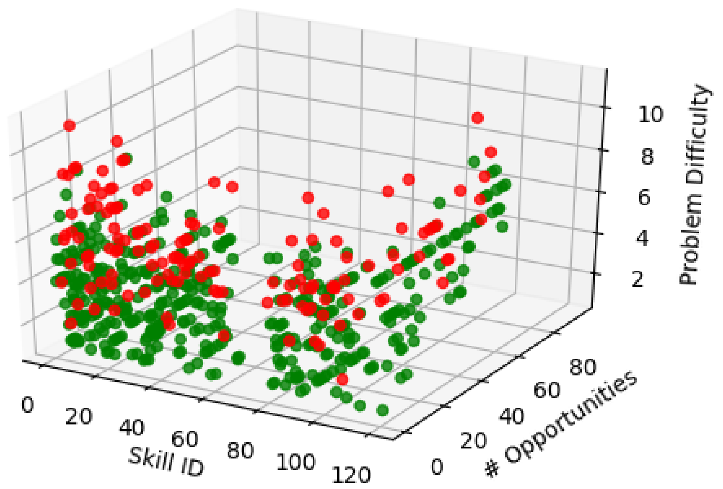


Figure 5.3 a student's responses in different levels of problem difficulty at each trial: red is incorrect response and green is a correct response.

