MSc thesis

Master's Programme in Computer Science

Modeling knowledge states in language learning

Anh-Duc Vu

June 8, 2020

Faculty of Science

University of Helsinki

Supervisor(s)

Prof. R. Yangarber

Examiner(s)

Prof. R. Yangarber and Prof. M. Koivisto

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki,Finland

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Koulutusohjelma — Utbildningsprogram — Study programme Master's Programme in Computer Science

Tekijä — Författare — Author

Anh-Duc Vu

Työn nimi — Arbetets titel — Title

Modeling knowledge states in language learning

Ohjaajat — Handledare — Supervisors

Prof. R. Yangarber

Työn laji — Arbetets art — Level MSc thesis	Aika — Datum — Month and year June 8, 2020	Sivumäärä — Sidoantal — Number of pages 53 pages, 6 appendice pages

Tiivistelmä — Referat — Abstract

Artificial intelligence (AI) is being increasingly applied in the field of intelligent tutoring systems (ITS). Knowledge space theory (KST) aims to model the main features of the process of learning new skills. Two basic components of ITS are the domain model and the student model. The student model provides an estimation of the state of the student's knowledge or proficiency, based on the student's performance on exercises. The domain model provides a model of relations between the concepts/skills in the domain. To learn the student model from data, some ITSs use the Bayesian Knowledge Tracing (BKT) algorithm, which is based on hidden Markov models (HMM).

This thesis investigates the applicability of KST to constructing these models. The contribution of the thesis is twofold. Firstly, we learn the student model by a modified BKT algorithm, which models forgetting of skills (which the standard BKT model does not do). We build one BKT model for each concept. However, rather than treating a single question as a step in the HMM, we treat an entire practice session as one step, on which the student receives a score between 0 and 1, which we assume to be normally distributed. Secondly, we propose algorithms to learn the "surmise" graph—the prerequisite relation between concepts—from "mastery data," estimated by the student model. The mastery data tells us the knowledge state of a student on a given concept. The learned graph is a representation of the knowledge domain. We use the student model to track the advancement of students, and use the domain model to propose the optimal study plan for students based on their current proficiency and targets of study.

ACM Computing Classification System (CCS)
Computing methodologies → Machine learning → Machine learning approaches → Learning in probabilistic graphical models → Bayesian network models
Computing methodologies → Artificial intelligence → Knowledge representation and reasoning → Probabilistic reasoning
Theory of computation → Theory and algorithms for application domains → Machine learning theory → Bayesian analysis

Avainsanat — Nyckelord — Keywords

Knowledge space theory, Bayesian knowledge tracing, GS algorithm, Bayesian network learning

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Algorithms, Data Analytics and Machine Learning subprogramme

Contents

1 Introduction

1.1 Motivation

The quality education has cultivated dramatic awareness with a view to optimizing education and heading to a sustainable development in the future. The importance of quality education has been acknowledged by several researchers forcing them to seek for multiple efficient ways. In the last few years, there has experimented a global evolving trend over different study domains of utilizing Artificial intelligence or Machine learning in gauging and optimizing the studying process.

One idea of applying Machine learning in teaching is intelligent tutoring system (ITS) which can support student to improve their learning curve. In order to achieve this objective, researchers firstly have a consideration of learning the study structure of the study domain. This idea is based on an assumption that we can split the study domain into several components, and understand the interrelationship among them. In terms of this idea, the Knowledge space theory (KST) [1, 7, 9] provides us with a view on decomposition a knowledge domain into components and connect them as a graph of knowledge states to exploit the optimal learning path and assess the competencies of learners. In a certain domain of knowledge, the learning space is a finite set of items which could be learning objects; and all possible collections of items from the learning space are called the knowledge states. For example in a knowledge domain S, which contains *summation*, *subtraction*, *division*, and *multiplication*; the corresponding knowledge states of this knowledge domain is presented in Figure 1.1.

This theory has been successfully applied in many learning domains, such as mathematics and psychology. For instance, Goncalves presented the ideas of KST in mathematics; while Spoto et al. attempted to connect KST with psychological assessment in research [25]. However, there have not been many attempts to apply KST in the language fields. Learning logical subjects like mathematics bears a difference to linguistic learning. In linguistic field, learning concept witnesses more abstractions and complications.

1.2 Objectives

The main goal of this thesis is applying the KST to the domain of language learning to learn the student model and the domain model, with three major objectives:

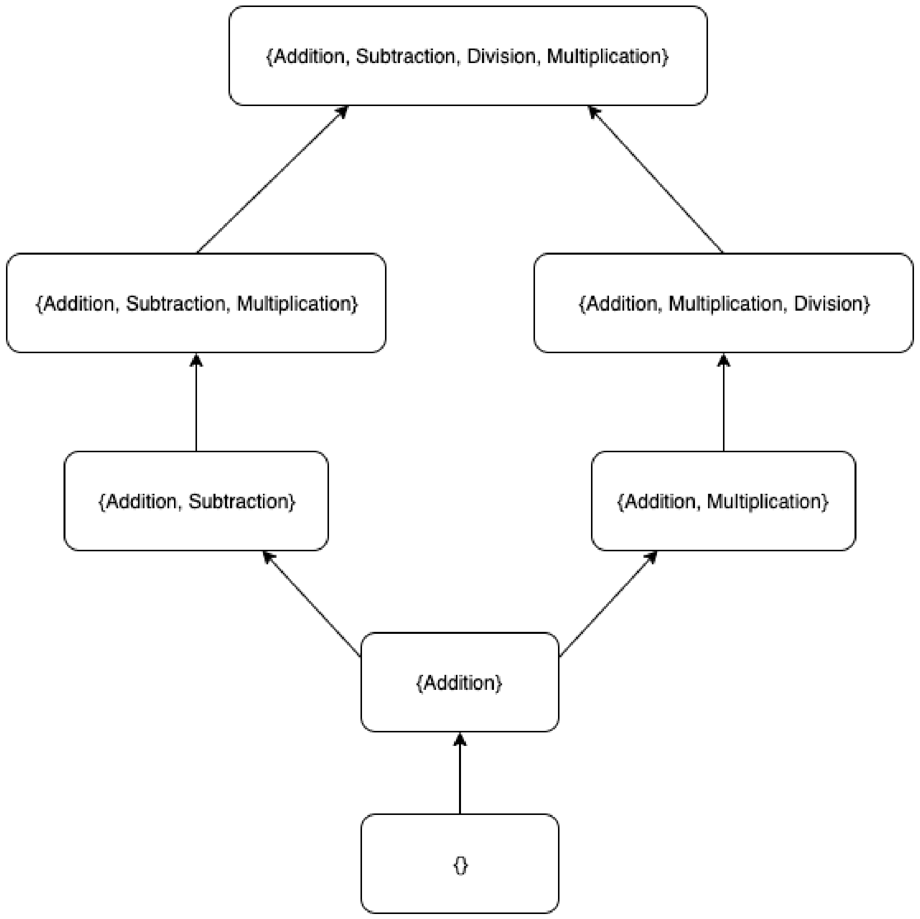
{Addition, Subtraction, Division, Multiplication}
{Addition, Subtraction, Multiplication}
{Addition, Multiplication, Division}
{Addition, Subtraction}
{Addition, Multiplication}
{Addition}
{}

Figure 1.1: Example of knowledge states of knowledge domain S

1. Tracking the advancement of students: the student model takes the sequence of performance of a student A on a concept/skill B to estimate the proficiency level of A on B, and to predict the performance of A on B in the future.

2. Measuring the difficulty of concepts: the student model uses the performance of all students to learn the correlation between mastery states, mean scores, and score variances, which gives an estimate of the difficulty of the concepts.

3. Proposing a learning plan for students: the domain model learns the "surmise" relation—the prerequisite relation—between concepts in the domain of language learning, to estimate the most suitable learning plan for students based on the history of their performance.

To learn the student models to estimate student's proficiency state, we use two algorithms: BKT and Threshold-based (Section 3.1). The learnt model is able to estimate the proficiency state (mastery state) of students on concepts based on their performance. The state of one student on one concept is labeled "learnt" or "not learnt". Estimating the state means estimating whether a student has mastered one concept. The student model can also learn the mean score and variance of score of each concept, which mean the student model can help us measure the hardness of concepts.

To learn the domain model, we construct a surmise graph of the domain as a Bayesian network with each node manifesting for one corresponding concept. Bayesian network is a probabilistic graphical model, which encodes the conditional dependencies among a set of variables in a directed acyclic graph. In the domain of learning Russian, the items are called concepts such as "Negative construction", "I conjugation", "Negative sentences", ... We apply five different algorithms (Section 5) to learn the structure of surmise graph from data. The surmise graph encodes the surmise relationship among concepts in topological order. The graph is able to suggest the optimal learning curve for students based on their current proficiency state and their target of study.

The domain model is able to suggest the optimal learning curve given the proficiency states of students of all concepts. The student is able to estimate the proficiency states of students of all concepts as the input of the domain model. By the domain model and student model, we can build a tutoring systems, which has the abilities to track the advancement of students, predict future performance of students, measure the hardness of concepts (feature of the student model), and provide the optimal learning curve for students (feature of the domain model).

1.3 Structure of the thesis

This thesis is organized as follows. Section 2 is the outlines of the data. In Section 3, the background review for all solutions and ideas used in this thesis are denoted. Section 4 presents related works with this paper. It is followed by a description of our solutions in Section 5. The empirical experiments of all solutions and methodologies are then presented in Section 6. Section 7 is the place for discussion of current obstacles and future ideas for this topic. The last part of this thesis, Section 8, contains the conclusion along with the summary of the whole paper and the acknowledgement. The final concept graph along with the concept list and their information can be found on appendix pages.

2 Data

2.1 Data Description

All data for this thesis are collected from the Revita system which is a language learning platform developed by the cooperation between Computer Science department, Modern Languages department of University of Helsinki and Foreign Languages and Literatures department of University of Milan [12, 13]. Revita is a part of ICALL, Intelligent Computer-Aided Language Learning created to support language education over many universities and organizations in the world.

There are about 5000 questions distributed over 88 concepts and 6 levels of the Common European Framework of Reference for Languages (CEFR) system which are A1, A2, B1, B2, C1, and C2. Those concepts along with their CEFR information are provided by linguistic experts in Russian, for instance, teachers from Russian departments. The samples of student performances are collected directly from real Russian learners on Revita system. There are about 500 students having done one or many tests and approximately 4000 test sessions in total. In each test session, the time is limited and all concepts are asked with a randomized number of questions sampled from the corresponding question bank; however, it is unnecessary that students answer all given questions in a test session. There are approximately 300 questions in each test in such way that each question belongs to only one concept and all concept is presented in each test session with at least one question. The timestamps, student identity numbers, and question CEFR levels are also recorded along with all questions in each test session.

2.2 Data Pre-processing

This thesis presents two experiments, the first using the original 88 concepts without taking into account their CEFR levels, and the second using a combination of concepts with CEFR levels. For example, concept "6, Lexicology. Lexical semantics" is distributed over three CEFR levels: B1, B2, and C1. In the second experiment, this concept is considered as 3 different concepts "6B1", "6B2" and "6C1". The total number of these "refined" concepts produced by this combination process is 127. Thus we have two experiments:

1. with the set of original 88 concepts, named *concepts without CEFR*, and

2. with the set of 127 fragmented concepts, named *concepts with CEFR*.

The methodology of our solution contains three main steps. Firstly, we attempt to learn the models tracking student's advancement. In the second step, we use the models trained after the first step to estimate the corresponding advancement states. Lastly, we recover the interrelationship between concepts based on the estimated advancement data from the second step. To do so, the accumulated examples are split into three disjoint datasets by grouping the students. The three disjoint datasets are:

- the "student" set (188 students, 1265 test sessions) used for training the student model,
- the "domain" set (189 students, 1489 test sessions) for constructing the domain model, and
- the "evaluation" set (189 students, 1255 test sessions) for evaluating.

Data of the first student group are used for step 1 in which the student models are built. The student models learnt from the first group are utilized to estimate state data of the second student group. Then, the state data of the second group is used to learn the concept graph (the knowledge structure). The third group is the evaluation data-set for the concept graph.

2.3 Data Table Format

The samples after pre-processing are accumulated by test sessions. Each session is identified by test session ID and contains not only the score of all involved concepts but also timestamp and student number. The score of each concept is from 0.0 to 1.0 which presents for the percentage of correct answers. If there is no response on one concept, the score information is marked as null value. The illustration of data table is displayed in Table 1.

Since we have two different configurations for concepts in domain of learning Russian which are concepts without CEFR and concepts with CEFR as mentioned above in Section 2.2, there are two data tables corresponding to two concept configurations.

Test session	Student ID	Timestamp	Concept 1	Concept 2	...	Concept N
001	01	1586203202	0.2	0.6	...	null
002	01	1586203206	0.6	null	...	0.9
...	...	...	...	...	...	...
999	99	1586203200	1.0	0.8	...	0.7

Table 2.1: Example of data format

3 Background

3.1 Knowledge space theory

Knowledge space theory (KST) is a framework for identifying the learning structure of a knowledge domain. This theory was invented by Doignon & Falmagne in 1985 [9] and then improved by Albert & Lukas in 1999 [1]. The theory assumes that in a certain study field, there is a finite set Q of problem types, skills or concepts representing some learning objects or levels in that knowledge domain; and set K which contains some subsets of Q. For illustration, a student needs to master basic knowledge of number in order to understand some mathematical formulas and be able to handle all basic operations such as addition, subtraction, multiplication and division in a simple equation. Therefore, the study domain of solving a simple equation could be decomposed into two concepts: numbers and operations. However, there are some subsets of Q are not knowledge states (not belong to K) since these subsets are not likely to exist. For example, one student cannot understand gradient without any knowledge about number and operations, hence there is likely no knowledge state contains gradient without numbers and operations.

The collection K of subsets of Q which includes the empty set and the Q itself is considered as the set of knowledge states of the study domain. All elements of K are related to each other in a topological order and together they construct a Bayesian network (Section 3.2) that the root node stands for the empty knowledge state and the last node stands for a knowledge state which contains all skills in Q. This Bayesian network is named the knowledge state graph describing the study domain. For illustration, Figure 3.1 presents an example graph of a knowledge structure with $Q = \{X, Y, Z\}$. In KST, set K and set Q define the domain model of a study field which is the first objective of an ITS.

To discover Q and K, there are some definitions that have been denoted by Doignon and Falmagne in paper "Knowledge space and learning space" [8]. Firstly, the discriminative definition that for all concepts X and Y, the two collections of all knowledge states containing X or Y are identical if and only if X is equal Y. The second definition relates to learning smoothness that if a knowledge state A contains knowledge state B then there is at least one topological path to reach A from B by learning concept one by one. The third definition states learning consistency which implies that understanding more concepts does not limit learning new things. For example, if there is a state A contains state B and state $B \cup \{q\}$

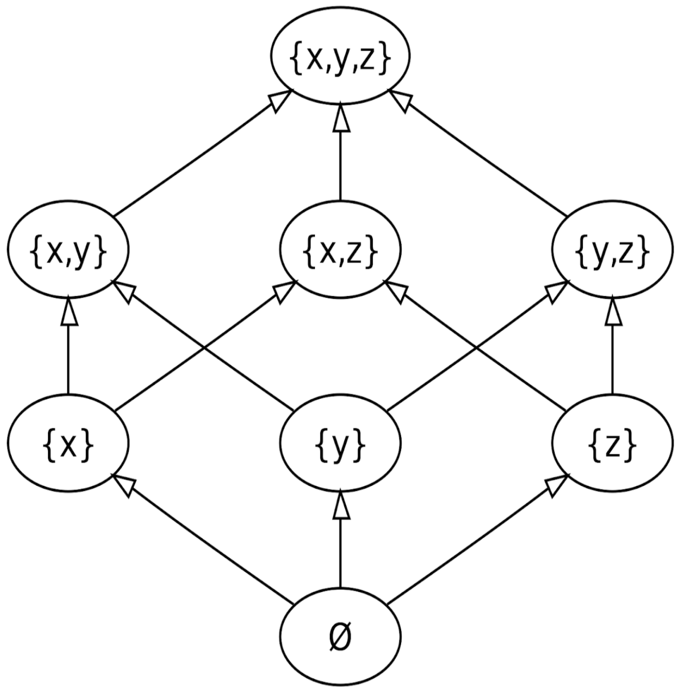

Figure 3.1: Knowledge structure graph with Q = {X, Y, Z} [7]

where q is a concept; then $A \cup \{q\}$ is also a state. The last definition, which is named "antimatroid", refers that every state in K except the empty state is able to be downgraded. A state A is able to be downgraded if A contains some concepts q that $A \setminus \{q\}$ is also a state.

There are many ideas to extract K or the knowledge state graph from students' test results such as the maximum residual method [19] or the distributed skill maps [26]. These methods introduce a term named *response pattern*. A response pattern is a unique pattern of answers from students for one set of questions. One pattern can be responded by one or many students; therefore, the frequency of each pattern play is commonly utilized to infer the knowledge state graph for a group of questions or concepts. The maximum residual method takes the frequencies of student's response patterns into account in order to find the most likely K, while the distributed skill maps solution tries to construct a skill map from response pattern and then derive the most probable knowledge state graph. The common point between those methods is trying to extract the K and knowledge state graph from response patterns. However, we don't know how many subsets of Q belong to the set K; therefore the size of K is not known as well. In the worst case, the size of K could be 2^n where n is the number of concepts. In order to reduce the size of the graph, rather than discovering the knowledge state graph, we try to discover the *surmise graph* of concepts, which we define next.

