

IMPROVING COMPUTER-ASSISTED LANGUAGE
LEARNING THROUGH HIERARCHICAL
KNOWLEDGE STRUCTURES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Shuhan Wang

May 2019

© 2019 Shuhan Wang
ALL RIGHTS RESERVED

IMPROVING COMPUTER-ASSISTED LANGUAGE LEARNING THROUGH HIERARCHICAL KNOWLEDGE STRUCTURES

Shuhan Wang, Ph.D.

Cornell University 2019

A common drawback in traditional language education is that all students in the same class use the same content. Since students may have different backgrounds such as prior knowledge and learning speed, one single curriculum may not be able to accommodate every student. Unfortunately, most students cannot afford personalized language learning, since preparing personalized learning content can be very time-consuming and potentially requires a significant amount of expert labor. Recently, researchers have proposed automatic systems to assist language education, such as Computer-based Assessment Systems (CAT) and Intelligent Tutoring Systems (ITS). However, previous work usually characterizes the student's knowledge and the difficulty of learning content using numeric scores, which may not be comprehensive.

To improve on this, this thesis introduces hierarchical knowledge structures to assist in multiple tasks in language education. First, this structure multidimensionally characterizes the difficulty of each learning material by its relative difficulty to other materials and models the whole corpus with a graph structure. Additionally, we can utilize the hierarchical knowledge structure to multidimensionally assess a student's prior knowledge, predict the student's future performance on a specific task, and recommend learning content that is appropriate for each student. Furthermore, the hierarchical knowledge structure enables us to build a framework to characterize existing learning curricula extracted from textbooks and online learning tools, and apply expert wisdom that we have discovered to automatically design learning curricula. The hierarchical

knowledge structure reduces the cost of expert labor and potentially makes language education more affordable and more engaging.

BIOGRAPHICAL SKETCH

Shuhan Wang completed his Bachelor of Science at Peking University in 2014. During his time at Peking, he took a course about the cultural geography of the world, which later became the source of his interest in natural languages. While at Cornell University, he minored in Linguistics and primarily worked on Syntax. His Ph.D. research on Computer-Assisted Language Education was greatly inspired by his own experience of learning foreign languages. In addition to Chinese and English, he also speaks Japanese, German and Spanish. He plans to learn more languages in the future.

Dedication

to my ninth cat

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Erik Andersen for numerous ideas regarding my research projects and the guidance of writing research papers. I'm also very grateful to my committee members: John Whitman and Dexter Kozen.

I would also like to thank the collaborators and developers who assisted me with my papers and projects: Fang He, Hao Wu, Ji Hun Kim, Brandon Cohen, Sixian Yi, Jung Yun Park, Nicholas Teo, Liye Zhong, Tong Mu, and Emma Brunskill. I could never finish all of this work without your help.

Special thanks to my best friend Xiang Long for his great help in writing good papers, as well as on living a good life throughout my Ph. D.

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1657176.

TABLE OF CONTENTS

Biographical Sketch	iii
Dedication	iv
Acknowledgements	v
Table of Contents	vi
List of Tables	viii
List of Figures	x
1 Introduction	1
2 Related Work	10
2.1 Intelligent tutoring systems	10
2.2 Difficulty evaluation of reading texts	11
2.3 Trace-based partial orderings	13
2.4 Knowledge assessment: IRT, KST, and CAT	13
2.5 Educational recommender systems	15
2.6 Practice problem ordering and progression synthesis	16
2.7 Forgetting curves and reinforcement scheduling	17
2.8 Student engagement in educational games	17
3 Organizing Learning Materials	18
3.1 Conceptual units in grammar learning	19
3.1.1 Grammatical template analysis	20
3.1.2 Statistics of grammatical templates	25
3.1.3 Difficulty level prediction	26
3.2 Hierarchical knowledge structure	31
3.2.1 Partial ordering graph	31
3.3 Scaling to vocabulary learning	35
3.3.1 Fuzzy partial orderings	36
3.4 Conclusion	40
4 Student Assessment and Adaptive Recommendation	42
4.1 Knowledge assessment	43
4.1.1 The knowledge boundary	43
4.1.2 Calculating the knowledge boundary	44
4.1.3 Predicting performance on specific problems	46
4.1.4 Evaluation	50
4.2 Incorporating assessment into recommendation	55
4.2.1 Adaptive assessment heuristic	56
4.2.2 ZPD-based recommendation heuristic	57
4.2.3 Balancing assessment and recommendation	58
4.2.4 Evaluation	59
4.3 Conclusion	62