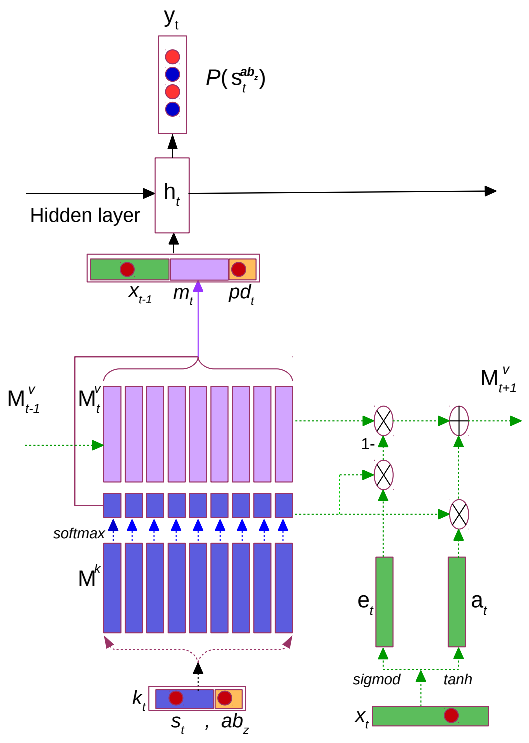


Figure 5.4 Architecture of DSCMN

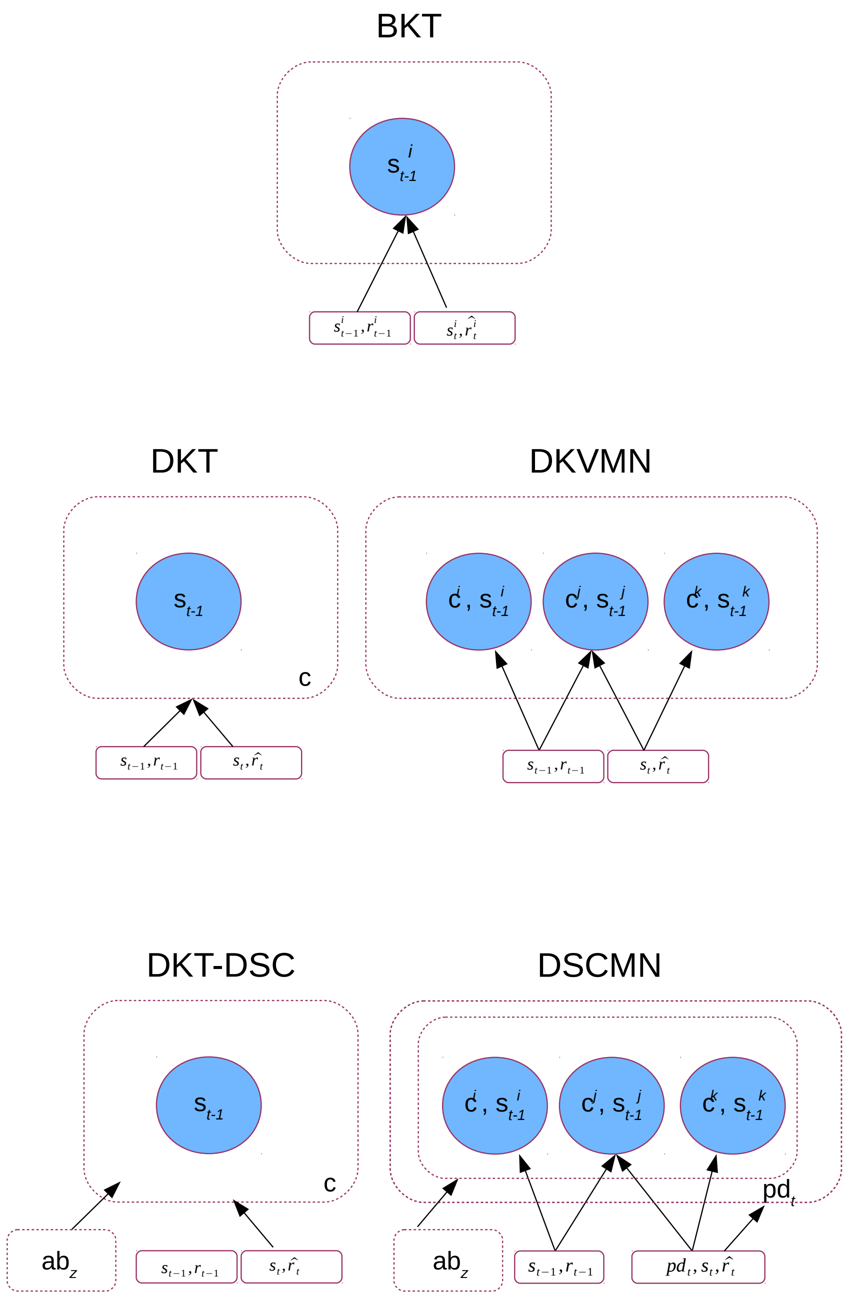


Figure 5.5 Graphical representation of model differences between BKT, DKT, DKVMN, DKT-DSC and DSCMN. BKT is based on skill specific modelling. DKT summarizes values in hidden layer to model the knowledge state. DKVMN uses the concept state for each concept for assessing the knowledge state of a student. DKT-DSC extends DKT by utilizing evaluated ability profile at each time interval. DSCMN traces students' knowledge state based on temporal ability profile and problem difficulty simultaneously.

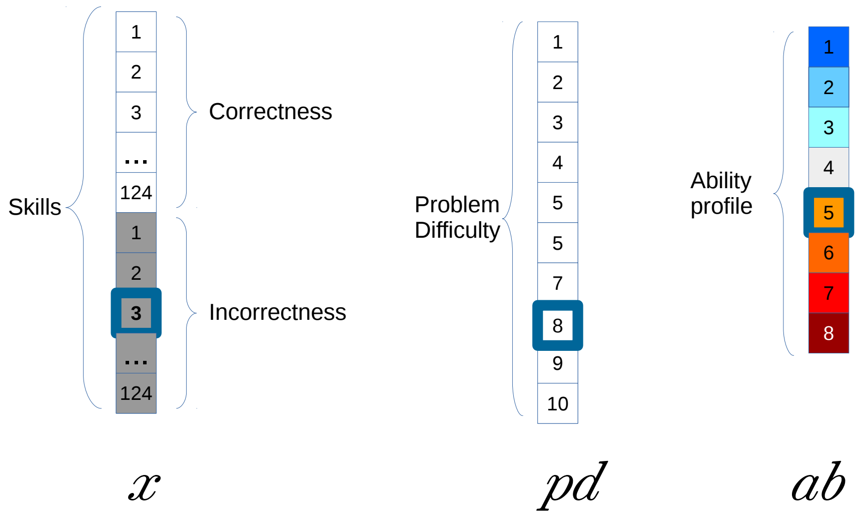


Figure 5.6 one-hot encoded vector for $x = (s, r)$ including skill s and response r, problem difficulty pd and ability profile ab.