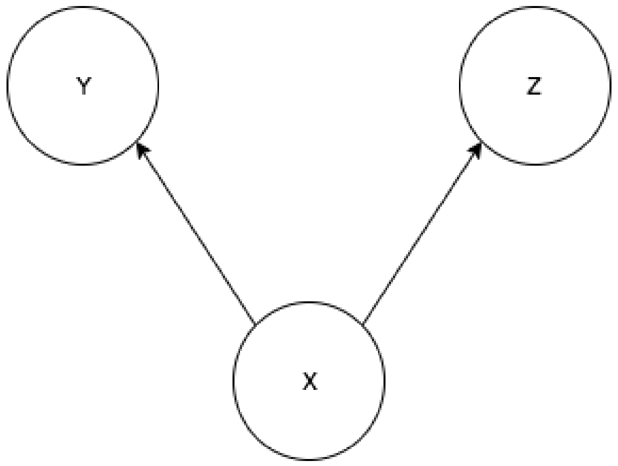

Figure 3.2: Example surmise graph given Q = {X, Y, Z}

The surmise graph of concepts is an alternative way to present the interrelationship among concepts in Q while ensuring the above definitions in KST, and it is also impossible to transform to the knowledge state graph. The surmise graph is a Bayesian network (Section 3.2) as well with each node representing one concept. The direction edges between these nodes standing for their surmise relationship. Figure 3.2 is one simple instance of concept graph and Figure 3.3 is the corresponding knowledge state graph for Figure 3.2.

The target for the domain model is alternative to extracting the most probable surmise graph of concepts from students' performance examples. We propose both theory-driven solution and data-driven solution. The theory-driven solution is based on prior theories that determines the topological surmise order of concepts. The data-driven methods such as the item tree analysis (Schrepp [22]) and Grow-shrink Markov blanket algorithm (Margaritis [14]) are capable to learn the statistical correlation and conditional probabilities among concepts from the given performance data. The detail description of both theory-driven and data-driven ideas are in the following sections on Bayesian networks.

3.2 Bayesian network

Bayesian network is a directed acyclic graph (DAG) that encodes all conditional independencies between a finite number of variables and their probability distribution P graphically as the minimal I-map (Independency-map). An I-map is a DAG, which encodes all independencies of one probability distribution. Pearl [16] defined that a DAG G is guaranteed to be an *I-map* of a probability distribution P if every conditional independence encoded in G is also valid in P. By the minimal I-map, we mean an I-map of P which can be considered as a Bayesian network if and only if that I-map has no edges which can be removed without

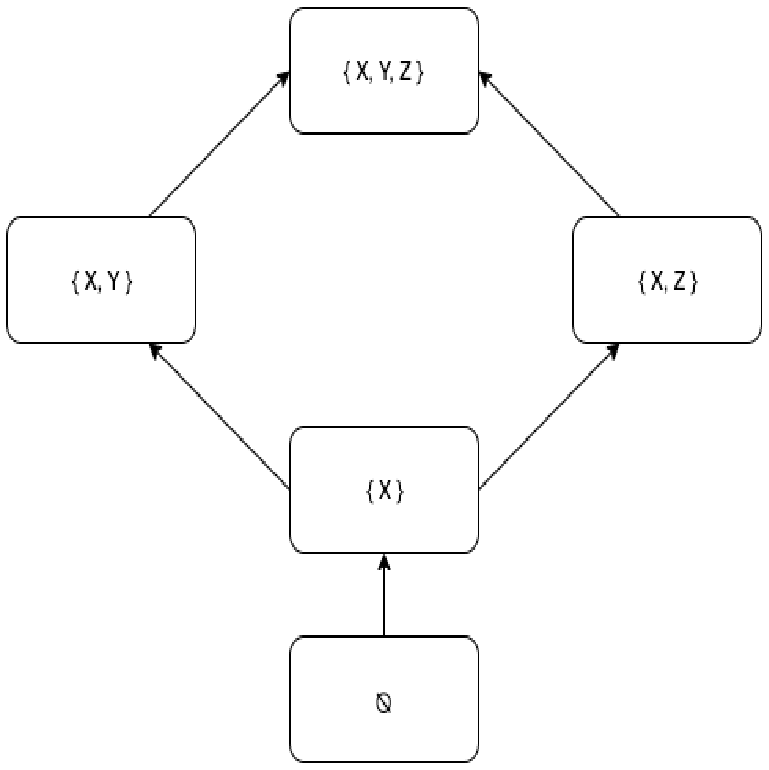

Figure 3.3: The knowledge state graph corresponding to Figure 3.2

invalidating the above rule on the distribution P.

One important note is that an I-map or Bayesian network can only maintain the conditional independencies in P, and the I-map that is valid for both conditional independencies and dependencies is called D-map of the distribution P. However, much research follows the faithfulness assumption that a network G and distribution P are faithful to each other when all conditional independencies in P are entailed by the Markov property on G. We also follow this assumption throughout this thesis. Figure 3.4 is an example of a Bayesian network with the conditional probabilities.

D-separation. D-separation is a criterion to identify the conditional independencies in a DAG. D-separation determines whether two variables X and Y in a DAG are independent given the evidence of several other variables in G, or unconditionally independent. This criterion detects the independencies by checking whether there are any *unblocked* paths (see below) between X and Y. In a DAG G of a probability distribution P, if there is no unblocked path between X and Y, we can say that X and Y are independent in G and P.

Graph G only ensures the conditional independencies of the variables. The dependencies in G are not faithful to the probability distribution P. If there is at least one unblocked path between two variables X and Y in G, then we can say that X and Y are dependent in G.

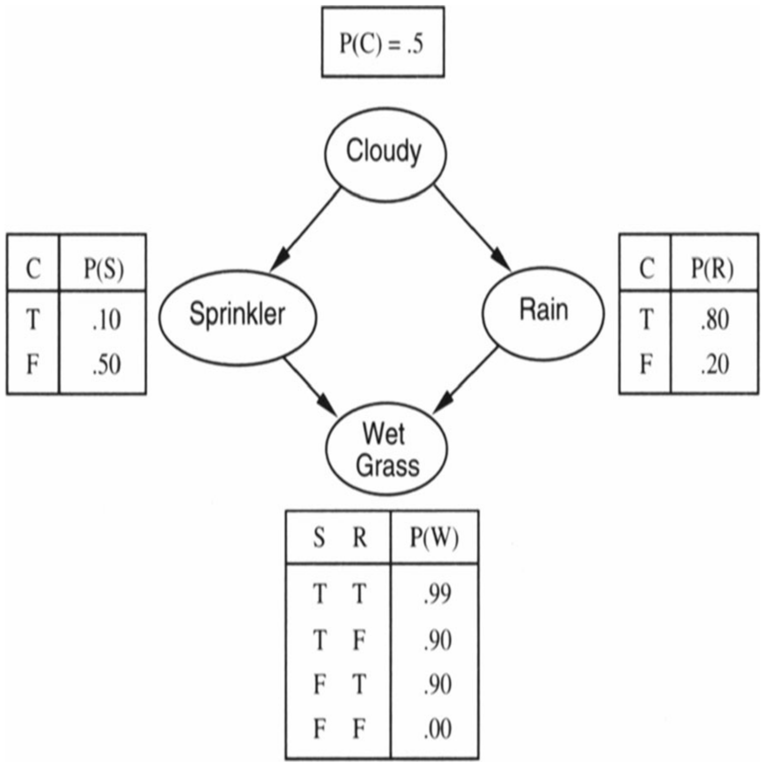

Figure 3.4: Example of Bayesian network [15]

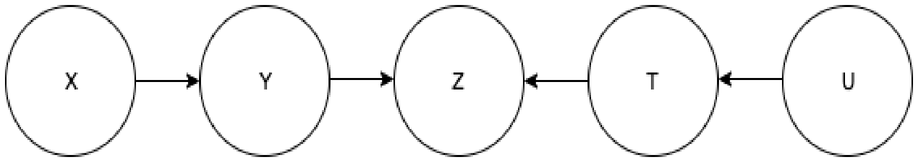

Figure 3.5: d-separation example

Following to the faithfulness assumption that the dependencies in graph G is faithful to P, we also can say that X and Y are dependent in P if there is at least one unblocked path between two variables X and Y in G.

In unconditional cases, the path in a graph is a sequence of edges, disregarding their direction. A path is considered unblocked when there is no “head-to-head” pair of edges with respect to the direction. This rule assumes that the direction of each edge stands for the passing line of information in the network, and any colliders that block the flow of information also separate the dependencies. For instance in Figure 3.5, path $X \rightarrow Y \rightarrow Z$ and $U \rightarrow T \rightarrow Z$ are unblocked but $X \rightarrow Y \rightarrow Z \leftarrow T \leftarrow U$ is blocked by the ”head-to-head” collider of two edges $Y \rightarrow ZandT \rightarrow Z$. Hence, X and U or X and T are not d-connected, while X and Y or X and Z is d-connected, given no conditions, which can be written as $X \perp T$, $X \perp U$, $X \not\perp Y$ and $X \not\perp Z$.

In the conditional case, when there is some observed evidence, we determine whether the

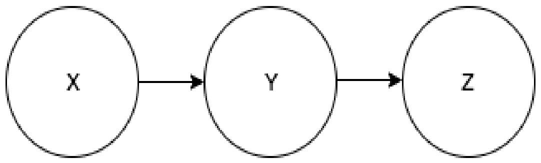

Figure 3.6: Casual chain triplet

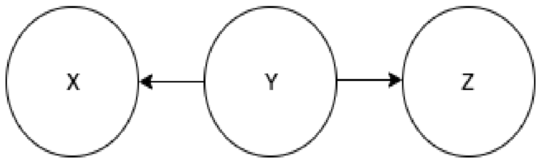

Figure 3.7: Common cause triplet

path is blocked or not blocked by three rules on three types of variable triplets. By a triplet, we mean a set of three variable nodes which are linked directly, disregarding the direction. A path is considered blocked when it contains at least one *inactive* triplet (see below), and a pair of variables is conditionally independent if every path between them is blocked. Three different types of triplets and their corresponding rules are: casual chain, common cause and common effect (v-structures), explained below, Figures 3.6, 3.7, 3.8.

A variable X is conditionally independent of all its non-descendant variables when the values of its parent variables are observed. In terms of the first type, this means if we can observe the middle variable in the causal chain of three variables, then two remaining variables are independent. For instance in Figure 3.6 we have that X and Z are independent given Y or we can say the evidence of Y makes the path from X to Z inactive, written as $X \perp Z \,|\, Y$.

The second rule for the "common cause" triplet implies that two variables are conditionally dependent if the common parent is observed while they are dependent without any conditions. For the triplet in Figure 3.7, we can say that X and Z are d-separated given Y, or the evidence of Y blocks the path from X to Z; we say in this case that the triplet (X, Y, Z) is *deactivated*.

In contrast to two above types, we know that if there is no evidence in the third type of triplet then two variables sharing same child are independent since there are "head-to-head" pair of edges between them. They are conditionally dependent with respect to the evidence of the common child. In the case of Figure 3.8, we can say that X and Z are independent without given Y and dependent with given Y, or the evidence of Y unblocks and makes the path from X to Z active.

Learning Bayesian networks. Bayesian network has been applied in many scientific fields such as biology, psychology, health care and physics, due to its ability to visualize the interrelationship between science concepts and phenomena. In situations where we have

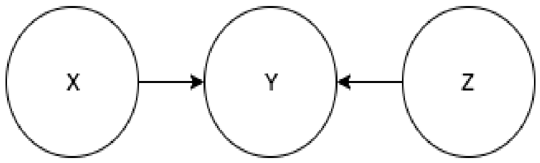

Figure 3.8: Common effect triplet

inherent knowledge about the structure of the network, the parameters to fit the given data can be estimated by calculating the conditional probability among variables. In problems where we do not have a fixed network structure, it is possible to recover the network structure following some prior theories, or we can learn the structure directly from data. However, learning Bayesian networks from data is a challenging problem.

Learning parameters of Bayesian network. We can learn the parameters of the network with a fixed structure by applying the Bayes theorem. There are a wide variety of versions for the mathematical formula of Bayes theorem, but the most popular one is defined as

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

In learning network parameters, we replace A by θ, and B by D and G, where θ stands for the parameters of the Bayesian network, D presents the given observed data, and G is the given graph structure. In case the structure is fixed, G does not play a pivotal role here. Now, the model formula to estimate the Bayesian network parameters is

$$P(\theta|D,G) \;=\; \frac{P(D,G|\theta)\,P(\theta)}{P(D,G)} \tag{3.1}$$

or

$$P(\theta|D) \;=\; \frac{P(D|\theta)\,P(\theta)}{P(D)}. \tag{3.2}$$

In both Formula 3.1 and Formula 3.2, the $P(\theta)$ is the prior, $P(D|\theta)$ is the likelihood , which represents the probability that D occur given θ, and $P(\theta|D)$ is the posterior probability for parameters θ with respect to data D. The objective is to find the best-fit parameters θ to the data D, which is equivalent to maximizing the posterior $P(\theta|D)$. Therefore the probability $P(D)$, could be ignored. We can rewrite the formula as follows:

$$P(\theta|D) \;\propto\; P(D|\theta)\,P(\theta), \tag{3.3}$$

where $\propto$ means "proportional to." To find the parameters of the graph, we find the value of θ, which derives the maximum value of $P(\theta|D)$ or $P(D|\theta)$.

The posterior $P(\theta|D)$ can be maximized by maximum a posteriori estimation (MAP). MAP is useful to avoid overfitting in the situation when we have limited examples. Maximizing $P(\theta|D)$ is equivalent to maximizing $P(D|\theta)\,P(\theta)$ according to Formula 3.3. Maximizing $P(D|\theta)$ can be solved by Maximum Likelihood Estimation (MLE). MLE is equivalent to MAP if the prior $P(\theta)$ is uniform, which means maximizing $P(\theta|D)$ is equivalent to maximizing $P(D|\theta)$.

Learning the structure of Bayesian network. In the literature, there are two basic data-driven solutions to extract the structure of the Bayesian network from data. The first solution is a score-based method [28]. This method assigns a formula to calculate a score for each version of the graph. The formula for the score usually describes the fitting of the graph to the observations, which is presented in Formula (3.4).

$$Score(G, D) \;=\; P(G|D), \tag{3.4}$$

where G represents one version of network structure and D is the data. We can apply Bayes theorem here to expand Formula 3.4) to Formula (3.5.

$$Score(G, D) \;=\; P(G|D) \;=\; \frac{P(D|G)P(G)}{P(D)}. \tag{3.5}$$

In order to find the greatest score, we need to find structure G that maximizes the $P(G|D)$. The denominator $P(D)$ does not depend on G; therefore, the maximization depends on the numerator $P(D|G)P(G)$. In the numerator, the probability distribution of G is our prior knowledge on G, and we assume that $P(G)$ is uniform and can be ignored. Hence, maximizing $P(D|G)$ or $P(G|D)$ results in the optimal score.

The second approach is constraint-based method [28], which constructs the network structure based on the status of conditional independence to learn the relationship among each pair of variables. We need to follow the above faithfulness assumption of Bayesian network for this method. The core component of this idea is the conditional independence test $Ind(X_i; X_j|S)$ to check the independence between X_i and X_j given the evidence S. According to the faithfulness assumption, if $Ind(X_i; X_j|S)$ returns that two variables X_i and X_j are dependent (or independent) given the evidence of S, then X_i and X_j are not d-separated (or d-separated, respectively) given the evidence of S. This method can be exemplified by SGS algorithm Spirtes, Glymour & Scheines [24], or Inductive Causation (IC) algorithm [18]. Both the

SGS algorithm and IC algorithm initializes with a complete non-directed graph, and keeps removing edges following the independence check $Ind(X_i; X_j|S)$, for each pair of variables.

The common problem of the two above structure learning methods is their time complexity. Those methods search for the best fit version of structure on data where the possible number of structures is 2^n, where n is the number of variables in the domain. For instance, the SGS algorithm computes the independencies for each pair of variables X and Y, with respect to all 2^{n-2} subsets of set Q, containing all variables in the domain except the two variables X and Y. Those methods with exponential complexity are impractical in our situation, when our domain is comprised of up to 127 variables (Section 2.2). Therefore, we introduce two alternative methods in the thesis. The first is a version of the score-based method, which is modified to find an approximate answer to save computation time. The second is an independence-based algorithm, called the Grow-shrink Markov blanket algorithm, which exploits the properties of the Markov blanket in a Bayesian network.

Markov blanket. For Bayesian networks in particular and directed acyclic graph in general, the Markov blanket of a node X in the graph is defined as the set of its parents, children and spouses, where the spouses are the nodes that share the one or more same children with X. More detail on Markov blanket can be found in the paper "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference" published by Pearl in 1988 [17]. Figure 3.9 is an example of Markov blanket for variable X. The evidence of X's parents block all paths from all of X's ancestors (except X's parents) to X with respect to the first rule of d-separation, while the second rule of d-separation declares that the evidence of X's parents also blocks all paths from X's siblings, where the siblings are nodes sharing one or more common parents with X. The evidence of X's children blocks all paths from X to X's descendants (except X's children) due to the first rule. The evidence of spouses of X makes all paths from all X's children to all outside nodes of the Markov blanket inactive. Accordingly, we have that for each variable X in the network if we can observe all variables in its Markov blanket then this variable X is independent of all other variables in the graph. In other words, the Markov blanket is all we need to predict the information of that node.