5	Learning Progression Analysis and Design	64
5.1	Progression analysis of expert-designed curricula	65
5.1.1	Composition: the balance of learning and review	65
5.1.2	Pace: the growth rate of knowledge	67
5.1.3	Studies on textbooks	68
5.1.4	Studies on online language learning tools	71
5.2	Parameter-based progression synthesis	73
5.3	Goal-based progression design	75
5.3.1	The two-step heuristic approach	78
5.3.2	Evaluation	83
5.4	Conclusion	92
6	Conclusion	94
6.1	Future work	96
	Bibliography	99

LIST OF TABLES

3.1	Grammatical Templates in Japanese, with hyphens denoting words to be filled in. Note that some grammatical templates may impose requirements of some properties (e.g. part of speech or form) on the missing words.	22
3.2	Identified grammatical templates of Japanese sentences. In sentences, pronunciations and translations, grammatical templates are in bold. The word <i>toukyou</i> in the first sentence is a noun (Tokyo, 東京), as characterized by template E. The word <i>mi</i> (to see, 見) in the second sentence is the i-form (動詞連用形) of a verb, as required by template F.	22
3.3	Distribution of grammatical templates of level N1(hard)-N5(easy) . . .	25
3.4	Number of templates of level N1(hard)-N5(easy) per 100 sentences . .	26
3.5	Accuracies of classifying N1-N5 texts	30
3.6	Traces of Integer Addition Problems	32
3.7	Japanese Sentences and Their Grammatical Templates	33
3.8	Sample Sentence Pairs in the fuzzy partial orderings with the fuzzy parameter $\alpha = 0.9/0.8/0.7/0.6$. For each fuzzy parameter α , the second sentence is α -fuzzily harder than the first sentence. As the fuzzy parameter α decreases, our confidence in the fuzzy partial orderings (the likelihood for a student to understand the first sentence if he/she understands the second one) also drops. Text source: NHK Easy [63].	39
4.1	Scoring User Responses in the Assessment Stage of J100	52
4.2	The average and standard deviation of J100 user responses, grouped by the calculated distance from the knowledge boundary. Note that the standard deviation of the user responses is maximized where the distance is at some point between -2 and -3, indicating that users are most unsure of their ability at this distance.	53
4.3	We ran Wilcoxon Rank-sum tests for all pairs of our four groups: Adaptive Recommendation (A.R.), Non-adaptive Recommendation (N.R.), Assessment-Only (A.O.) and Random (Rand.). The difference between adaptive recommendation and non-adaptive recommendation was statistically significant ($p = .035$).	61
5.1	Synthesized progressions with desired parameters. The ‘ <i>Genki Simulation</i> ’ progression is synthesized with parameters extracted from the textbook. The other three progressions are generated with specific parameter settings. This demonstrates the potential of our progression synthesis framework to map characteristics of one progression onto another, and to create progressions that are tailored to the needs of an individual student.	75
5.2	First eight tasks of Language Zen [111], an online Spanish learning tool (as of December 2018).	76

5.3	Sample Results of Algorithm 2, which scales a reinforcement-free progression into longer and slower progressions given the minimum memory retention parameter <i>minR</i> . Levels in italics are added by the algorithm to help student review. The parameter <i>minR</i> controls the progression length (learning pace): the larger the parameter <i>minR</i> , the longer the target progression.	82
5.4	Expert progression and synthesized progression in <i>Katchi</i> . Goal levels are in bold. Note that the expert progression puts all goal levels at its end whereas synthesized progression introduces goal levels starting at level 15 and reinforces them afterwards.	87