CHAPTER 6 CONCLUSION

In this chapter, I summarize the conclusions of all studies on Q-matrix refinement and knowledge tracing problems, then discuss the limitations and future works.

6.1 Conclusion

In this thesis, I worked on two kinds of problems: the Q-matrix refinement problem and the knowledge tracing problem. I proposed a Q-matrix refinement framework with an ensemble learning technique that relies on multi-label classification. I also proposed two knowledge tracing models called Deep Knowledge Tracing and Dynamic Student Classification (DKT-DSC) and Dynamic Student Classification on Memory Networks (DSCMN) based on RNN for student performance prediction.

Let us mention about Q-matrix refinement problem first. In which we proposed a multi-label skill refinement method that combines three data driven techniques (MinRss, MaxDiff and ALSC) and each of two multi-label classification techniques to refine the skills required for tasks of students. For each possible mapping of task to skill, the classifier has to decide whether the expert's advice is correct, or incorrect. Whereas most algorithms are working at the level of individual mappings, we proposed an approach based on a multi-label classification algorithm that is trained on the mapping of a task to all skills simultaneously. Besides, we also proposed a data filtering method that improves and accelerates refinement algorithm. This work can be seen as improving the performance of Q-matrix refinement algorithms by using a supervised model trained by huge number of synthetic Q-matrices. The results reveal that the proposed method provides better performance than state-of-the-art approaches and it is more significant when Q-matrices contain more errors, and substantially outperform in Q-matrices with larger number of skills. As described in chapter 3, experimental results show that proposed refinement methods outperform than three well-known algorithms in literature and can recover more accurate skill vector in Q-matrix when it only contains noises as same as when we trained.

For knowledge tracing and student performance prediction problem, we proposed two RNN based knowledge tracing models: DKT-DSC and DSCMN, both of these two models can capture student's ability profile at each time interval and assign a student into a distinct group of students with the similar ability dynamically. Student's knowledge is traced based on the ability profile that she possesses in each time interval. Student's ability profile serves as a knowledge transfer mechanism that is evaluated by using student's historical performance records and updated after each time interval in assessment of knowledge mastery. Then performance prediction is performed based on student current knowledge state. Besides, DSCMN utilizes problem difficulty in student's performance prediction, that provide critical information to model and improves predictive performance of the model where abilities of the students are evaluated at each time interval dynamically. We individualize input vector by taking both student's ability and problem difficulty in each step. Instead of using skill level alone, incorporating student's ability and problem difficulty in DKT-DSC and DSCMN yield an improvement in the prediction of student performance. Dynamically assessing student's ability at each time interval and estimating next problem difficulty play the critical role and helps proposed models to capture more variance in the data, leading to more accurate predictions.

In most of previous models, whether a student can solve a problem or not, that only depends on student current knowledge state associated with skills behind that problem. The most important contributions of this research are follows: proposing an efficient student model that does learning transfer from students' previous practices into account in assessment of knowledge mastery; Exploring student performance is not only reflected from mastery of particular skills behind that problem but also related with how much difficulty of problem that student can handle. So we need to consider the problem difficulty when prediction is made for student performance on next problem. The significant gain obtained by our proposed models can make a difference in current aspects of knowledge tracing and student performance prediction by considering transferred learning, problem difficulty and knowledge mastery.

6.2 Limitations and Future Works

For Q-matrix refinement problem, we only dealt with static Q-matrix (which presents required skills of the tasks by using binary values). In the future, we are going to investigate refinement methods with various optimization techniques for providing improvement in other Q-matrices with larger number of skills in the datatsets. It will be also interesting to see how this method performs when multiple perturbation are injected in training data, and dealing with non-binary Q-matrices, and skills are varying over time. The another biggest challenge is finding out micro level influences over these skills. We look further into refinement of Q-matrix with dynamic data where student ability is changing over time.