3.3 Grow-shrink Markov blanket algorithm

The Grow-shrink Markov blanket algorithm is described by Margaritis in his doctoral thesis [14]. He indicates that the algorithm focuses on the usage of Markov blanket in reconstructing the network structure. According to the above definition of Markov blanket, we have that all variables (except the spouses) in a Markov blanket of a variable X are the direct neighbors

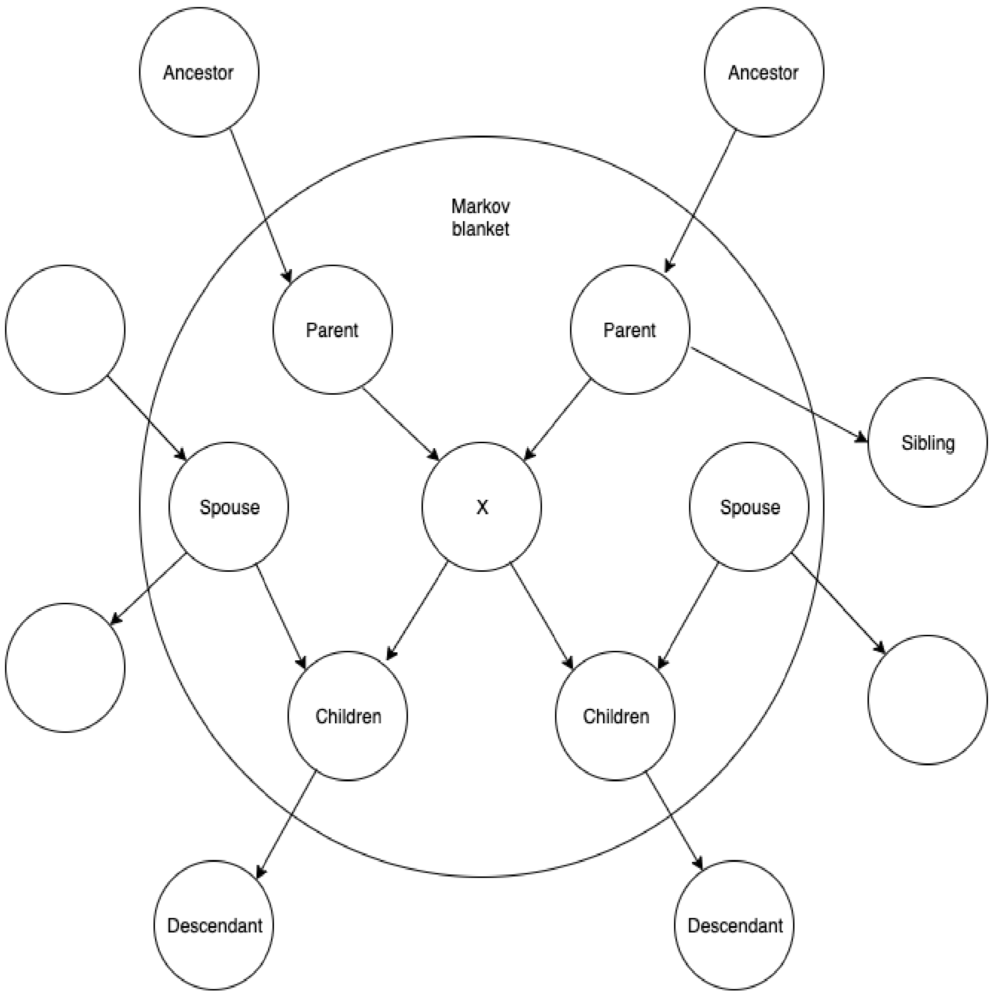

Figure 3.9: Markov blanket of variable X

of X, and the evidence of the Markov blanket of X d-separates X from all other variables in the domain. The algorithm exploits the blanket to find all neighbors for each variable, then estimates the direction for each connection to their neighbors.

Step 1—Markov blanket estimation. First, we need to estimate the Markov blanket for each variable. There are two main phases to do so, which are "Growing phase" and "Shrinking phase". Margaritis [14] defines these phases as the following pseudo-code:

Algorithm 1 Pseudo-code to extract Markov blanket

```
1. Initialization:
S ← ∅
U ← All variables in domain,
X ← considering variable

2. Growing phase:
while $\exists Y \in U - \{X\}$ such as $X \not\perp Y|S$ do
    $S \leftarrow S \cup \{Y\}$
end while

3. Shrinking phase:
while $\exists Y \in S$ such as $X \perp Y|S - \{Y\}$ do
    $S \leftarrow S - \{Y\}$
end while

4. Return S as Markov blanket of X
```

According to the above algorithm, the growing step begins with an empty set S and keeps adding into S the variables in the domain, except the considered variable that is dependent on X with respect to current content of S. This phase simple tries to find a subset S of all variables which does not violate the property of Markov blanket, that a blanket of a variable should isolate it to other variables. However, this first stage does not guarantee that S is the Markov blanket of X; therefore the shrinking phase shrinks the redundant variables of S. The faithfulness assumption declares that any variable Y in blanket of X is dependent on X given the evidence of the blanket without Y, or we can say Y and X is d-connected given $Blanket(X) - \{Y\}, \forall Y \in Blanket(X)$. The shrink phase filters out the redundant variable in S that do not satisfy the above faithfulness assumption, then returns the Markov blanket of variable X.

The time complexity of calculating Markov blanket for one single variable in the worst-case scenario is linear with respect to n and $|D|$, where n is the number of variables in the domain, and $|D|$ is the number of examples in the training data. In the growing phase, the maximum number of iterations that extend the set S without breaking the features of Markov blanket is n. Each iteration needs to check one conditional probability with respect to current content of S which need to go over all input examples. Hence, the growing phase consumes $O(n|D|)$ time complexity. In the shrink phase, the size of set S is always smaller than the number of variables in the domain, so checking all elements in set S needs less than n loops. Similarly to the growing step, the time complexity for each iteration is $O(|D|)$ to compute the conditional probability from examples. By summing up the time of both growing phase and shrinking phase, the time complexity to achieve one Markov blanket is $O(n|D| + n|D|) = O(n|D|)$. Here, the number of examples $|D|$ could be omitted; resulting the time complexity is $O(n)$ which is linear time.

Step 2—Recover network structure. After identifying the Markov blanket for all variables in the domain, Margaritis proposes his Grow-Shrink (GS) algorithm to use the estimated blankets to reconstruct the network structure. For each node X, the GS algorithm firstly removes the spouses contained in the blanket of X, to collect the set of direct neighbors of X. To do so, the algorithm checks one by one variables Y in the Markov blanket to find the spouses of X, all of which are unconditionally independent of X, and dependent on X given the evidence of at least one subset of $Blanket(X) - \{Y\}$. After this step, we have the neighbors of X, but not the direction of the edges to the neighbors. In the next step, the GS algorithm detects the direction of the edges by finding the parent nodes in each neighborhood, in such way that the evidence of X makes them dependent (Section 3.2). For a pair of nodes, Y and Z, in the neighborhood of X, we have three possible configurations for the triplet. Case 1 is v-structure (Y and Z are parents of X). The third rule of d-separation for v-structure in Bayesian network says the evidence of a common child of two variables makes them dependent. Case 2: according to "casual chain" d-separation rule, if Y is parent while Z is a child of X, then they are not dependent given X. Case 3: if Y and Z are both children of X, they are also independent given X.

After learning the parents, children and spouses of each node, we are able to identify the direction for edges in the network; however, there is no guarantee that there is no cycle in the structure due to the noise in sample data. The GS algorithm handles this problem by reversing the edge that belongs to the most cycles, and repeat this until no cycles remain. Lastly, the algorithm directs all edges that have no direction to complete the Bayesian network structure (without creating new cycles). The pseudo-code of GS algorithm [14], shown in

Algorithm 2.

Time complexity. The time complexity to calculate the Markov blanket for one variable is $O(n)$, where n is the number of variables in the domain. Therefore the time complexity to compute Markov blanket for all variables is $O(n^2)$. If b is the length of the biggest Markov blanket, b is always smaller than the number of all variables n. The set $\mathbf{T}$ in each iteration $Y \in \mathbf{B}(X)$ of step 2 has the maximum size b, so there are at most 2^b subsets $\mathbf{S}$ of $\mathbf{T}$. Therefore, the time complexity to compute the set of directed neighbors for all variables is $O(nb2^b)$. The time complexity for step 2 in the worst-case scenario is $O(n^2 2^n)$ with respect to $max(b) = n$. In step 3, the maximum length of the neighbor set of one variable is also b since $\forall X : \mathbf{N}(X) \subset \mathbf{B}(X)$. Similarly to step 2, there are 2^b different combinations of $S \cup \{X\}$, therefore computing step 3 takes $O(nb^2 2^b)$. In order to detect cycles in the structure, the Depth First Search (DFS) traversal algorithm can determine whether or not one edge is in cycle in $O(n + m)$ time complexity where n is the number of vertices and m is the number of edges. Hence, the time complexity to detect all cycles in graph is $O(m(n + m))$, where m is at most n^2. Finally, the algorithm iterates over all edges with $O(m)$ time in the last step; however this step can be done along with DFS traversal. The time complexity of combination of step 4 and 5 is $O(nb(n + m))$. By summing up all the steps, the total time complexity for the whole algorithm is $O(n^2+nb2^b+nb^2 2^b+nb(n+m)) = O(n^2+nb2^b+nb^2 2^b+n^2 b+nbm) = O(nb^2 2^b + n^2 b)$ or $O(n^2 + nb2^b + nb^2 2^b + mn + m^2 + m) = O(n^2 + nb^2 2^b)$. By assuming b is a constant, the time complexity of GS algorithm is $O(n^2)$ which is practical.

The biggest advantage of the GS algorithm is its computational speed, because the number of conditional independence tests required in GS is lower than in the greedy algorithm. This benefit is shown through the time complexity which are $O(n^2)$ and $O(2^n)$ for GS algorithm and greedy algorithm respectively. However, there are some uncertainties about the GS algorithm that the Markov blankets could be incorrect due to the limitations of sample data. The number of training observations is limited, and those observations are not able to completely represent the real interrelationship among variables; this could result in the wrong conclusion of conditional independence tests. The step computing the parents for each node is not applicable for nodes, which have a single parent or multi-connected parents. The GS algorithm resolves this problem in the last step, which orients the non-directed edges without generating cycles. The algorithm also deals with the cycles with the minimal number of edges needed be reversed. Margaritis [14] also presents an alternative algorithm named "Randomized Version of Grow-Shrink Algorithm" to resolve this problem. However, this updated version of GS algorithm is not applied in the scope of our paper yet.

Algorithm 2 Grow-Shrink Markov blanket algorithm

```
1: $U \leftarrow$ All variables in domain
2: [**1. Compute Markov blankets**]
3: **for each** $X \in U$ **do**
4:     Calculate Markov blanket of X: $\mathbf{B}(X)$
5: **end for**
6: [**2. Compute direct neighbors**]
7: **for each** $X \in U$ **do**
8:     Initialize the neighbor set of X: $\mathbf{N}$(X) $\leftarrow \emptyset$
9:     **for each** $Y \in \mathbf{B}(X)$ **do**
10:        Compute set $\mathbf{T}$ as the smaller set between $\mathbf{B}$(X) - {Y} and $\mathbf{B}$(Y) - {X}
11:        **if** $X \not\perp Y|S \ \ \forall S \subset \mathbf{T}$ **then**
12:            $\mathbf{N}(X) \leftarrow \mathbf{N}(X) \cup \{Y\}$
13:        **end if**
14:    **end for**
15: **end for**
16: [**3. Compute parents**]
17: **for each** $X \in U$ *and* $Y \in \mathbf{N}(X)$ **do**
18:    **for each** $Z \in \mathbf{N}(X) - \mathbf{N}(Y) - \{Y\}$ **do**
19:        Compute set $\mathbf{T}$ as the smaller set between $\mathbf{B}$(Y) - {X, Z} and $\mathbf{B}$(Z) - {X,Y}
20:        **if** $\exists Z$ that $Y \perp Z|S \cup \{X\} \ \ \forall S \subset \mathbf{T}$ **then**
21:            Orient $Y \rightarrow X$ (Y is parent of X)
22:        **end if**
23:    **end for**
24: **end for**
25: [**4. Remove cycles**]
26: **while** there exist cycles in network **do**
27:    Compute set $\mathbf{R}$ containing all edges in all cycles with duplication
28:    Reverse edge with the highest count of appearances in $\mathbf{R}$
29: **end while**
30: [**5, Complete the network**]
31: **for each** $X \in U$ *and* $Y \in \mathbf{N}(X)$ **do**
32:    **if** $X - Y$ is not oriented **then**
33:        Orient $X \rightarrow Y$
34:    **end if**
35: **end for**
```

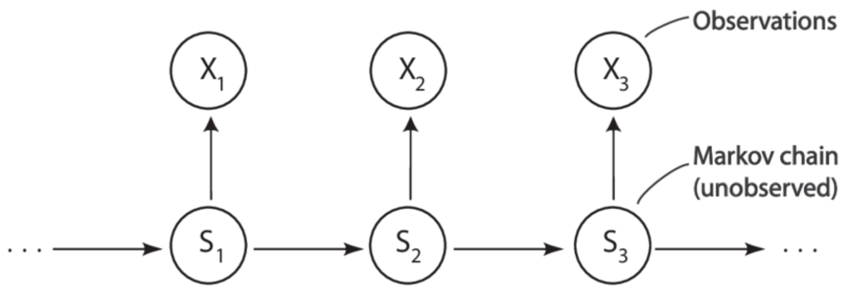

Figure 3.10: Example of Hidden Markov model [11]

3.4 Hidden Markov model (HMM)

Hidden Markov model (HMM) and Bayesian network are probabilistic graphical models, which are represented in a DAG. The basic idea of HMM is based on Markov chain. Gagniuc [11] defines that Markov chain is a model encoding the probability of a sequence of variables, where each variable is drawn from some set of variables. This technique is useful to predict the future of a sequence with respect to the current state. Markov chain is built based on the Markov assumption, that the probability of one variable in a sequence depends only on the previous variable. In other words, the present is the only matter, which impacts predicting the future.

Markov assumption: $P(X_i|X_1, X_2, ..., X_{i-1}) = P(X_i|X_{i-1})$.

HMM is an extension of the Markov chain which has a sequence of hidden states, which can be estimated by observing the process of emissions. An HMM is described by N—the number of possible values of the hidden state (a finite number), and 3 sets of parameters—probability distributions:

1. I - initial state probability distribution,
2. E - emission probability distribution given the state,
3. T - transition probability distribution matrix given the state.

HMM is widely used to find the most probable sequence of hidden states given the sequence of observations of emissions.

3.5 Bayesian knowledge tracing model

Bayesian Knowledge Tracing (BKT) model was presented by Corbett and Anderson in 1994 [4], and has been applied widely in ITS field. The fundamental idea behind a BKT model is a hidden Markov model. Our purpose for a student model is to predict students' mastery state on one skill through their performance in multiple tests, where the observed results over tests are the emission of their state. In our BKT models, the hidden variables are the mastery states, representing whether student has "learnt" or "not-learnt" a specific skill, while the emission variables are the performance over time of student on that skill. By assessing the students' performance over time, the BKT models can monitor whether the student has understood the skill at each test session. The models also can predict the students' results of that skill for the next upcoming test sessions, based on the latest performance of the student.