LIST OF FIGURES

3.1	Grammatical difficulty in the N1/N2 texts	27
3.2	Grammatical difficulty in the N4/N5 texts	28
3.3	Multilevel Linear Classification (MLC). ‘>N5?’ represents the linear classifier judging whether a text is harder than N5. The classifiers are similar for the other levels.	28
3.4	A sample partial ordering graph. Uppercase letters represent concepts and the nodes with strings represent problems. Each directed edge from problem s_i to s_j represents s_j is directly harder than s_i	35
3.5	Number of edges in the fuzzy partial ordering graph over different fuzzy parameters, in our corpus of 4,269 texts. Decreasing the fuzzy parameter will increase the graph density exponentially.	38
4.1	A Student’s Knowledge boundary in the partial ordering graph. Uppercase letters represent concepts and the nodes with strings represent problems. The student can solve the green problems but cannot solve the red ones. The distances from each problem to the knowledge boundary are indicated as d . The null problem s^0 is an auxiliary problem that is added to help us compute the distances.	44
4.2	Coloring the partial ordering graph based on the student’s response. If the student solves s^* , all the nodes that are at most as hard as s^* (including s^* itself) will be colored ‘solvable’ (green); if the student fails to solve s^* , all the nodes that are at least as hard as s^* (including s^* itself) will be colored ‘unsolvable’ (red).	45
4.3	Screenshot of J100 Platform (Assessment Stage)	51
4.4	The adapted Rasch model nicely fits the J100 user data. The blue data points show the average user response to problems with the same distance from the knowledge boundary, and the red curve is the <i>Item Characteristic Curve</i> (ICC) calculated based on the adapted Rasch model.	54
4.5	Screenshot of <i>JRec</i> , a Japanese reading text recommendation tool. When using this tool, users are directed to an NHK Easy webpage [63], read a recommended text, and respond whether or not they understand it. Our tool highlights the recommended text and grays out the rest of the webpage.	60
4.6	Proportion of users remaining after reading certain amount of texts. We observed that users in the adaptive recommendation group read 62.5% more texts than those in the non-adaptive recommendation group, which indicates that incorporating adaptive assessment significantly improved student engagement in learning material recommendation.	61
5.1	Progression composition of two Japanese textbooks: <i>Standard Japanese</i> and <i>Genki</i> . The proportions are fairly similar. Note that it usually takes a student 2 semesters to learn units 1-6.	68

5.2	Proportion of reinforcement(RI), recombination(RC) and Introduction(IN) in two series of Japanese textbooks: <i>Standard Japanese</i> (SJ) and <i>Genki</i> (GK). Here x-axis represents the learning progress: unit 1-6, and y-axis demonstrates the proportions.	69
5.3	Paces of various progressions. Here, the x-axis represents the normalized learning time (namely, the normalized number of sentences learned), and the y-axis corresponds to the normalized knowledge size. (a) shows that both textbooks introduce the same amount of knowledge over the same period of time, which indicates that they use a steady pace. (b) and (c) compare the textbook progression to other baseline progressions. When compared with the textbook progression, random progressions are too difficult at the beginning, and progressions generated by a topological sort are too easy at the beginning.	70
5.4	Progression composition of two online Spanish learning tools: Duolingo and Language Zen. Compared with Duolingo, Language Zen has less reinforcement and less introduction, but much more recombination. Note that it usually takes a student a couple of days to learn 30 problems.	72
5.5	A sample partial ordering graph in goal-based progression synthesis. Each node represents a level containing a specific set of required concepts. Each directed edge represents a “harder than” relation between two levels. The red numbers next to each node denote the traversal order of each level in the post-order DFS, namely the order of each level in the reinforcement-free progression.	79
5.6	Screenshots of <i>Katchi</i> , a Korean learning puzzle game that teaches basic vocabulary such as colors, numbers, shapes and conjunctions. There are two types of game levels in <i>Katchi</i> : <i>selection levels</i> , in which players need to either select the correct graphic object based on the given Korean word(s); <i>construction levels</i> , in which players need to drag-and-drop the correct Korean word(s) to describe the target objects.	84
5.7	Proportion of users remaining after playing our game for certain amounts of time. We did not find a statistically significant difference between the two progressions. It appears that the synthesized progression engaged players slightly less than the expert progression, but they are quite comparable. The median player in the synthesized progression played for 42 seconds and the median player in the expert-designed progression played for 45 seconds.	88

5.8	Proportion of users remaining after learning certain amounts of knowledge. We use <i>knowledge size</i> , number of levels that either have been completed or are easier than some completed levels, to measure the amount of knowledge learned. There is no statistically significant difference between the two progressions. The proportion of players with knowledge size between 10 and 20 is higher for the expert progression, while the two curves are quite similar otherwise. The median knowledge size is 5.5 for the synthesized progression and 5 for the expert-designed progression.	90
5.9	Total game progress over different average times spent on each level. Each blue plot represents a player. Typically, players who finished at least 90% of the game (green zone) spent 7-12 seconds on each level. Players who spent less than 7 seconds (red zone) or more than 12 seconds (orange zone) tended to leave early before 90% of the game, which probably indicates that we should generate a shorter/longer progression to accommodate their faster/slower learning pace respectively.	92

CHAPTER 1

INTRODUCTION

A common drawback of traditional classroom education is that all students in the same class use the same content, but a single series of curricula cannot accommodate students of different backgrounds, such as different prior knowledge or different preferred learning speed. Although most textbooks are carefully designed by experienced teachers and experts in order to fit most students' needs (the "general" learning content), some students may still find that the textbooks introduce too much content in a certain period of time (too fast) or the textbooks are far beyond their current level (too hard). If these students do not receive help promptly, they may lose interest in learning, or even quit school, which I believe is not a rare case around us. Unfortunately, most schools, especially in developing countries and areas, typically do not have enough teachers dedicated to each student.