In aspect of dealing with knowledge tracing, we are only able to use public datasets from ASSISTment and Cognitive Tutor tutoring systems in our experiments. Due to unknown detailed problem contents and randomized order of problems to each student, this limits my dissertation that does not thoroughly investigated of how students' knowledge would improve if students are assigned with structured order of problems and which critical factors make the student fail to answer the problem after achieving mastery (for example: misconception or forgetting). An additional limitation is that I have only investigated topics in algebra, but it can also be used our proposed models across other domains as an approximation of new problems.

We will adapt this model for problems with multiple prerequisite skills in the system and investigate the micro level factors of why students fail to answer the problem under different knowledge states by justifying problem difficulty and controlling various treatments. Then we will try to improve our models towards interpretable models based on transferred learning, knowledge mastery and problem difficulty.

Finally, one important future work is to provide next problem recommendation, that should be done according to the knowledge mastery level and the ability that student possesses and problem difficulty that she can handle in zone of proximal development and improve their knowledge gain under adaptive learning tracks where in our research we assumed that the space is equally important.

REFERENCES

[1] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," 1999.

[2] B. S. Bloom, "The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring," *Educational researcher*, vol. 13, no. 6, pp. 4–16, 1984.

[3] R. S. Baker, "Stupid tutoring systems, intelligent humans," *International Journal of Artificial Intelligence in Education*, vol. 26, no. 2, pp. 600–614, 2016.

[4] J.-P. Doignon and J.-C. Falmagne, "Spaces for the assessment of knowledge," *International Journal of Man-Machine Studies*, vol. 23, pp. 175–196, 1985.

[5] J. Templin, R. A. Henson *et al.*, *Diagnostic measurement: Theory, methods, and applications.* Guilford Press, 2010.

[6] J. L. Templin and R. A. Henson, "Measurement of psychological disorders using cognitive diagnosis models." *Psychological methods*, vol. 11, no. 3, p. 287, 2006.

[7] C.-Y. Chiu, "Statistical refinement of the q-matrix in cognitive diagnosis," *Applied Psychological Measurement*, vol. 37, no. 8, pp. 598–618, 2013.

[8] J. De La Torre, "An empirically based method of q-matrix validation for the DINA model: Development and applications," *Journal of educational measurement*, vol. 45, no. 4, pp. 343–362, 2008.

[9] ——, "DINA model and parameter estimation: A didactic," *Journal of Educational and Behavioral Statistics*, vol. 34, no. 1, pp. 115–130, 2009.

[10] M. C. Desmarais and R. Naceur, "A matrix factorization method for mapping items to skills and for enhancing expert-based Q-matrices," in *Artificial Intelligence in Education.* Springer, 2013, pp. 441–450.

[11] K. R. Koedinger, A. T. Corbett, and C. Perfetti, "The knowledge-learning-instruction framework: Bridging the science-practice chasm to enhance robust student learning," *Cognitive science*, vol. 36, no. 5, pp. 757–798, 2012.

[12] K. K. Tatsuoka, "Rule space: An approach for dealing with misconceptions based on item response theory," *Journal of educational measurement*, vol. 20, no. 4, pp. 345–354, 1983.

[13] R. A. Henson, J. L. Templin, and J. T. Willse, "Defining a family of cognitive diagnosis models using log-linear models with latent variables," *Psychometrika*, vol. 74, no. 2, pp. 191–210, 2009.

[14] A. A. Rupp and J. Templin, "The effects of q-matrix misspecification on parameter estimates and classification accuracy in the DINA model," *Educational and Psychological Measurement*, vol. 68, no. 1, pp. 78–96, 2008.

[15] M. J. Madison and L. P. Bradshaw, "The effects of q-matrix design on classification accuracy in the log-linear cognitive diagnosis model," *Educational and Psychological Measurement*, vol. 75, no. 3, pp. 491–511, 2015.

[16] T. Barnes, "Novel derivation and application of skill matrices: The Q-matrix method," *Handbook on educational data mining*, pp. 159–172, 2010.

[17] J. Nižnan, R. Pelánek, and J. Řihák, "Mapping problems to skills combining expert opinion and student data," in *Mathematical and Engineering Methods in Computer Science.* Springer, 2014, pp. 113–124.