Van de Sande [20] has summarized the properties of the BKT models that a BKT model consider the hidden states (mastery states) to have only 2 values, which are "learnt" or "not learnt", with 4 parameters to optimize:

- $P(L_0)$ the probability that student has learnt the skill before the first attempt (corresponding to distribution I in HMM)
- $P(G)$ is the probability that a student has not learnt the skill, but luckily guesses the correct answer (corresponding to distribution E in HMM)
- $P(S)$ is the probability that a student has learnt the skill, but accidentally makes a slip and gives a wrong answer (corresponding to distribution E in HMM)
- $P(T)$ is the learning probability that present the chance a student can learn the skill after one attempt. This $P(T)$ is assumed to be constant over time (corresponding to distributions T in HMM)

In this paper, we take into account the chance that a student can forget a skill that he has learnt with probability $P(F)$ after one test session. The matrix

$$\begin{bmatrix} \text{1 - P(T)} & \text{P(T)} \\ \text{P(F)} & \text{1 - P(F)} \end{bmatrix}$$

is the matrix of hidden state transitions. The state transition could be

- learning the skill (from "not-learnt" state to "learnt" state),
- forgetting learnt skill (from "learnt" state to "not-learnt" state)

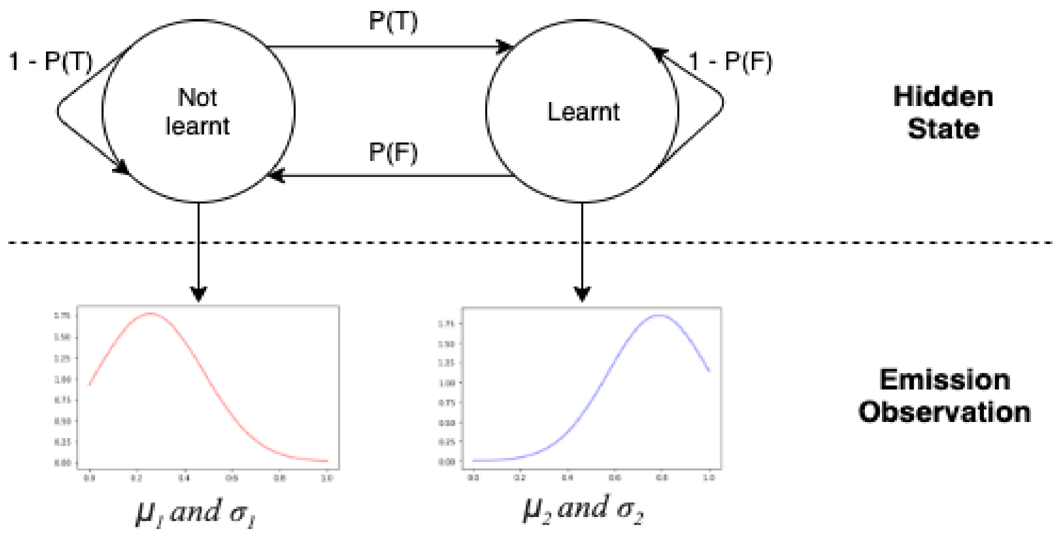

Figure 3.11: BKT parameters diagram (1 means "not learnt", and 2 means "learnt". X-axis is the score on the concept, Y-axis is the density.

- or staying the same state.

We assume that $P(T)$ and $P(F)$ are constant over time. Our target is estimating the mastery state for each skill of students based on their performance on that skill over test sessions. Hence, instead of using sequences of questions with only "wrong" and "correct" responses (0 or 1), which are predicted by the state of mastery of the student and $P(G)$ and $P(S)$; we use the list of test sessions, where questions are grouped by concept, with the emission values (in the interval between 0.0 and 1.0). The emission values are drawn from two normal distributions representing the two hidden mastery states: "learnt" and "not learnt". Consequently, the parameters of the BKT model become $P(L_0)$, $P(T)$, $P(F)$, μ and σ where $\mu = [\ \mu_{not\,learnt},\ \mu_{learnt}\ ]$ and $\sigma = [\ \sigma_{not\,learnt},\ \sigma_{learnt}\ ]$. The emission probability distribution $P(E|S = learnt)$ and $P(E|S = not\_learnt)$ now are calculated by the probability density function of the normal distribution of each hidden state rather than based on $P(S)$ and $P(G)$. The detail of connection between these parameters is presented in Figure 3.11.

The probability that a student has learnt the skill at each test session is dependent on the updated performance of that student; thus that probability at each test session is not constant over time. Let k be the index of the step (test session) in the sequence of test sessions; $P(L_k)$ is defined as the probability that the student has learnt the skill at the k^{th} test session and E_k defines the performance of student on that skill (0.0 to 1.0) at the k^{th} test session. The performance E_k is drawn from the corresponding normal distribution of the state at test session k. Figure 3.12 displays the structure of a BKT model.

In Figure 3.11 and Figure 3.12, the formulas for calculating $P(L_k)$ and $P(E_k)$ are

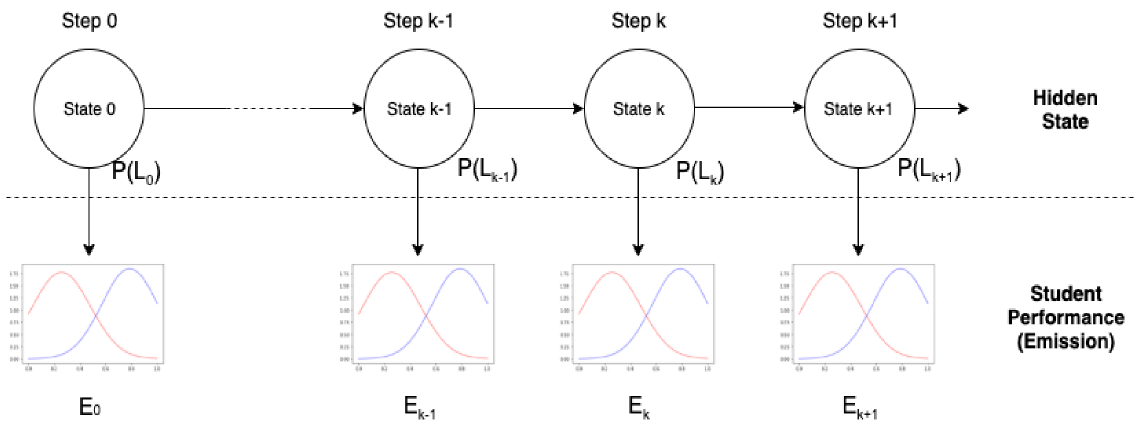

Figure 3.12: Bayesian Knowledge Tracing model

$$P(L_k) = P(L_{k-1})\,(1 - P(F)) + (1 - P(L_{k-1}))\,P(T),$$

and

$$P(E_k) = P(L_k)P(E_k|S_{learnt}) + (1 - P(L_k))P(E_k|S_{not\ learnt}),$$

where

$$P(E_k|S_{not\ learnt}) = \frac{\exp\left(\frac{-(E_k-\mu_1)^2}{2(\sigma_1)^2}\right)}{\sigma_1\sqrt{2\pi}},$$

and

$$P(E_k|S_{learnt}) = \frac{\exp\left(\frac{-(E_k-\mu_2)^2}{2(\sigma_2)^2}\right)}{\sigma_2\sqrt{2\pi}}.$$

Given the emission sequence E, the log-likelihood of E, $P(E)$ is calculated as

$$P(E) = \prod P(E_k), \quad for\ all\ k,$$

or

$$\log(P(E)) = \sum \log(P(E_k)), \quad for\ all\ k,$$

where E_k is the observed test results of the student over test sessions. We can maximize the log-likelihood of the sequence of emissions, $\log(P(E))$, by applying the Expectation Maximization (EM) algorithm to learn parameters of a BKT model.

4 Related Work

There are three major objectives of our paper, which can be achieved by learning the student model and the domain model. The student model is used to track the advancement of the proficiency level of students on concepts. The domain model is used to represent the structure of the knowledge domain and the connections among all concepts.

Bayesian Knowledge Tracing model is one common technique to track the advancement of proficiency levels (knowledge states) of students over test sessions based on the students' performance. The standard BKT contains 2 knowledge states which are "learnt" and "not-learnt", and probabilities to determine the advancement process [**bkt˙related**]. Yudelson et al. [**bkt˙related**] present a promising result for applying BKT on estimating individual knowledge advancement. Zhanga and Yao [29] introduce an update version of the BKT, which has the name is three learning states BKT (TLS-BKT) with 3 states "not-learnt", "learning" and "learnt". They believe that the TLS-BKT has a better performance than the original BKT with 2 states. The common point between BKT and TLS-BKT is that they both ignore the probability of forgetting. In this thesis, we utilize the standard BKT with 2 states, but we take the forgetting into account to track the knowledge states of students.

There are many solutions have been proposed to learn knowledge structure as a corresponding knowledge state graph such as BLIM [10], distributed learning object [26], or maximum residuals method [19]. One popular mechanism to construct knowledge state graph is using the frequency of response pattern from student's answers. The BLIM algorithm which is invented by Falmagne and Doignon in 1988 attempts to select the most probable knowledge states belong to knowledge structure K. To access the states, the algorithm takes into account both conditional probability and frequency of knowledge states and answer response patterns from students to find the most likely set of knowledge states that generates the observed responses. Schrepp [23] proposes a method to determine which response pattern is belong to the knowledge domain. He assumes that a response pattern could be generated from one knowledge state in domain in order find an optimal frequency threshold L in such way that all response patterns arriving more than L times are considered as knowledge states in domain. However due to the probability of lucky guess or careless mistake, the frequency non-state-patterns are possibly higher than some state-patterns. To handle this issue, Robusto and Stefanutti [19] proposed a mechanism that appends new factor named expected frequencies of the response patterns. This expected frequency are able to be obtained by BLIM algorithm [10]. The common point of all the above methods is around observing the frequencies of response patterns to construct a DAG state graph. Constructing a state graph for a large domain with many concepts and skills or the graph is actually flatten (the component level is small) consumes a lot of resources since the number of possible knowledge states in the graph could be exponential (Section 3.1).

To alter the knowledge state graph in order to represent the study domain, the surmise graph of concepts is a decent candidate where the skill map is still able to encode the surmise relationships among skills in domain as well as be easily converted to knowledge state graph. Desmarais and Gagnon have introduced quite similar idea in their research on learning knowledge structure as item graph [6]. They attempt to present a knowledge domain in form of a Bayesian network presenting the item to item structure. By the item to item structure, they mean a DAG structure that represents all surmise relationships between all individual items in the domain. To do so, they propose two approaches. The first approach learns the corresponding Bayesian network on observed data with K2 algorithm [3] and PC algorithm [24]. The second approach believes that the knowledge items in a study domain are in surmise relations following a partial order in which some skills should be mastered before the others. They utilize this topological theory in order to iterate over all pair of items to find the surmise relation in each pair. They consider that $itemA \succ itemB$ (the mastery of B precedes the mastery of A) if the following conditions are satisfied

$$P(B|A) \geq p_c,$$

and

$$P(\bar{A}|\bar{B}) \geq p_c,$$

and

$$P(B|A) \neq P(B),$$

where A and B stands for “mastering” item A and item B while $\bar{A}$ and $\bar{B}$ presents for “not mastering” these items. They experiment their theory with $p_c = 0.5$.

Hou et al. [12] try to learn the study domain as a skill map structure with Elo-based methodology. Their result skill map is flattened, therefore the size of corresponding state structure is exponential if we try to recover it. Those works of learning item to item structure or the surmise graph of concepts deduces that it is possible to understand the study domain as a surmise graph in which learning surmise graph is more achievable in some scenarios.

Because our second objective is learning the surmise graph, which is a Bayesian network, from the mastery data; algorithms for Bayesian network learning also can be used to construct the domain model. Yuan and Malone [28] show that there are three main approaches to learn a

Bayesian network: score-based learning, constraint-based learning, and hybrid methods. The score-based learning is NP-hard, which is only suitable for small dataset. The score-based learning evaluates the quality of the network by applying a score formula on all possible network structures in solution space. The constraint-based learning is able to scale the large dataset but vulnerable to the noises. The hybrid learning is the integration between the advantages of two base learning methods: score-based learning and constraint-based learning. To reduce the running for score-based learning, Yuan and Malone [28] present a dynamic programming approach, which constructs the order tree and then they use a search algorithm such A* to find the optimal path to the best network structure. By the order tree, they mean a state space graph for learning Bayesian network, which has the most-top node is a network structure with only one random node (a start node) and the most-bottom nodes are the network structure with all involved nodes. In terms of constraint-based and hybrid learning, Margaritis [14] presents two solution: Grow-shrink Markov blanket algorithm and Randomized Version of Grow-Shrink Algorithm, which try to identify the conditional independence relations from the data to construct the network structure.

5 Research Methodology

The target of this thesis is applying the KST to the domain of language learning to construct the corresponding domain model and student model with three main objectives). The student model is also used to estimate the mastery data, which is the input for learning the domain model.

To learn the domain model, we apply Greedy algorithm and two Bayesian network learning libraries, which are *Gobnilp* (Globally Optimal Bayesian Network learning using Integer Linear Programming) [5] and *pomegranate* [21] as baseline algorithms. We also apply GS algorithm and Theory-based algorithm to compare with three baseline algorithms (Greedy algorithm, Gobnilp, pomegranate).

To learn the student model, we use *hmmlearn* library in Python [27] to learn the BKT models because the basic idea of BKT models is Hidden Markov Models. We use *GaussianHMM* class to construct the HMM of our BKT models since we assume that the score of a mastery state ("learnt" or "not-learnt") is a variable with normal distribution (Gaussian distribution).

The methodology flow is displayed in Figure 5.1. The student model (BKT models or Threshold-based estimator) estimates the mastery state data for the input score data. The estimated mastery data is the input for one of three GS, Greedy score-based, Theory-based algorithm, Gobnilp library and pomegranate library to learn the surmise graph of concepts (the domain model), which represent the domain of learning language. We have 10 configurations of the methodology to learn the student model and the domain model, which are BKT-GS, BKT-Greedy, BKT-Theory, BKT-Gobnilp, BKT-pomegranate, Threshold-GS, Threshold-Greedy, Threshold-Theory, Threshold-Gobnilp, and Threshold-pomegranate.

According to Section 3.1, the domain model of a knowledge space is an Bayesian network, which encodes the surmise relationship among concepts in the domain, and the student model contains a BKT model for each concept, to estimate the proficiency level by tracking students' performance. Section 2.2 presents that there are 88 concepts, which are distributed among six levels of CEFR; therefore we we have two different sets of concept to represent the knowledge space. In this thesis, we experiment with the KST on both two sets of concepts.

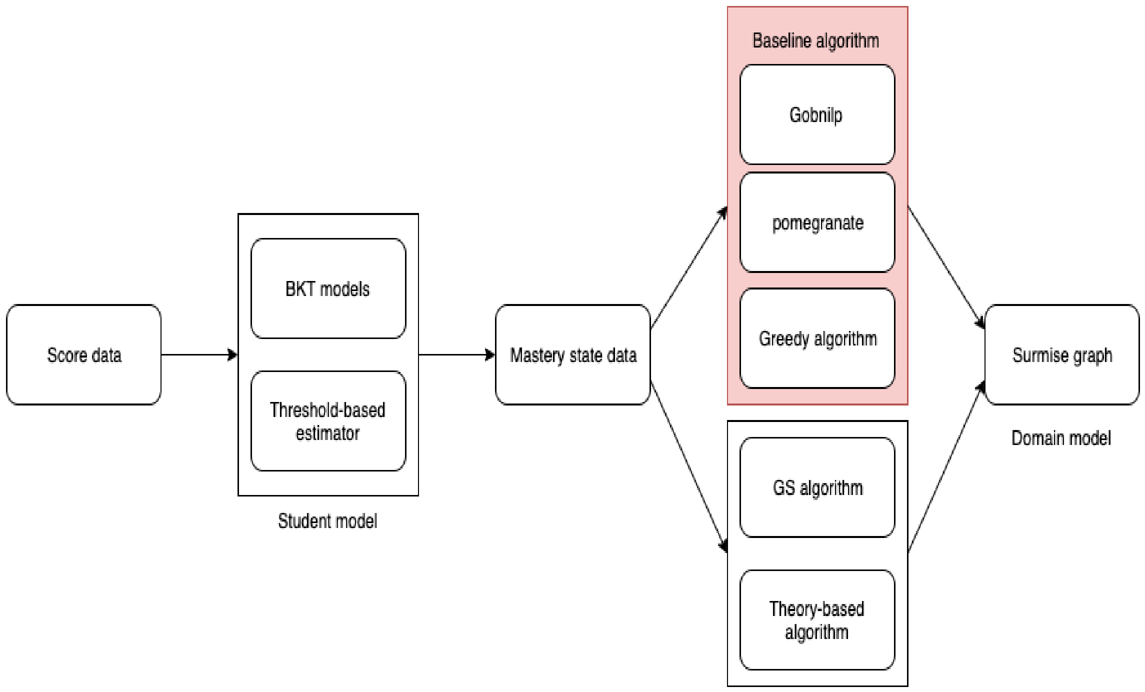

Figure 5.1: Methodology flow

5.1 Learning student models

The function of the student model is analysing the test result of students and assessing their mastery state on each skill in each test. We propose two mechanisms to learn the student model that can estimate the proficiency levels of students based on their sequence of performance. Both techniques are applied the "student" data set.

5.1.1 Threshold-based model

The first is a naive approach, which determines the mastery state by a mastery threshold, θ, (a hyper-parameter, which we tune) on the performance of students for each skill. We say that a student has learnt the concept (mastery state) if their score on this concept is equal or higher than the mastery threshold. For example, if the score of student A on concept B in test session C is 0.8 (or 0.6), and the mastery threshold θ is 0.7, then we consider that student A has learnt (or not learnt) concept B, respectively, at the time of session C. More detailed examples are displayed in Table 5.1 and Table 5.2.

Table 5.1 is a sample data table of score over N concepts in three test sessions. In the data, each test session also includes the timestamp and student identity number; however that information is not used in this first approach. The corresponding state data table for Table 5.1 with respect to $\theta = 0.7$ is presented in Table 5.2.

Test session	Concept 1	Concept 2	...	Concept N
001	0.2	0.6	...	null
002	0.7	null	...	0.8
003	0.4	0.5	...	0.7

Table 5.1: Example score data table

Test session	Concept 1	Concept 2	...	Concept N
001	0	0	...	null
002	1	null	...	1
003	0	0	...	1

Table 5.2: State data corresponding to Table 2 (with $\theta = 0.7$)

This method ignores the time factor, and treats each test session as if it is done by different a individual, disregarding the student ID. Therefore, the first technique is simple with only one parameter θ ($\theta = 1 - P(S)$), which involves the probability $P(S)$ that a student who has learnt the concept makes accidental mistake/slip. In this thesis, the parameter θ is tuned from 0.5 to 0.9, with an increment of 0.05, and the impact report is presented in Experiment section.