Ideally, we would have computers do some of the work for teachers, such as collecting materials, assessing students, and select appropriate content for each student. The most important advantage of computer-assisted education is "automatic": the entire pipeline of traditional education is streamlined and integrated into a unified computer system, in which much of the expert labor is saved, the time and monetary costs are reduced, and thus personalized education can be affordable for everyone. By learning personalized content that is highly adapted to individual needs, students can achieve better learning effectiveness and be more engaged. Spending the same amount of time and money, people can potentially explore a wider range of topics or dive deeper in the direction they are interested in with the help of computer-aided education. In short, computer science techniques have a good potential of boosting traditional education in various perspectives.

Although most models and algorithms in this thesis can be naturally applied to other educational domains, my Ph.D. work mainly focused on learning natural languages. I believe learning a foreign language is important because as an international student, I can see a variety of people around me want or need to learn a second language: students who want to study abroad, kids who like watching Japanese animations, or people like me who simply enjoy learning multiple languages. Although my knowledge of foreign languages (other than English) may not necessarily contribute significantly to my future career (it did help while curating source data or interim results in some Natural Language Processing projects), learning a foreign language brings a lot of joy to me and many of my friends. In fact, many people may not have enough time or money to go to a professional language class. Instead, they rely more on online language learning. Therefore, I would hope to improve computer-assisted language learning, in theory and in practice, to provide them with a better experience of learning a foreign language.

Researchers have developed many theories and frameworks to help language students. Krashen's *Input Hypothesis* proposes that students gain the knowledge of a language when they try to understand the content that is slightly beyond his/her current level (this is also known as "i+1 theory") [48]. Vygotsky's *Zone of Proximal Development* (ZPD) suggests that students are able to finish the tasks that are just beyond their ability with external assistance [94]. Both of these well-established theories suggest that Computer-Assistant Language Learning (CALL) systems need to target the content that is one step beyond a student's current knowledge. To do this, a CALL system needs to organize the corpus of available learning materials based on their difficulty and evaluate the prior knowledge of each student.

Item Response Theory (IRT) is one of the most widely-used tools for student assessment. IRT stipulates that a student's performance on a specific problem (item)

is a function of the difference between the student's ability and the problem's difficulty [25, 27, 72]. This function typically has one to three parameters that need to be calculated based on a large amount of data of the student's previous performance on the problems in the library (For example, the 3PL model [56] and the Rasch model [71]). *Knowledge Tracing* (KT) is also an important theory that has been applied to many intelligent tutors [19, 45]. KT models the process of a student's learning a new concept with a Hidden Markov Model, and the probability of the student's transitioning from the "unmastered" state to the "mastered" state is a parameter specified by the content that needs to be calculated based on previous student data. However, these two theories may encounter significant issues when applied to online language education, since students favor authentic and up-to-date materials yet the data of the student's previous performance on those materials is often missing. As a result, they are not able to compute their required parameters in order to characterize those materials. In addition, these two theories normally use unidimensional numeric scores to characterize the difficulty of learning materials and the ability of students. However, knowledge is not unidimensional: materials with difficulty level 8 may require different sets of knowledge, students with competency level 7 may understand different sets of knowledge too, and we cannot be sure that any level-8 material is "just-beyond" the knowledge of any level-7 student. Therefore, we need a more comprehensive way to characterize learning materials and assess students. Ideally, we would build a *hierarchical knowledge structure* that records all the information of which materials are harder (or even "just harder") than some other materials (namely, the relative difficulty between materials), and characterize a student's knowledge within this hierarchical knowledge structure using the set of materials that he/she understands. By doing this, we can easily recommend each student with the appropriate materials that are slightly harder than his/her ability.

There are also studies that help students to learn new knowledge over a long-term

period and monitor how they forget things throughout the learning process. Ebbinghaus' *Forgetting Curve* is an old but very famous theory that models memory and forgetting. It stipulates that a student's memory decays exponentially over time [26]. More recently, researchers have presented a series of work on scheduling reinforcement: Leitner introduced a flashcard system in which previously learned knowledge is reviewed at increasing intervals [31]. Pavlik and Anderson examined different strategies to optimize schedules of reinforcement [66,67]. Wozniak proposed the *SuperMemo* algorithm to calculate intervals of time between multiple repetitions of the same learning content [104], which has been applied to several computer-assisted learning tools such as Anki [7] and Mnemosyne [59]. However, a common drawback of these learning tools is that they are flashcard-based, which means students can only learn a vocabulary word or a phrase at a time without any context, and students may quickly get bored of this. Ideally, a good CALL system should recommend a sequence of authentic articles instead of a series of individual words/phrases in order to better engage students, and my work seeks to do this based on existing theories of forgetting.