[18] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User modeling and user-adapted interaction*, vol. 4, no. 4, pp. 253–278, 1994.

[19] R. S. Baker and K. Yacef, "The state of educational data mining in 2009: A review and future visions," *JEDM-Journal of Educational Data Mining*, vol. 1, no. 1, pp. 3–17, 2009.

[20] J. E. Beck and Y. Gong, "Wheel-spinning: Students who fail to master a skill," in *International Conference on Artificial Intelligence in Education.* Springer, 2013, pp. 431–440.

[21] M. Feng, N. Heffernan, and K. Koedinger, "Addressing the assessment challenge with an online system that tutors as it assesses," *User Modeling and User-Adapted Interaction*, vol. 19, no. 3, pp. 243–266, 2009.

[22] H. Wan and J. B. Beck, "Considering the influence of prerequisite performance on wheel spinning." *International Educational Data Mining Society*, 2015.

[23] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," in *Advances in Neural Information Processing Systems*, 2015, pp. 505–513.

[24] K. R. Koedinger, R. S. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper, "A data repository for the edm community: The pslc datashop," *Handbook of educational data mining*, vol. 43, pp. 43–56, 2010.

[25] C.-Y. Chiu and J. Douglas, "A nonparametric approach to cognitive diagnosis by proximity to ideal response patterns," *Journal of Classification*, vol. 30, no. 2, pp. 225–250, 2013.

[26] Z. A. Pardos and N. T. Heffernan, "Modeling individualization in a bayesian networks implementation of knowledge tracing," in *International Conference on User Modeling, Adaptation, and Personalization.* Springer, 2010, pp. 255–266.

[27] Z. Pardos and N. Heffernan, "Kt-idem: introducing item difficulty to the knowledge tracing model," *User Modeling, Adaption and Personalization*, pp. 243–254, 2011.

[28] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning*, 2013, pp. 1310–1318.

[29] M. C. Desmarais, "Mapping question items to skills with non-negative matrix factorization," *ACM KDD-Explorations*, vol. 13, no. 2, pp. 30–36, 2011.

[30] E. H. Haertel, "Using restricted latent class models to map the skill structure of achievement items," *Journal of Educational Measurement*, pp. 301–321, 1989.

[31] E. Maris, "Psychometric latent response models," *Psychometrika*, vol. 60, no. 4, pp. 523–547, 1995.

[32] ——, "Estimating multiple classification latent class models," *Psychometrika*, vol. 64, no. 2, pp. 187–212, 1999.

[33] A. Rupp, "The answer is in the question: A guide for investigating the theoretical potentials and practical limitations of cognitive psychometric models," *International Journal of Testing*, vol. 7, pp. 95–125, 2007.

[34] A. A. Rupp and R. J. Mislevy, "Cognitive foundations of structured item response models," *Cognitive diagnostic assessment for education: Theories and applications*, pp. 205–240, 2007.

[35] X. Xu and M. Davier, "Fitting the structured general diagnostic model to naep data," *ETS Research Report Series*, vol. 2008, no. 1, pp. i–18, 2008.

[36] S. Wang and J. Douglas, "Consistency of nonparametric classification in cognitive diagnosis," *Psychometrika*, vol. 80, no. 1, pp. 85–100, 2015.

[37] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[38] M. C. Desmarais, "Mapping question items to skills with non-negative matrix factorization," *ACM SIGKDD Explorations Newsletter*, vol. 13, no. 2, pp. 30–36, 2012.

[39] M. Desmarais, B. Beheshti, and P. Xu, "The refinement of a q-matrix: Assessing methods to validate tasks to skills mapping," in *Educational Data Mining 2014*, 2014.

[40] M. C. Desmarais, P. Xu, and B. Beheshti, "Combining techniques to refine item to skills q-matrices with a partition tree," in *Educational Data Mining 2015*, 2015.

[41] Y. Sun, S. Ye, S. Inoue, and Y. Sun, "Alternating recursive method for q-matrix learning," in *Educational Data Mining 2014*, 2014.