5.1.2 Bayesian Knowledge Tracing model

The second approach utilizes the Bayesian knowledge tracing model that takes both the timestamps and the IDs of the students into account to estimate the mastery state. This solution treats test sessions in sequences in such way that they are grouped by student ID and ordered by timestamp. The mastery states on each concept are estimated by not only the performance of this concept on the current test session, but also by the previous sessions. We train one BKT model for each concept in the domain. The BKT model is used to learn the mastery state data, which is used to learn the surmise graph, for example, by the GS algorithm and other algorithms, in Section 5.2.

In a test sequence of one student on a single concept, there are originally five parameters that we want to learn, which are the probability, $P(L_0)$, that a student already learnt the concept in the first attempt, the probability, $P(L)$, that a student is able to learn the concept after one opportunity, the probability, $P(F)$, that a student forgets the learnt concept after one test session, $P(G)$ and $P(S)$, which are the probability that a student makes a lucky guess or a slip. In these parameters, $P(G)$ and $P(S)$ contributes to the score distribution with respect to the mastery state of the current performance. We use two normal distributions to replace $P(G)$ and $P(S)$, as we discussed in Section 3.5.

The student models learnt by this step are used to predict the mastery state for the "Domain" dataset as the input for training domain model, as well as to estimate the mastery state data for the "Evaluation" dataset in order to evaluate the domain model (Section 2.2).

5.2 Learning domain models

The objectives of domain model is to encode the inter-dependencies among concepts in the domain. Section 3.1 mentions that the domain model is originally a directed acyclic graph of knowledge states (knowledge state graph), but it can be represented by a surmise graph of concepts. As mentioned in Section 2.2, we utilize the student models, which are learnt from the *student* dataset to estimate the mastery state data for the *domain* dataset. The estimated state data of the *domain* dataset is the input for learning the surmise graph of concepts. To learn the surmise graph of concepts from mastery state data, we apply five solutions: GS algorithm, Theory-based algorithm, Greedy algorithm, Gobnilp library and pomegranate library. The Greedy algorithm, the Gobnilp library and the pomegranate library are baseline methods, which are used as options for comparison to the GS algorithm and Theory-based algorithm. The Gobnilp library [5] and the pomegranate library [21] are Python libraries to learnt Bayesian network from data. The methodology of experiment with the GS algorithm, the Theory-based algorithm and the Greedy algorithm is described as follows.

5.2.1 Theory-driven method

The first solution utilizes our prior assumption on the surmise relationship between concepts: the assumption is that for any pair of concepts A and B, 1. A and B are not connected by a surmise relation, or 2. A is prerequisite for B, or 3. B is prerequisite for A. We believe that some concepts are prerequisite for some other concepts in such way that all concepts connect to each other in a topological order. This assumption is based on the reference to the *parital order knowledge structures* (Desmarais & Gagnon [6]). The surmise relationship between two concepts A and B is described in Table 5.3, where "+" represents for "learnt" the concept and "-" for "not-learnt" the concept.

According to the Table 5.3, if $A \succ B$ then the probability that a student has mastered concept B given that he has mastered concept A should be close to 100 percent. In other words,

Concept A	Concept B	B is prerequisite for A
+	+	compatible
+	-	incompatible
-	+	compatible
-	-	compatible

Table 5.3: Compatibility of states of A and B with the prerequisite relation between concepts $A \succ B$

if you mastered the harder concept, you should have mastered the prerequisite concepts. If you are not able to understand the prerequisite concept then you are not likely to master the harder concept.

For every pair of concepts A and B, this method follows the rules between concepts in Table 5.3 to check whether one concept is prerequisite for the other. To check the compatibility, we look up the input samples to calculate the corresponding conditional probability for each case in Table 5.3. Due to the uncertainty of estimated state data, we assume that the conditional probabilities are not zero and one. For example, $P(Learnt\,B \mid Learnt\,A)$ does not need to be 1.0 to consider the first case is compatible. We use a threshold hyper-parameter ϕ in such a way that $P(Learnt\,B \mid Learnt\,A)$ should be equal or higher than ϕ (rather than equal 1.0) or the probability $P(Not-Learnt\,B \mid Learnt\,A)$ should be lower than $1-\phi$ rather than 0.0 to assess the second case as incompatible. Parameter ϕ is also tuned in range 0.5 to 1.0 with step 0.05. The concept pairs that are not compatible the prerequisite connection indicate these pairs of concepts have no surmise relation.

5.2.2 Greedy score-based data-driven method

The second approach is a score-based data-driven method, which tries to find the most probable surmise graph of concepts given the input mastery data. We start by the following formula for the score function

$$Score(G, D) = P(G|D) = \frac{P(D|G)P(G)}{P(D)}$$

This function is mentioned in Section 3.2 where G stands for a graph structure and D stands for training data, which is the estimated mastery data. The goal is finding the structure G that results in the highest value of score on D by maximizing $P(G|D)$ or $P(D|G)$, where the distribution $P(G)$ is uniform.

The probability $P(D|G)$ can be factorized—written as a product of the individual density functions for all variables with respect to the evidence of their parents. The function $P(D|G)$

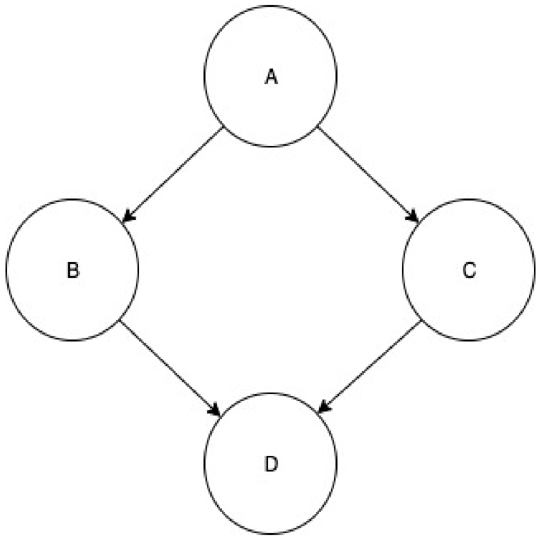

Figure 5.2: Directed Acyclic Graph G'

can be presented as follows

$$P(D|G) = \prod P(X_v|G, Parent(X_v)), \quad \forall v \in Q$$

or

$$\log(P(D|G)) = \sum \log(P(X_v|G, Parent(X_v))), \quad \forall v \in Q,$$

where Q is the set of all concepts in the domain. Equation (5.1) is an example for network factorization for graph G' in Figure 5.2.

$$P(A, B, C, D) \; = \; P(A)\; P(B|A)\; P(C|A)\; P(D|B, C) \tag{5.1}$$

According to the factorization of the likelihood $P(D|G)$, we can maximize each part of the factorization in order to achieve the greatest value for $P(D|G)$. It can be done by trying all parent configurations for each variable to find the best configuration for the greatest conditional probability, $P(X_v|G, Parent(X_v))$. This method iterates all possible structures of the surmise graph to select the structure with the highest likelihood on the training data. However, we have 88 or 127 concepts, which means that the total number of possible structures is 2^{88} or 2^{127} respectively. Therefore, this naive approach is not practical.

In order to reduce the time complexity, we use an alternative approach to trade-off quality for speed to search for an approximate structure. We start with a random structure and keep modifying it to boost the likelihood on data until we decide to stop. The modification at each iteration is possibly removing an edge, reversing an edge or adding a new edge. The algorithm selects the modification on each iteration that drives the factorization changes with the highest increase of likelihood on the training data. The early stop rules are determined by two factors: the maximum number of iterations (a suitable number of iterations for our environment) and a threshold of minimum likelihood score increase after each iteration. We experiment with this technique with several initial structures, which includes an empty network (DAG without any edges), a fully-connected network with randomly assigned directions, and some random networks (not empty and fully-connected).

5.2.3 Grow-shrink Markov blanket data-driven method

Grow-shrink Markov blanket algorithm, as described in Section 3.3, is another data-driven option to learn the Bayesian network structure. This algorithm has lower time complexity than the greedy score-based solution. In this paper, we also attempt to apply the GS algorithm [14] to learn the surmise graph of concepts. The detailed explanation of GS algorithm can be found in Section 3.3. The running time and performance report for GS algorithm is presented along with two first solutions in Section 6.

6 Experiment Results

The libraries of Gobnilp and pomegranate in Python are not able to construct the Bayesian network among 88 or 127 concepts. They utilize more than $32GB$ memory (our maximum memory) and get stopped during the learning. Gobnilp is able to learn the network structure for maximum 20 concepts with respect to our resource. Pomegranate is able to learn the network structure for maximum 23 concepts with respect to our resource. On the other hand, the GS algorithm, Greedy score-based algorithm and Theory-based algorithm are possible to handle all 88 or 127 concepts with less memory usage (up to $16GB$)

6.1 Concepts without CEFR

In this experiment, we attempt to learn the knowledge space for the 88 concepts, not using their CEFR information. The learnt surmise graphs are evaluated based on the evaluation dataset. The evaluation dataset has been transformed into mastery states by BKT model or Threshold-based estimator in Section 3.5 and Section 5.1.1. The log-likelihood is computed from the mastery states $M(D)$ of the evaluation dataset given the surmise graph G, by $logP(M(D)|G)$. The likelihood $P(M(D)|G)$ is computed by following formula

$$P(M(D)|G) = \prod P(S|G),\ \forall S \in M(D),$$

where S is one test session in $M(D)$. The probability $P(S|G)$ for each test session S in $M(D)$ is calculated by the factorization, as following formula

$$P(S|G) = \prod P(C_S|Parent(C_S)),\ \forall C \in Q,$$

where Q is set of all concepts and C_S is mastery state of concept C in test session S.

6.1.1 Bayesian knowledge tracing models

We train 88 BKT models for all concepts. The total time for training all models is approximately 82 seconds, using 4 processors. The mean scores for the "learnt" state for each concept are mostly in the interval between 0.75 and 0.85, except concepts labeled "7", "8", "191", "230", "85" and "89". The mean scores for the "not-learnt" state are mostly in the interval between 0.1 and 0.3. There are two significant exceptions, which are concept "85"

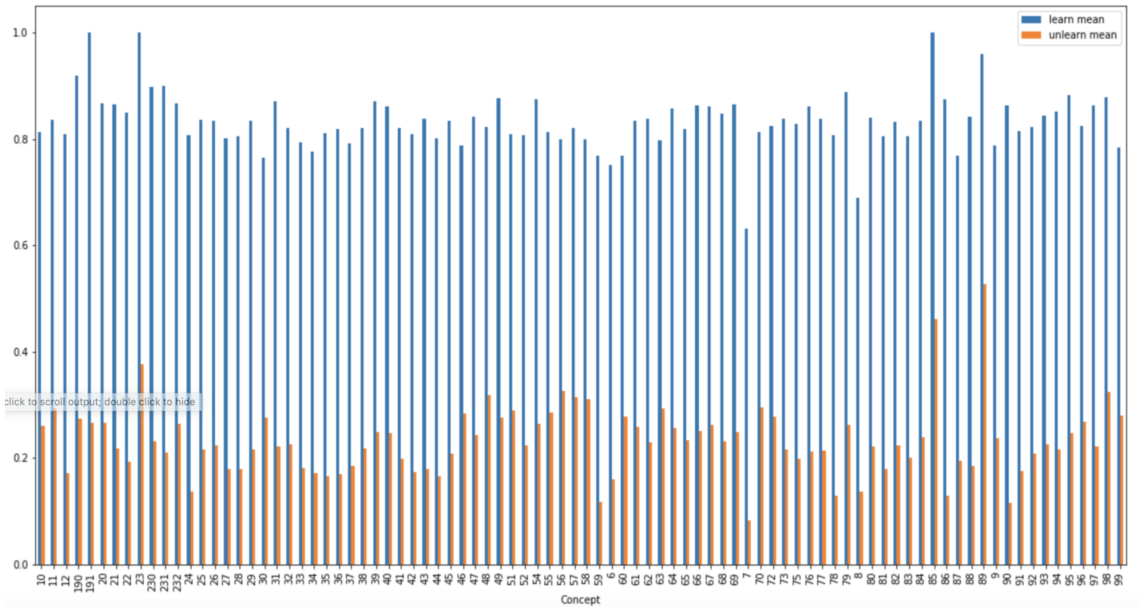

Figure 6.1: Mean score of "learnt" and "not-learnt" of concepts (without CEFR), learned by the BKT model

and "89", which have higher scores for "not-learnt" state than mean value for "not-learnt" state of the other concepts. These means are displayed in Figure 6.1.

Figure 6.2 is a box plot for average values of all parameters over all 88 BKT models. In the figure, $P(T)$ and $P(F)$ stands for the average probability of transition from "not-learnt" state to "learnt" state and vice versa. The terms "learn mean" and "learn var" are mean values of μ and σ^2, respectively, for the score distribution of "learnt" state; the terms "unlearn mean" and "unlearn var" are the corresponding parameters for score distribution of "not-learnt" state.

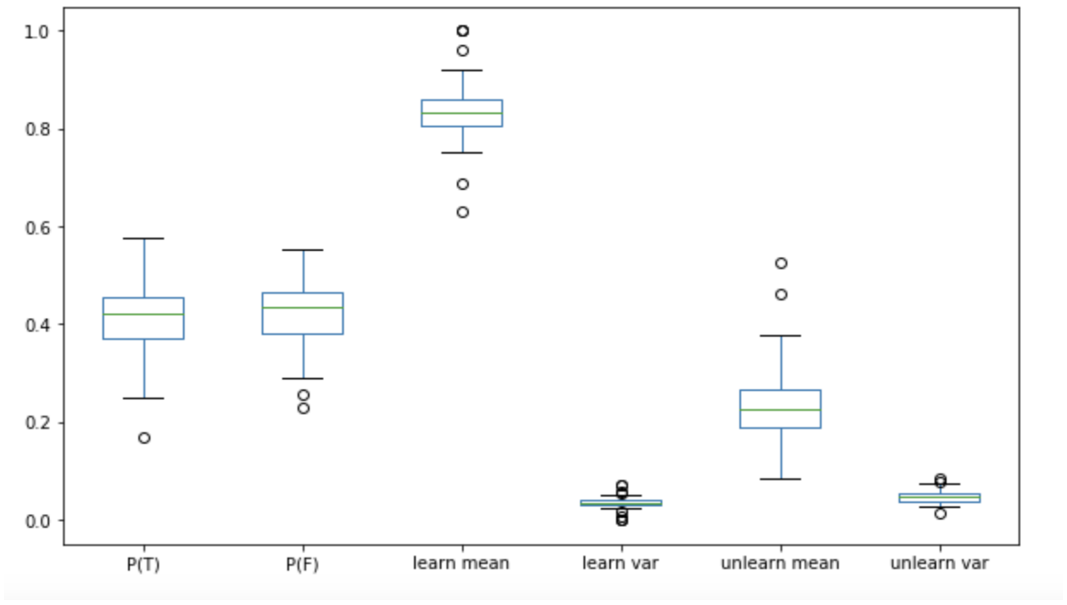

Figure 6.2: Average values of the BKT parameters over all 88 concepts, without CEFR

In the box plot in Figure 6.2, the average values of $P(T)$ and $P(F)$ are approximately 0.4. The score distribution of "learnt" state have mean value around 0.82 with a narrow variance. The mean score for "not-learnt" state is roughly 0.25 with a narrow variance, but slightly wider comparing to the "learnt" state. The outliers in the box plot could come from the "sparsity" of data, which is explained in Section 7.

In Figure 6.3, the only obvious correlations are between variances and means of "learnt" and "not-learnt" states. Although, we cannot observe a clear correlation between mean score of "learnt" and "not-learnt" state in Figure 6.3, the trend of variability of mean score of "learnt" and "not-learnt" state over concepts are somewhat similar in Figure 6.1, with the coefficient of the correlation is 0.21. The increase of mean score of "learnt" state decreases the variance of state, while the increase of mean score of "not-learnt" state increases "not-learnt" variance. As mentioned above, the higher mean score of "learnt" state comes with a somewhat higher mean score of "not-learnt" state. Students seem to make less slips and more lucky guesses on easier concepts, which have a higher mean score of "learnt" and "not-learnt" state.

6.1.2 Learning the domain model

Threshold-based estimation for mastery state:

To estimate the mastery state by naive threshold method, we experiment with thresholds in range 0.5 (50%) to 1.0 (100%). The performance of surmise graphs, which are learnt from threshold-based mastery states, on evaluation data is plotted in Figure 6.4. The y-axis is the score value with the score is the log-likelihood of learnt surmise graphs. The real scores are

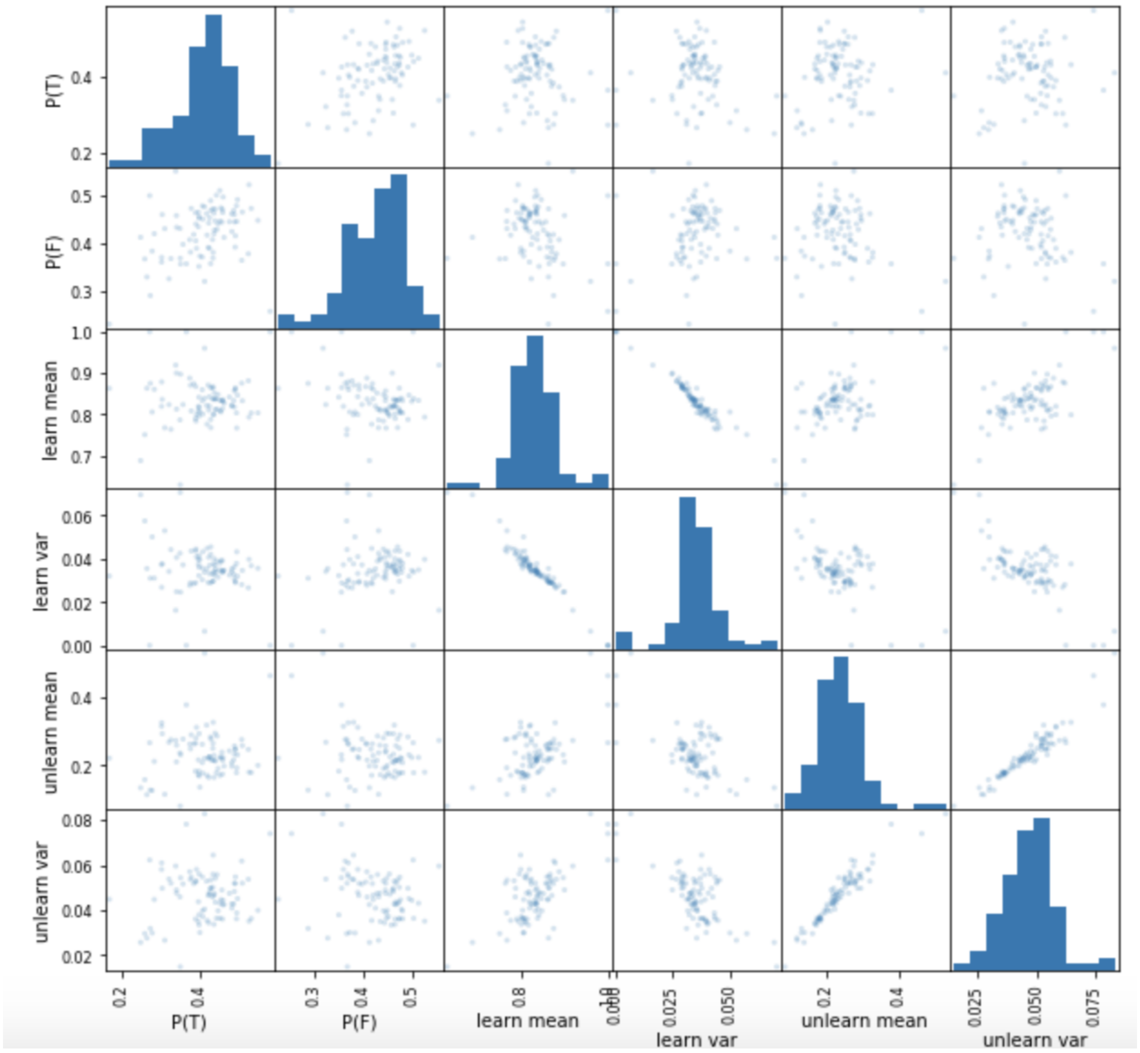

Figure 6.3: Correlation between BKT parameters of concepts without CEFR

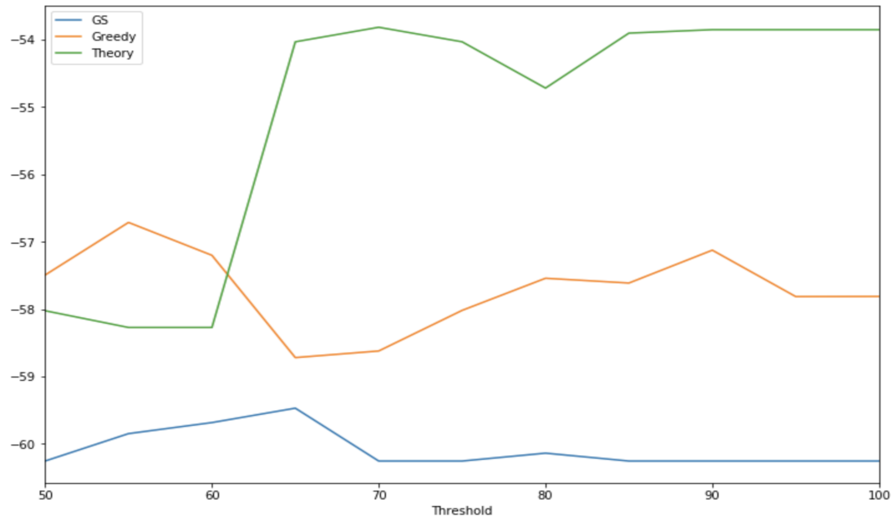

Figure 6.4: Performance of threshold-based domain model of concepts without CEFR

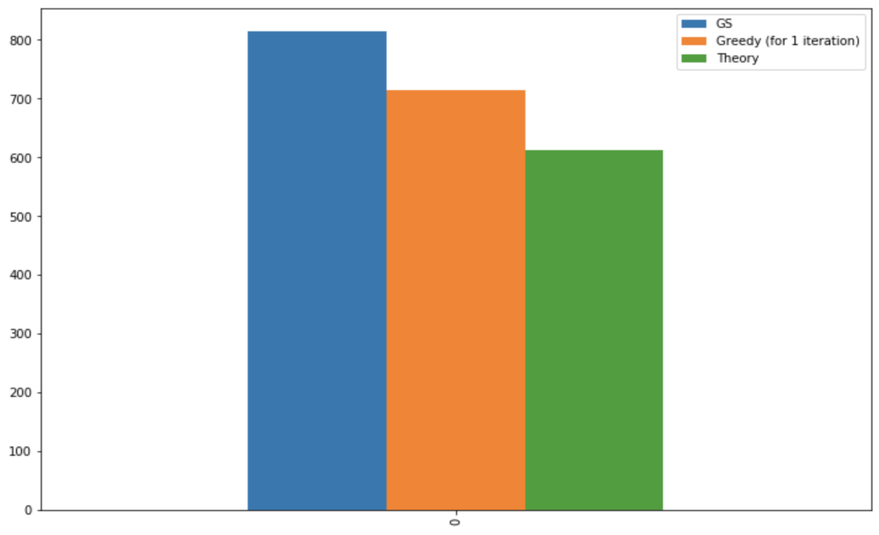

Figure 6.5: Running time (seconds) of learning the domain model on the mastery state data of concepts without CEFR

the multiplication between the values in the figure and 1000.

The score of GS algorithm is worse than Greedy score-based algorithm and Theory-based algorithm. The score in both GS algorithm reaches its peak at threshold 65, where Theory-based algorithm is also near-peak, though it still improves at 70. However, the score of Theory-based solution is mostly stable at the peak after reaching it at threshold 70, while the GS algorithm drops after that. There is no clear trend of score for Greedy score-based method. It seems that the "learnt" state tends to have score higher than 0.65.

Theory-based algorithm is the fastest solution, while the Greedy algorithm is the slowest option, which spends the average time for one iteration almost as the running time of the entire GS algorithm. The running time of the GS algorithm on mastery data, which is estimated by threshold 65, is the fastest (around 550 seconds), but it found almost no surmise relations: the recovered surmise graph has 8 edges. The comparison of running time in average over all threshold values is displayed in Figure 6.5, where y-axis is the average time in seconds. All algorithms are run in the same computing environment.

The Greedy algorithm tends to construct a fully connected DAG, when there is no regularization term penalizing the complexity of the network. In Section 5.2.2, we did not use a regularization term on the complexity of the network, which is used in learning DAG structure in other research, [**score˙based˙complex˙term**]. We compensate for the lack of a regularization term by using stopping criteria for the maximum number of iterations, and the minimum likelihood improvement at each iteration. If we initialize the learning algorithm with an empty DAG, the modifications only *add a new edge* at each iteration. If we initialize with the experiment with a random fully connected DAG, the algorithm performs only

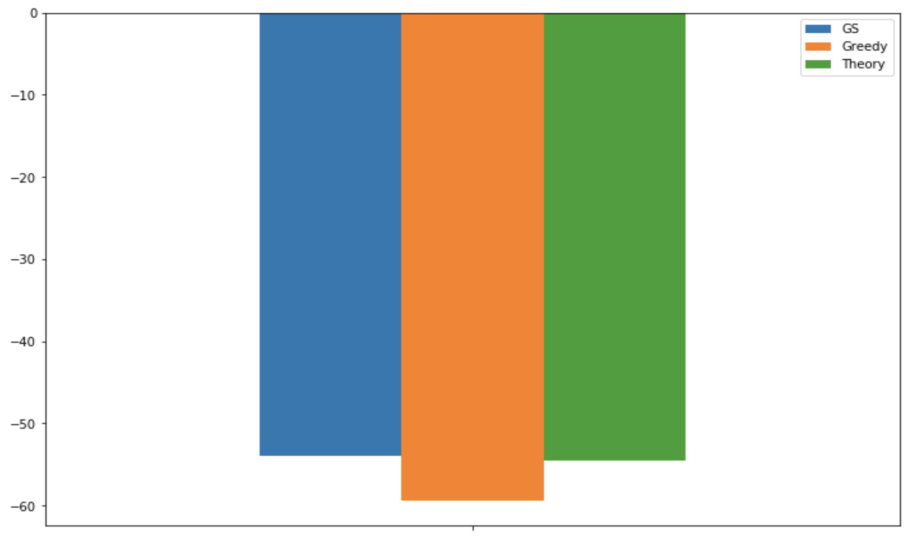

Figure 6.6: Performance of BKT-based domain model on evaluation dataset (concepts without CEFR)

reverse edge modifications at each iteration. In all runs of the algorithm, there is no *remove edge* modifications. In contrast, the GS algorithm recovers almost no surmise relations among concepts; as a result, the log-likelihood score of the GS algorithm has the lowest value.

BKT-based estimation for mastery state:
Figure 6.6 shows the log-likelihood score of surmise graph learnt by GS, Greedy and Theory-based algorithms on the mastery data estimated by BKT models. The performance of GS algorithm and Theory-based algorithm are approximately same (around −54000). The Greedy algorithm has the worst performance, which has the log-likelihood score approximately −59000.

The performance of GS algorithm on the mastery data estimated BKT models is slightly better than the performance of the GS algorithm on mastery state data estimated by the Threshold-based method—0.65 was the best threshold in Threshold-based estimation.

6.2 Concepts with CEFR

In this experiment, we learn the surmise graph from 127 concepts with CEFR. Some concepts such as concept "85B2" and concept "89B2" are distributed over only one CEFR level; therefore, they are similar to the corresponding concepts without CEFR in the experiment in Section 6.1. The log-likelihood of the surmise graph on the evaluation data is calculated in the same way as evaluation of the surmise graph in the experiment with concepts without CEFR (Section 6.1).

The discussion of results again follows the methodology work flow shown in Figure 5.1.

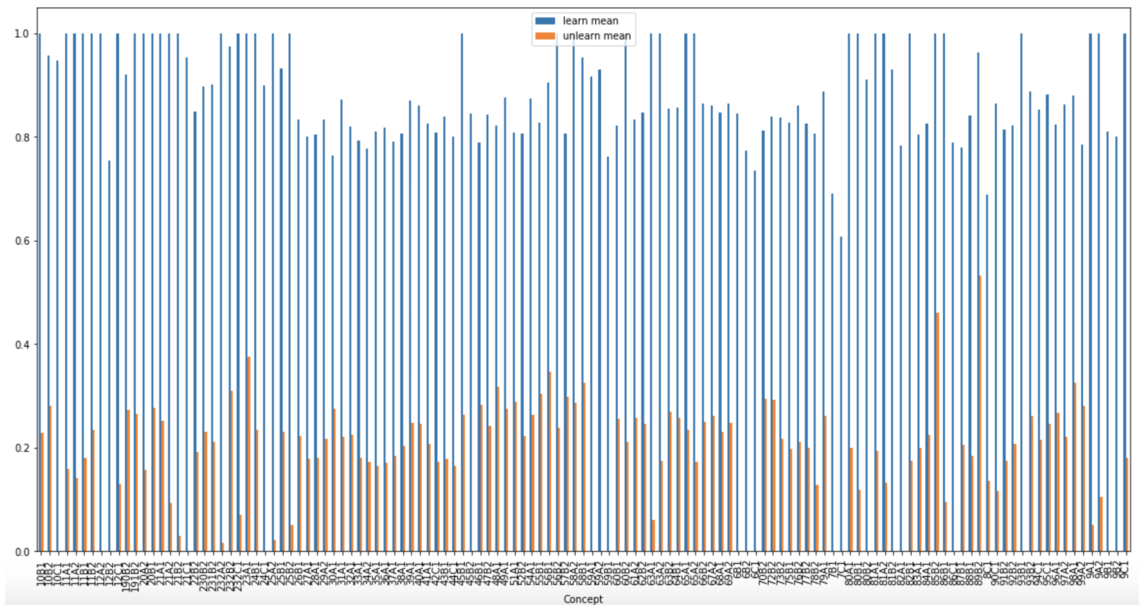

Figure 6.7: Mean score of "learnt" and "not-learnt" of concepts with CEFR

6.2.1 Bayesian knowledge tracing models

We trained 127 BKT models for all concepts. The total time for training 127 models is approximately 135 seconds with 4 processors. Figure 6.7 is the plot of mean score generated by "learnt" and "non-learnt" state.

In Figure 6.7, the mean score of the "learnt" states of many concepts are 1.0 (100%), and the score "not-learnt" states of many concepts are 0.0 (0%). This is the result of the "sparsity" of the data, which is explained in Section 7. Concept "85B2" and "89B2" have mean scores of "not-learnt" state that are especially higher than other concepts.

Figure 6.8 is the box plot for average values of all parameters in BKT models. The terms $P(T)$ and $P(F)$ are the probabilities of learning and forgetting one concept after one test session, respectively (Section 3.5). In the figure, "learn mean" and "learn var" are the mean score and variance of the score distribution of "learnt" state; "unlearn mean" and "unlearn var" are the mean score and variance of the score distribution of "not-learnt" state.

In Figure 6.8, the mean values of $P(T)$ and $P(F)$ are approximately the same as in the experiment with concepts without CEFR. The distribution of the mean score and variance of both "learnt" and "not-learnt" state are wider than these values in the experiment with concepts without CEFR (Section 6.1.1). The mean values of mean score and variance of both states are approximately the same as these parameters in the experiment without CEFR (Section 6.1.1). The mean score of the "learnt" state of concepts with CEFR is in the interval from 0.8 to 0.9. The mean score of the "not-learnt" state of concepts without CEFR is in

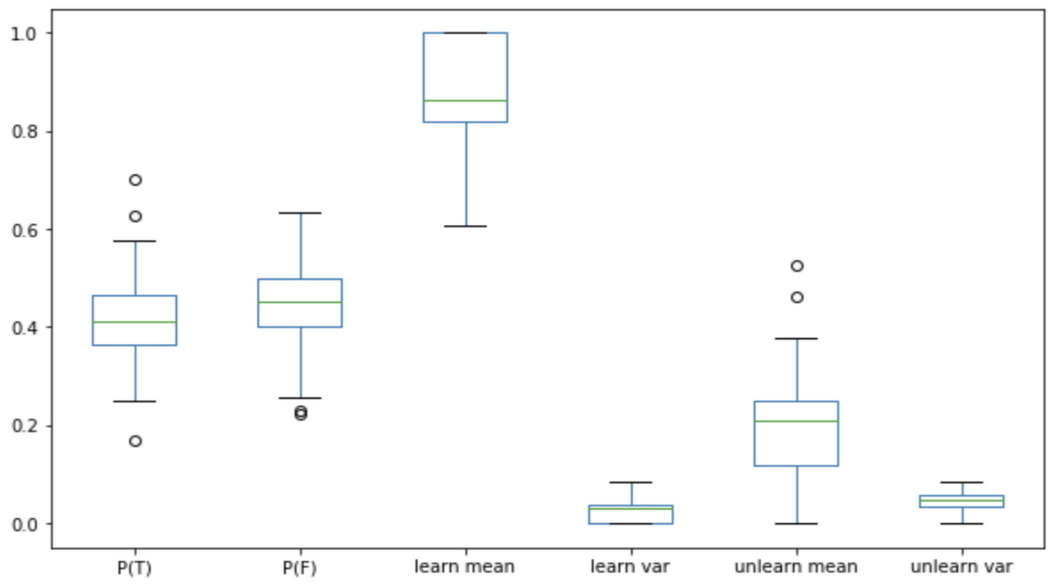

Figure 6.8: Avg. values of BKT parameters of concepts with CEFR

the range from 0.15 to 0.25.

In Figure 6.9, the most obvious correlations are between the variance and mean score of "learnt" state, and the variance and mean of the "not-learnt" state (which is similar to the result in the experiment on concepts without CEFR, Section 6.1.1). An increase in the mean score of the "learnt" state decreases its variance, while an increase in the mean score of the "not-learnt" state increases its variance. Similarly to the experiment on concepts without CEFR, this strongly supports the intuition that students seem to make fewer slips and more lucky guesses on *the easier* concepts.