[42] G. Durand, N. Belacel, and C. Goutte, "Evaluation of expert-based q-matrices predictive quality in matrix factorization models," in *Design for teaching and learning in a networked world.* Springer, 2015, pp. 56–69.

[43] J. S. Brown and R. R. Burton, "Diagnostic models for procedural bugs in basic mathematical skills," *Cognitive science*, vol. 2, no. 2, pp. 155–192, 1978.

[44] R. Nkambou, R. Mizoguchi, and J. Bourdeau, *Advances in intelligent tutoring systems.* Springer Science & Business Media, 2010, vol. 308.

[45] M. C. Polson and J. J. Richardson, *Foundations of intelligent tutoring systems.* Psychology Press, 2013.

[46] M. C. Desmarais and R. S. Baker, "A review of recent advances in learner and skill modeling in intelligent learning environments," *User Modeling and User-Adapted Interaction*, vol. 22, no. 1-2, pp. 9–38, 2012.

[47] K. H. Wilson, Y. Karklin, B. Han, and C. Ekanadham, "Back to the basics: Bayesian extensions of irt outperform neural networks for proficiency estimation," *arXiv preprint arXiv:1604.02336*, 2016.

[48] W. J. van der Linden and R. K. Hambleton, *Handbook of modern item response theory.* Springer Science & Business Media, 2013.

[49] J. González-Brenes, Y. Huang, and P. Brusilovsky, "General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge," in *The 7th International Conference on Educational Data Mining.* University of Pittsburgh, 2014, pp. 84–91.

[50] C. Ekanadham and Y. Karklin, "T-skirt: Online estimation of student proficiency in an adaptive learning system," *arXiv preprint arXiv:1702.04282*, 2017.

[51] R. K. Hambleton, H. Swaminathan, and H. J. Rogers, *Fundamentals of item response theory.* Sage, 1991, vol. 2.

[52] F. B. Baker and S.-H. Kim, *Item response theory: Parameter estimation techniques.* CRC Press, 2004.

[53] M. D. Reckase and R. L. McKinley, "The discriminating power of items that measure more than one dimension," *Applied psychological measurement*, vol. 15, no. 4, pp. 361–373, 1991.

[54] D. C. Briggs and M. Wilson, "An introduction to multidimensional measurement using rasch models," 2003.

[55] R. S. d Baker, A. T. Corbett, and V. Aleven, "More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing," in *International Conference on Intelligent Tutoring Systems.* Springer, 2008, pp. 406–415.

[56] H. Cen, K. Koedinger, and B. Junker, "Learning factors analysis–a general method for cognitive model evaluation and improvement," in *International Conference on Intelligent Tutoring Systems.* Springer, 2006, pp. 164–175.

[57] P. I. J. Pavlik, H. Cen, and K. R. Koedinger, "Performance factors analysis–a new alternative to knowledge tracing." *Online Submission*, 2009.

[58] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[59] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[60] M. X. Chen, O. Firat, A. Bapna, M. Johnson, W. Macherey, G. Foster, L. Jones, N. Parmar, M. Schuster, Z. Chen *et al.*, "The best of both worlds: Combining recent advances in neural machine translation," *arXiv preprint arXiv:1804.09849*, 2018.

[61] S. Edunov, M. Ott, M. Auli, and D. Grangier, "Understanding back-translation at scale," *arXiv preprint arXiv:1808.09381*, 2018.

[62] B. Dhingra, H. Liu, Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Gated-attention readers for text comprehension," *arXiv preprint arXiv:1606.01549*, 2016.

[63] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[64] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[65] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[66] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[67] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[68] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of cnn and rnn for natural language processing," *arXiv preprint arXiv:1702.01923*, 2017.

[69] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," in *Advances in neural information processing systems*, 2015, pp. 2440–2448.

[70] J. Zhang, X. Shi, I. King, and D.-Y. Yeung, "Dynamic key-value memory networks for knowledge tracing," in *Proceedings of the 26th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 2017, pp. 765–774.

[71] S. Minn, M. C. Desmarais, and S. Fu, "Refinement of a q-matrix with an ensemble technique based on multi-label classification algorithms," in *European Conference on Technology Enhanced Learning.* Springer, 2016, pp. 165–178.