In the plot of the correlation between the mean score of "learnt" and "not-learnt" state in Figure 6.9, there is some noise, because some concepts (with CEFR) have a mean score of "learnt" at 1.0 and the mean of "not-learnt" state at 0.0. If we ignore the noise, the Pearson correlation between the mean score of the "learnt" state with the mean score of the "not-learnt" state is approximately 0.74, which is quite high. This means that an increase in the mean score of the "learnt" state also gives an increase in the mean score of the "not-learnt" state. This suggests strongly that the *easier* concepts have higher mean score of the "learnt" state and of the "not-learnt" state.

6.2.2 Domain model learning

Threshold-based estimation for mastery state:

The performance of the surmise graph of concepts, which are learnt from the mastery states

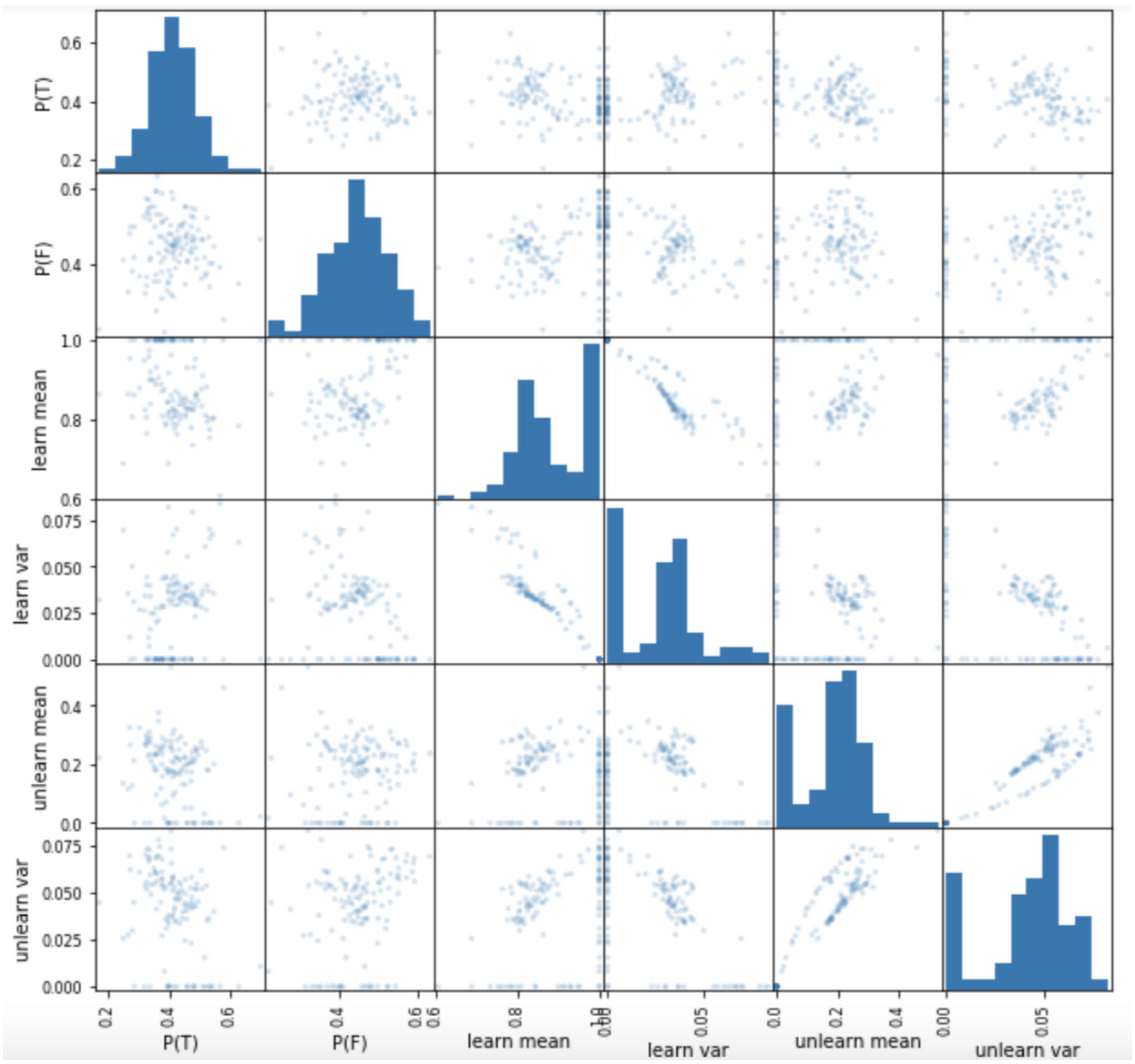

Figure 6.9: Correlation between BKT parameters of concepts with CEFR

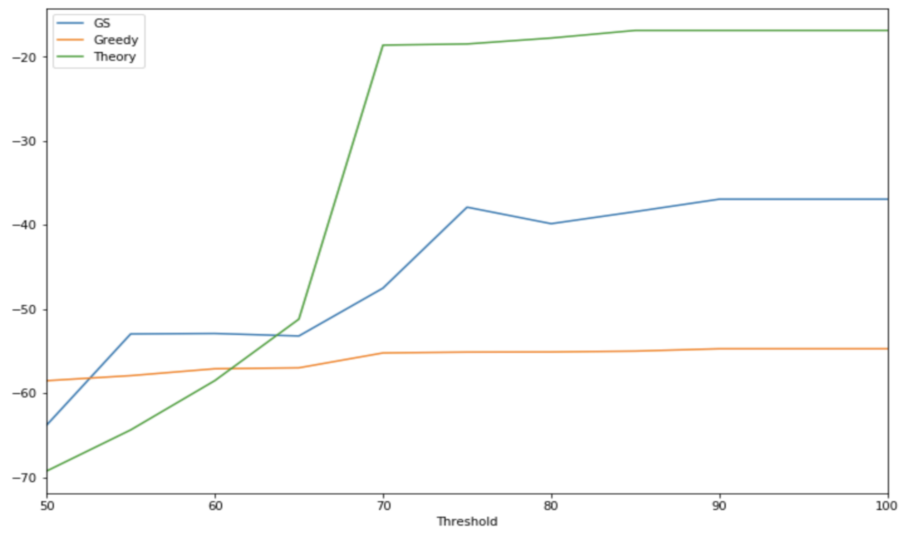

Figure 6.10: Performance of threshold-based domain model of concepts with CEFR

by Threshold-based estimator is plotted in Figure 6.10. In the figure, the x-axis shows the values of thresholds and y-axis shows the log-likelihoods (the y-values are the log-likelihoods divided by 1000) of the surmise graphs on the evaluation dataset. We tried thresholds in range 0.5 (50%) to 1.0 (100%). The general performance of the GS, Theory-based and Greedy score-based algorithms on the mastery states from the Threshold-based estimator in the experiment on concepts with CEFR have higher average log-likelihoods than in experiment on concepts without CEFR (comparing Figure 6.10 and Figure 6.4).

In Figure 6.10, the general performance of the Theory-based method on the mastery data, from the Threshold-based estimator, is better than the GS and Greedy method. The performance of the GS algorithm and Theory-based method have a similar trend, which increases strongly from about threshold 50 to 70 or 75 then is stable from threshold 75 to 100. This shows that the threshold 70 or 75 seems to be the most probable mastery threshold. This means that students who learnt skill A tend to have scores for skill A higher than 0.70 or 0.75.

The GS algorithm is the fastest, which is four times faster than the Theory-based algorithm and hundreds of times faster than the Greedy score-based algorithm. The details of running time is presented in Figure 6.11, where the running time is given on the y-axis in seconds.

BKT-based estimation for mastery state:
Figure 6.12, shows the comparison between the performance of the GS, Greedy and Theory-based algorithms on the mastery data that is estimated by BKT models. The y-axis is the log-likelihood (divided by 1000) of the surmise graphs obtained from the mastery data. The GS algorithm is the winner with the highest log-likelihood, -22000, and the Theory-based algorithm is the much worse, with the log-likelihood almost -49000. The log-likelihood of

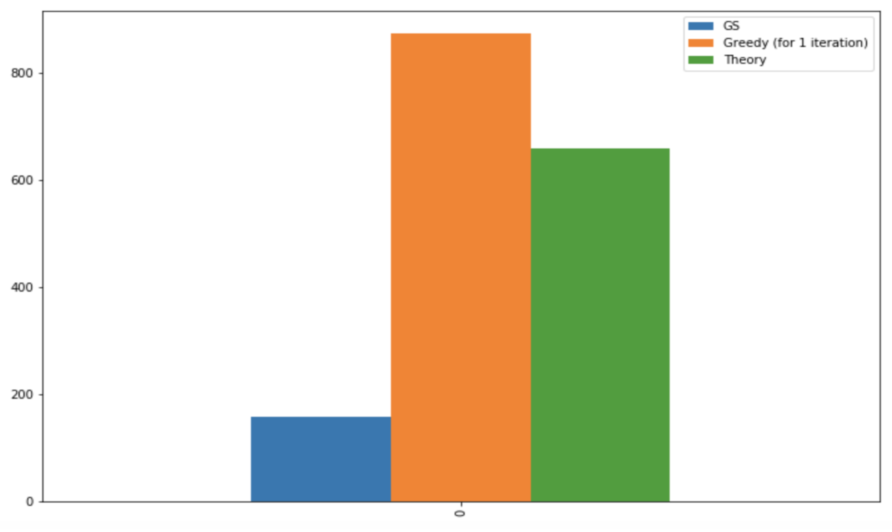

Figure 6.11: Running time of threshold-based domain model of concepts with CEFR

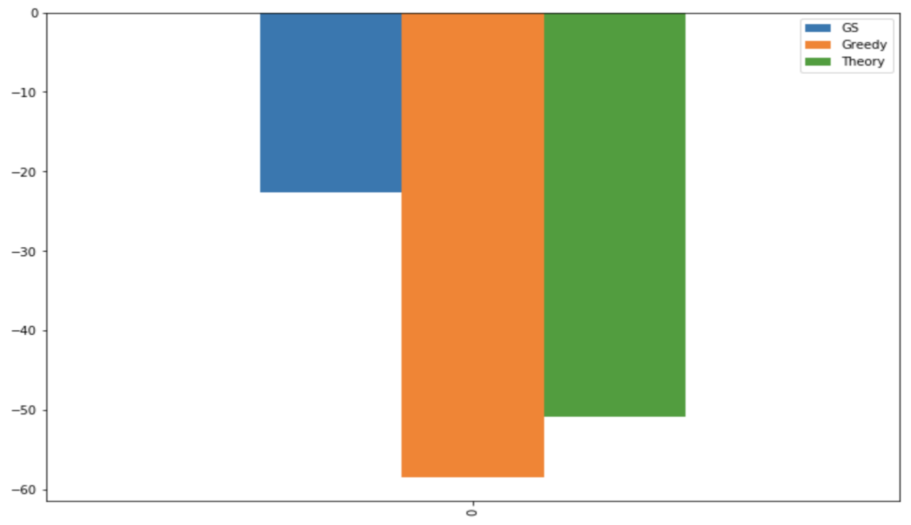

Figure 6.12: Performance of BKT-based domain model on evaluation dataset (concepts with CEFR)

the Greedy score-based algorithm is almost −59000.

The performance of the GS algorithm on the mastery data estimated by BKT models is significantly higher than its performance on the mastery states estimated by Threshold-based estimator (where 0.7 is the best threshold). The combination of BKT method and GS algorithm on concepts with CEFR gives the best result on both quality and running time.

6.3 Connection to objectives

Objective 1—Tracking advancement of proficiency level of student. The Threshold-based estimator has the best performance with threshold from 65 to 70. The Threshold-based estimator is able to extract the probability that a student with “learnt” state makes slips. The Threshold-based estimator is decent to estimate the proficiency level of students based on their scores, however, the Threshold-based estimator is not able to track the advancement, which contains the probabilities of learning and forgetting. On the other hand, the BKT models can extract both the probabilities of making slips and lucky guesses, and the probabilities of learning and forgetting. BKT models is able to predict the future scores based on the historical performance, while the Threshold-based estimator is not able to do so.

Objective 2—Measuring the difficulty of concepts. BKT models is able to extract the correlation between “learn” and “not-learnt” state, and the correlation between the mean of score and the variance of score. The variance of score can tell us the probabilities of lucky guesses and slips. The positive correlation between mean score of “learnt” state and mean score of “not-learnt” state derives that harder concepts tend to have higher score with both states. The correlation between mean score and score variance derives the changes of the probabilities of making slips and lucky guesses according to the hardness of concepts. By the experiment results, students seem to make more lucky guesses and fewer slips at easier concepts.

Objective 3—Proposing learning plans for students. The student model is able to estimate the current knowledge state of a students. The domain model can detect that current knowledge state in the surmise graph and find the optimal learning curve to help the student achieve the study target. Refining concepts with the CEFR levels boosts the performance of learnt domain model, which means that the surmise graph of concepts with CEFR seem to be the better guideline of learning curve for students. The Theory-based algorithm has a drawback that this algorithm seems to be not able to distinguish easier concepts and prerequisite concepts. Theory-based algorithm learns that the concept “85—Place of adverbs of time, place and manner” and “89—Place of pronouns in a phrase” are the prerequisite concepts for the others, which seems to be not realistic. GS algorithm extracts better surmise graph with more sensible points. For example, the GS algorithm learnt that concept “24B1—Expressions of time, place and manner. Preposition-free expressions” is the prerequisite of concept “85B2—Place of adverbs of time, place and manner”, which is reasonable (Figure B.1). Figure B.1 presents the learnt surmise with the best performance on evaluation data, which is learnt by GS algorithm on the mastery data estimated by BKT models on concepts with CEFR.

7 Discussion

This thesis indicates that the surmise graph of concepts is a practical alternative to the knowledge state graph to understand the knowledge domain from student data.

The mastery data estimated by Bayesian Knowledge Tracing is better than data estimated by the Threshold-based method for learning the surmise graph, when the domain is the set of concepts with CEFR. BKT boosts the quality of the mastery data only slightly, when the domain is concepts without CEFR. The mean value of $P(F)$, which is the probability for a student could forget a learnt skill after one test session, is 0.4, which is quite high—in both experiments (with and without CEFR). This means that the probability of forgetting play a non-trivial role in tracking the advancement of students. There is some noise that the mean scores of the “learnt” state for some concepts is 1.0, and of the “not-learnt” state for some concepts are 0.0, in the experiment on concepts with CEFR. These cases are the result of some concepts occurring only once in a test session (Figure 6.7). This problem also presents in concepts in the experiment with concepts without CEFR, when the number of questions answered per test session for that concept are small (from 1 to 5 times per test session). This results from the sparsity of scores with only a few possible values. This sparsity causes limitations on the test score data, which impacts on the BKT models’ ability to learn the score distributions. By comparing the mean score and variance of the “learnt” and the “not-learnt” states, we can see the level of difficulty of each concept. Easier concepts tend to have higher mean scores for both states, higher variance of the “not-learnt” state (more lucky guesses) and lower variance of the “learnt” state (fewer slips).

To learn the knowledge structure, the Greedy score-based algorithm is not suitable when the number of variables is high, due to the exponential number of possible structures. Furthermore, this solution seems to overfit regardless of network complexity, since the algorithm keeps adding new edges (or reversing edges, if the graph is fully connected) at all iterations. This behavior possibly results in a fully connected DAG if we did not have an early stopping rule or add a regularization term to the score function, which is based on the number of edges in the network [2]. The GS algorithm gives the best results and its running time is practical.

Applying the GS algorithm on the mastery data estimated by the BKT models on the concepts with CEFR gives the best result when learning the knowledge domain, which has the highest likelihood on the evaluation data. The performance on concepts with CEFR is better than the performance on concepts without CEFR; therefore, considering CEFR levels with concepts is the better way to decompose the domain of learning Russian. The surmise graph, which is obtained by GS algorithm with the mastery data estimated by BKT model on the concepts with CEFR is presented in Figure B.1 (Appendix B).

It is interesting to compare our results with results presented in paper "Modeling language learning using specialized Elo ratings" [12]. That paper reports two knowledge graphs learned by different techniques. We find some similar points between our graph and their graph. For example, we observe that concepts "10", "11" and "12" are positioned very low in all graphs, meaning that they have few prerequisites, and are prerequisite to many other concepts. Similarly concept "85" is a basic concept (having no children/prerequisites) in all of these graphs as well. This kind agreement makes us hopeful that these methods are learning some fundamental properties of the domain.

In the scope of this paper, we have not considered the frequencies of response patterns. Furthermore, assuming that $P(T)$ and $P(F)$ are constant over time is not realistic. For example, the probability of forgetting a skill, which a student has mastered over many consecutive test sessions should not be not the same as for a student who just mastered it over one test session; we would expect that

$$P(F_k|S_{k-1} = 1, S_{k-2} = 1, S_{k-3} = 1) < P(F_k|S_{k-1} = 1, S_{k-2} = 0, S_{k-3} = 0).$$

In future research, we will attempt to take into account the frequency into our solution, as well as experiment with shifting of $P(T)$ and $P(F)$ in BKT models over time. We will also consider solutions to enhance the quality of the data with larger number of answered questions for each concept per test session. The data we used was collected last year, and the Revita platform accumulates more data over time. We believe that the results presented in this thesis can be improved with more data, which would reduce the data sparseness problems. The quality of the models can be further evaluated by actual users. We can measure the performance of the advancement prediction as well as the satisfaction of users on the guideline of learning curve learnt from the domain model.

8 Conclusions

This thesis presents the adaption of Bayesian knowledge tracing model to tracking student's advancement in language learning, as well as the application of the Grow-shrink Markov blanket algorithm, Greedy score-based algorithm and Theory-driven algorithm to learn the network structure for the knowledge domain, based on data from students learning Russian. The result indicates the effectiveness of the BKT model in order to estimate students' mastery state on every skill and of the GS algorithm on recovering the domain structure. We learn the surmise relation among concepts and build a compete network of concepts in the domain.

These surmise connections can be used as guideline for the students to improve the learning curve, while the tracing model can be used to estimate the proficiency levels of students over their test results. The experiments also suggest that decomposing the domain into finer concepts based on their CEFR levels is more effective than ignoring the CEFR levels.

Future research will be focused on extending the BKT model with the flexible probabilities for learning and forgetting a skill, as well as considering the frequency factor of knowledge states. Furthermore, we will try to accumulate more data with high diversity in order to have better experiments.

Acknowledgement

We would like to thank the team of the Revita Project, which conducts research on language learning systems, at the Department of Computer Science, Department of Digital Humanities, and the Department of Modern Languages, University of Helsinki, for their data and support in completing this research project.

Bibliography

[1] D. Albert & J. Lukas. *Knowledge Spaces: Theories, Empirical Research, Applications.* Lawrence Erlbaum Associates, 1999.

[2] S. Beretta, M. Castelli, I. Gonçalves, R. Henriques & D. Ramazzotti. “Learning the structure of Bayesian Networks: A quantitative assessment of the effect of different algorithmic schemes”. In: *Complexity* 2018 (2018).

[3] G. F. Cooper & E. Herskovits. “A Bayesian method for the induction of probabilistic networks from data”. In: *Machine Learning* 9 (1992), pp. 309–347. DOI: 10.1007/BF00994110.

[4] A. T. Corbett & J. Anderson. “Knowledge tracing: Modeling the acquisition of procedural knowledge”. In: *User Modeling and User-Adapted Interaction* 4 (1994), pp. 253–278. DOI: 10.1007/BF01099821.

[5] J. Cussens & M. Bartlett. *GOBNILP 1.6.2 User/Developer Manual.* University of York, 2015.

[6] M. Desmarais & M. Gagnon. “Bayesian Student Models Based on Item to Item Knowledge Structures”. In: *European Conference on Technology Enhanced Learning.* 2006, pp. 111–124. DOI: 10.1007/11876663_11.

[7] J.-P. Doignon & J.-C. Falmagne. *Knowledge Spaces.* Berlin: Springer-Verlag, 1999.

[8] J.-P. Doignon & J.-C. Falmagne. “Knowledge spaces and learning spaces”. In: *New handbook of mathematical psychology* 2 (2016), pp. 274–321.

[9] J.-P. Doignon & J.-C. Falmagne. “Spaces for the assessment of knowledge”. In: *International Journal of Man-Machine Studies* 23.2 (August 1985), pp. 175–196. DOI: 10.1016/S0020-7373(85)80031-6.

[10] J.-C. Falmagne, D. Albert, C. Doble, D. Eppstein & X. Hu, eds. *Knowledge spaces. Applications in education.* 2013. DOI: 10.1007/978-3-642-35329-1.

[11] P. A. Gagniuc. *Markov Chains: From Theory to Implementation and Experimentation.* Hoboken, NJ : John Wiley & Sons, 2017.

[12] J. Hou, K. Maximilian, J. M. Hoya Quecedo, N. Stoyanova & R. Yangarber. "Modeling language learning using specialized Elo rating". In: *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 494–506. DOI: 10.18653/v1/W19-4451.

[13] A. Katinskaia, J. Nouri & R. Yangarber. "Revita: a language-learning platform at the intersection of ITS and CALL". In: *Proceedings of LREC: 11th International Conference on Language Resources and Evaluation*. Miyazaki, Japan, 2018.

[14] D. Margaritis. "Learning Bayesian Network Model Structure from Data". PhD thesis. Pittsburgh, PA 15213: Carnegie Mellon University, 2003.

[15] J. Pearl. "Causal Diagrams for Empirical Research". In: *Biometrika* 82.4 (1995), pp. 669–688. DOI: 10.2307/2337329.

[16] J. Pearl. *Causality*. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511803161.

[17] J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Francisco, CA, United States, 1988.

[18] J. Pearl & T. S.Verma. "A theory of inferred causation". In: *Studies in Logic and the Foundations of Mathematics* 134 (1995), pp. 789–811. DOI: 10.1016/S0049-237X(06)80074-1.

[19] E. Robusto & L. Stefanutti. "Extracting a knowledge structure from the data by a maximum residuals method". In: *TPM - Testing, Psychometrics, Methodology in Applied Psychology* 21 (2014), pp. 421–433. DOI: 10.4473/TPM21.4.4.

[20] B. V. de Sande. "Properties of the Bayesian Knowledge Tracing Model". In: *Educational Data Mining* 5.2 (2013), pp. 253–278. DOI: 10.5281/zenodo.3554629.

[21] J. Schreiber. "pomegranate: Fast and Flexible Probabilistic Modeling in Python". In: *Machine Learning Research* 18 (2018).

[22] M. Schrepp. "A method for the analysis of hierarchical dependencies between items of a questionnaire". In: *Methods of Psychological Research Online* 8 (2003), pp. 43–79.

[23] M. Schrepp. "Extracting knowledge structures from observed data". In: *British Journal of Mathematical and Statistical Psychology* 52 (1999), pp. 213–224. DOI: 10.1348/000711099159071.

[24] P. Spirtes, C. Glymour & R. Scheines. *Causation, Prediction, and Search*. 1993. DOI: 10.1007/978-1-4612-2748-9.

[25] A. Spoto, L. Stefanutti & G. Vidotto. “Knowledge space theory, formal concept analysis, and computerized psychological assessment”. In: *Behavior Research Methods* 42.1 (February 2010), pp. 342–350. DOI: 10.3758/BRM.42.1.342.

[26] L. Stefanutti, D. Albert & C. Hockemeyer. “Derivation of Knowledge Structures for Distributed Learning Objects”. In: *Cognitive Science Section* (2003).

[27] A. Tandon & S. Soni. *Introduction to Artificial Intelligence using Python*. Book Bazooka Publication, 2020. URL: https://books.google.fi/books?id=ExTNDwAAQBAJ.

[28] C. Yuan & B. Malone. “Learning Optimal Bayesian Networks: A Shortest Path Perspective”. In: *Journal of Artificial Intelligence Research* 48 (2013), pp. 23–65.

[29] K. Zhanga & Y. Yao. “A three learning states Bayesian knowledge tracing model”. In: *Knowledge-Based Systems* 148 (2018), pp. 189–201.

Appendix A Concept information table

Id	Concept name	CEFR levels
6	Lexicology. Lexical semantics	'B2', 'B1, 'C1'
7	Collocations	'B1', 'C1'
8	Lexicology. Coordination of words	'C1'
9	Verb. Case government	'A1', 'A2', 'B1', 'B2', 'C1'
10	Verb. Prepositional government	'B1', 'B2', 'C1'
11	Adjectives	'A1', 'A2', 'B1', 'B2'
12	Nouns	'A2', 'B2', 'C1'
20	Predicative adverbs and their government. Existence, state, time	'A2', 'B1'
21	Predicative adverbs and their government. Necessity, possibility, impossibility	'A1', 'A2', 'B2', 'C1'
22	Negative constructions with predicative 'be' (and synonyms), genitive of negation	'B2'
23	Sentences with dative subject: 'Кате 25 лет'	'A1'
24	Expressions of time, place and manner. Preposition-free expressions	'B1', 'C1'
25	Constructions with cardinal numerals	'A2', 'B1', 'B2'
26	Constructions with collective numerals	'B1'
27	Genitive plural of Pluralia tantum words	'A1'
28	I declension. Type 'музеи-музеи, воробеи-воробьи'	'A1'
29	I declension. Type 'санатории'	'A2'
30	I declension. Fleeting vowels and alternations я, е, е / и ('заяц-заица, заем-заима')	'A1'
31	I declension. Type 'карандаш'	'A1'
32	I declension. Type 'адрес-адреа"	'A1'
33	I declension. Type 'солдат-много солдат, сапог-парасапог"	'A1'
34	I declension. Type '-анин/-янин, -ин'	'A2'

Id	Concept name	CEFR levels
35	I declension. Type ‘дерево-деревя”	’A1’
36	II declension. Type ‘армия’	’A1’
37	II declension. Type на -ня	’A1’
38	II declension. Type ‘статья’	’A1’
39	Fleeting vowel in genitive plural	’A1’
40	Nouns with prepositions в/на ending in -у/-ю in prepositional singular	’A1’
41	Nouns ending in -у/-ю in prepositional singular and -а in nominative plural	’A1’
42	Possessive adjectives. Type ‘лисии’	’C1’
43	Ordinal numbers. Type ‘третии’	’B1’
44	Possessive adjectives. Type ‘мамин’	’C1’
45	Cardinal numbers. ‘Сто’ vs. ‘пятьсот, шестьсот, семьсот, девятьсот’’	’B1’, ’B2’
46	Quantifiers. Collective quantifiers in oblique cases	’B1’
47	Quantifiers. Collective quantifiers ‘оба, обе’	’B2’
48	I conjugation. Type ‘плакать’	’A1’
49	I conjugation. Type ‘рисовать’	’A1’
51	II conjugation. Type ‘молчать’	’A1’
52	Preterite. Type ‘исчезнуть’	’B2’
54	Regular verbs with vowel alternation	’A1’
55	Resultative	’B1’
56	Iterative / potential iterative / qualities	’B1’, ’B2’
57	Expression of duration - ‘за какое время’	’B2’
58	Factual meaning of verbs	’A2’, ’B1’
59	Aspect, expression of action completed in the past	’A1’, ’A2’, ’B1’
60	Aspect, expression of capability/incapability. (‘Тебе этого не понимать/понять!’)	’B1’, ’B2’
61	Inception of action	’A2’
62	‘Забыть, успеть, удаться’ + infinitive	’B2’
63	‘Уметь, нравиться, любить’ etc. + infinitive	’A1’, ’A2’, ’B2’
64	‘Пора, скорее’ + infinitive	’B1’

Id	Concept name	CEFR levels
65	‘Нельзя, невозможно, не могу’ + infinitive	’A1’, ’A2’
66	‘Не’ + infinitive	’B2’
67	Negative sentences	’A2’
68	Imperative	’A1’
69	Proximal future	’A2’
70	Impersonal sentences. Infinitival sentences. Subordinate sentences (‘если / прежде чем’ + infinitive)	’B2’
72	Impersonal sentences. Infinitival sentences. Sentences, with negative pronouns and adverbs	’B2’
73	Generis-personal sentences	’B2’
75	Passive and its relation to indefinite-personal sentences (equivalent of Finnish impersonal passive)	’B1’
76	Stress: forms of verbal preterite	’B2’
77	Stress: short adjectives	’B2’
78	Stress: participles	’B2’
79	Nouns. Type A: stress on the stem	’A1’
80	Nouns. Type B, B1, B2: stress on the ending	’A1’, ’B1’, ’B2’
81	Nouns. Type C, C1: stress on the stem in singular, on the ending in plural	’A1’, ’A2’, ’B2’
82	Nouns. Type D, D1: stress on the ending in singular, on the stem in plural	’A1’, ’B1’
83	Nouns. Type ‘нож-ножом, сторож-сторожем’	’A1’
84	Declension of adjectives. Type ‘хорошего’	’A1’
85	Place of adverbs of time, place and manner	’B2’
86	Place of negation in the sentence	’B1’, ’C1’
87	Place of participles in the sentence	’B1’
88	Place of gerunds in the sentence	’B1’
89	Place of pronouns in a phrase	’B2’
90	Word order in sentences introducing direct quotations	’C1’
91	Second person singular imperative in conditionals (‘если’)	’B2’
92	Usage of ‘сам’ and ’один’	’B2’
93	Sentences of type ‘Знаю его как врача	’B1’, ’B2’

Id	Concept name	CEFR levels
94	Sentences of type ‘Быть грозе’	’C1’
95	Sentences of type ‘Лодку унесло вет ром	’C1’
96	Genitive plural	’A1’
97	Frequent prefixed verbs of motion + prepositional constructions	’A2’
98	Animate noun object	’A1’
99	Unprefixed verbs of motion	’A2’
190	Accusative/ergative subject + Impersonal verb: ‘Васю тошнит’	’B2’
191	Prep+genitive subject + Impersonal verb: ‘У меня болит голова/шумит в голове’	’B2’
230	Dative subject + adverb: ‘мне (стало) плохо/нужно/скучно’	’B2’
231	Dative subject + impersonal verb: ‘мне идет/надоело/везет/хватит’	’B2’
232	Dative subject + impersonal-reflexive verb: ‘мне нравится/кажется/пришлось’	’A2’, ’B2’, ’C1’

Appendix B Learnt surmise graph

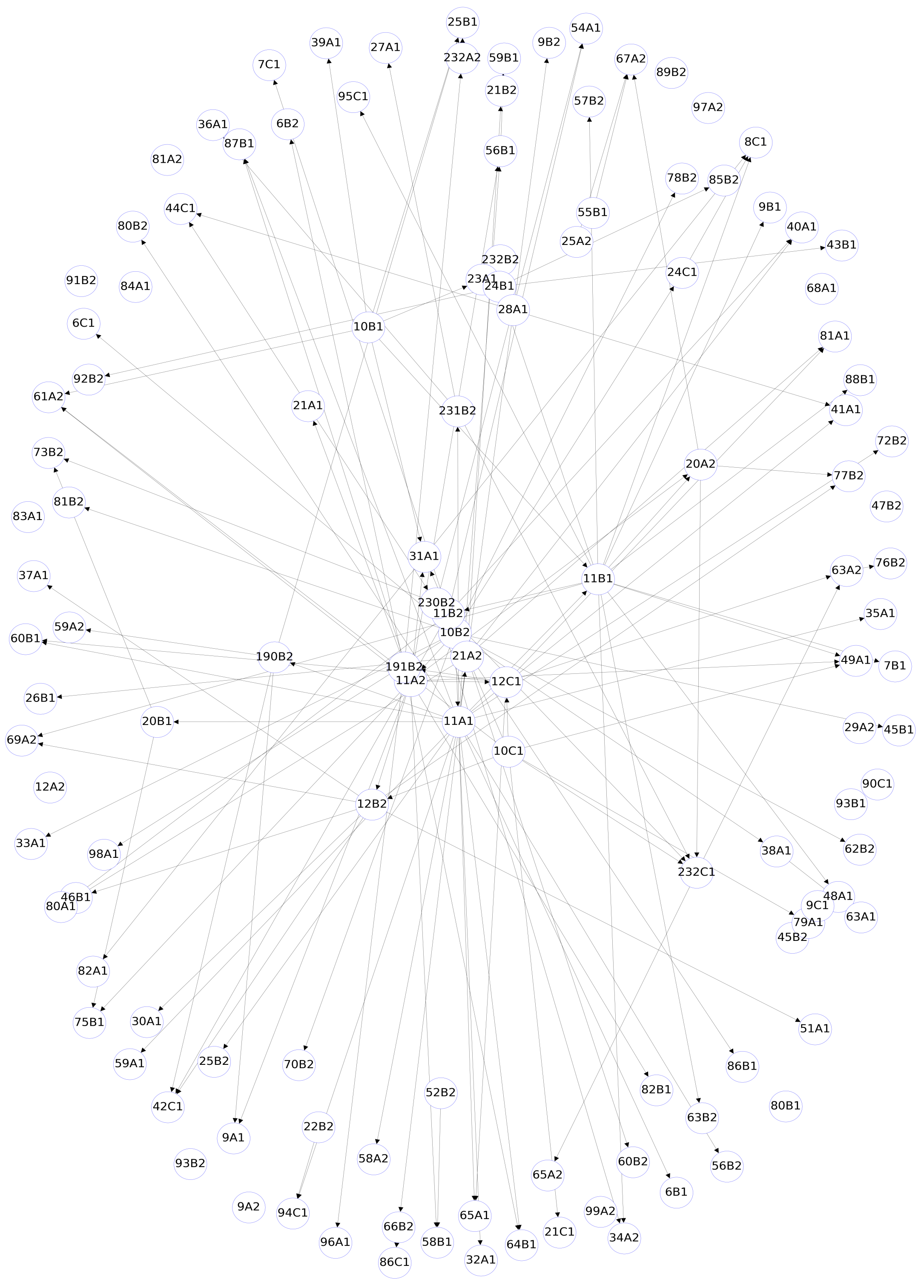

Figure B.1: Surmise graph learnt by BKT and GS algorithm

Figure B.1 is the learnt surmise graph with the best performance on the evaluation data. This graph is learnt by applying Grow-Shrink Markov blanket algorithm on the mastery data, which is estimated by the Bayesian Knowledge Tracing models.

In Figure B.1, the arrows present for the surmise relationship, where $A \rightarrow B$ means A should be learnt before B. The concepts in the middle area are more likely to be the base concepts for the others.