[72] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine learning*, vol. 85, no. 3, pp. 333–359, 2011.

[73] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-labelsets for multi-label classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 7, pp. 1079–1089, 2011.

[74] A. Robitzsch, T. Kiefer, A. C. George, and A. Uenlue, *CDM: Cognitive Diagnosis Modeling*, 2015, r package version 4.5-0. [Online]. Available: http://CRAN.R-project.org/package=CDM

[75] I. V. Grigorios Tsoumakas, Ioannis Katakis, *Data Mining and Knowledge Discovery Handbook*. Springer, 2010.

[76] P. Xu and M. C. Desmarais, "Boosted decision tree for q-matrix refinement." *International Educational Data Mining Society*, 2016.

[77] S. Minn, Y. Yu, M. C. Desmarais, F. Zhu, and J. J. Vie, "Deep knowledge tracing and dynamic student classification for knowledge tracing," *IEEE International Conference on Data Mining*, 2018.

[78] P. I. Pavlik, M. Yudelson, and K. R. Koedinger, "Using contextual factors analysis to explain transfer of least common multiple skills," in *International Conference on Artificial Intelligence in Education*. Springer, 2011, pp. 256–263.

[79] M. Yudelson, P. I. Pavlik, and K. R. Koedinger, "Towards better understanding of transfer in cognitive models of practice," in *EDM*, 2011, pp. 373–374.

[80] R. S. d Baker, S. M. Gowda, and A. T. Corbett, "Towards predicting future transfer of learning," in *International Conference on Artificial Intelligence in Education*. Springer, 2011, pp. 23–30.

[81] P. Nedungadi and M. Remya, "Predicting students' performance on intelligent tutoring system—personalized clustered BKT (PC-BKT) model," in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*. IEEE, 2014, pp. 1–6.

[82] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[83] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[84] K. M. Tarwani, "Swathi edem,"survey on recurrent neural network in natural language processing"," *International Journal of Engineering Trends and Technology (IJETT)*, vol. 48, no. 6, 2017.

[85] A. Vinciarelli and J. Luettin, “A new normalization technique for cursive handwritten words,” *Pattern recognition letters*, vol. 22, no. 9, pp. 1043–1050, 2001.

[86] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[87] A. Merceron and K. Yacef, “Clustering students to help evaluate learning,” in *Technology Enhanced Learning.* Springer, 2005, pp. 31–42.

[88] S. Trivedi, Z. A. Pardos, and N. T. Heffernan, “Clustering students to generate an ensemble to improve standard test score predictions,” in *International Conference on Artificial Intelligence in Education.* Springer, 2011, pp. 377–384.

[89] J.-J. Vie and H. Kashima, “Knowledge tracing machines: Factorization machines for knowledge tracing,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 750–757.

[90] J. MacQueen *et al.*, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA, 1967, pp. 281–297.

[91] G. Ball and I. Hall Dj, “A novel method of data analysis and pattern classification. isodata, a novel method of data analysis and pattern classification. tch. report 5ri, project 5533,” 1965.

[92] A. Corbett, “Cognitive computer tutors: Solving the two-sigma problem,” *User Modeling 2001*, pp. 137–147, 2001.

[93] L. Razzaq, M. Feng, G. Nuzzo-Jones, N. Heffernan, K. Koedinger, B. Junker, S. Ritter, A. Knight, C. Aniszczyk, S. Choksey *et al.*, “The assistment project: Blending assessment and assisting,” in *Proceedings of the 12th Annual Conference on Artificial Intelligence in Education*, 2005, pp. 555–562.

[94] X. Xiong, S. Zhao, E. Van Inwegen, and J. Beck, “Going deeper with deep knowledge tracing.” in *EDM*, 2016, pp. 545–550.

[95] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[96] S. Minn, M. C. Desmarais, F. Zhu, X. Jing, and W. Jianzong, "Dynamic student classiffication on memory networks for knowledge tracing," *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2019.

[97] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.

[98] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[99] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, p. 471, 2016.

[100] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International conference on machine learning*, 2016, pp. 1842–1850.