This thesis aims to improve existing computer-assisted language learning systems in three aspects: materials, students, and curricula. The first research problem I would like to tackle is how to comprehensively characterize and organize a corpus of learning materials. Language learners want to read authentic and up-to-date texts with a variety of topics, but it is prohibitively expensive to ask experts to evaluate the difficulty of a large number of texts, and existing data-driven approaches (such as IRT [92, 96, 101, 109]) will also fail to work when we do not have sufficient student data. Moreover, previous automatic approaches usually measure the difficulty of learning materials using unidimensional numeric scores [34, 56, 71], which is not comprehensive [28]. In Chapter 3, I will propose a novel way to organize a corpus of learning materials: we build the *hierarchical knowledge structure* for the corpus, in which the difficulty of a material

is measured based on its relative difficulty to other materials. This structure is also based on the decomposition of vocabulary and grammatical knowledge and determines whether one material is harder than another by decomposing them into sets of conceptual units and then comparing them. Specifically, we model the hierarchical knowledge structure for a given corpus using the *partial ordering graph*, and this graph can be generated automatically without requiring expert labor or student data. This graph will also be a powerful tool for the assessment of a student's knowledge and the synthesis of a curriculum, which I will show in the following two chapters.

The second problem I want to address is how to recommend learning materials to students based on their current language levels. Although we have already seen some material recommendation systems that proved to be very useful in classrooms [17, 39, 41], it can be very different for the adults who are only able to spend their spare time learning a foreign language online: in formal learning scenarios such as schools and universities, we can assess a student's current language level using a standardized language placement test, yet in informal learning scenarios such as online learning, those tests are usually not feasible, and self-assessment results are often inaccurate and standardized [57]. In Chapter 4, I will present an interactive assessment algorithm to comprehensively evaluate a student's prior knowledge within the hierarchical knowledge structure (partial ordering graphs), as well as an evaluation demonstrating that this framework can produce accurate assessment results. I will also show that this algorithm can be incorporated into our text recommendation algorithm in order to make adaptive recommendations and increase student engagement.

The third research problem I worked on is how to synthesize and optimize a sequence of learning materials, namely, the *learning progression*. Since it is reported that preparing even an hour of learning content can take experts hundreds of hours [2, 3], we hope

to find automatic approaches to designing curricula in order to make language learning more affordable. Although previous researchers have proposed several principles to decide in which order we should present practice problems to students [4, 48, 51, 76, 94], yet it is still unclear how to apply these principles in practice. In Chapter 5, I will try to bridge this gap by introducing a framework to characterize existing expert-designed learning progressions extracted from both traditional textbooks and online language learning tools, as well as automatic approaches to synthesize progressions that are similar and comparable to the progressions created by experts.

Thesis Statement:

By building hierarchical knowledge structures of language learning content, automatic techniques can assist in multiple critical tasks in language education. Specifically, we can i) multidimensionally measure the difficulty of learning materials and organize a corpus of learning content with a graph structure; ii) comprehensively assess a student's knowledge and accordingly recommend content of appropriate difficulty; and iii) discover general principles of how experts design curricula by analyzing existing textbooks and learning tools, and apply those principles to synthesize sequences of learning materials for students to learn gradually. This will potentially reduce the cost of expert labor and make language learning more affordable and more engaging.

In Chapter 3, I will propose the *partial ordering graph* to model the hierarchical structure of grammatical and vocabulary knowledge within a corpus. Based on knowledge decomposition, this model characterizes each learning material as a set of its required conceptual units, and then measures the difficulty of a material based on its relative difficulty to other materials in the corpus, rather than unidimensional numeric scores. Moreover, this chapter will provide more technical details about how to apply this model to grammar and vocabulary learning. I will introduce *grammatical*

templates (units of grammar that professional language instructors identified and specifically teach in language classes) as the conceptual units in grammar learning such that grammatical knowledge can be decomposed. The experiment on the Japanese Language Placement Test (JLPT¹) corpus demonstrates that these templates can be important signals to predict text difficulty. I will also present the *fuzzy partial ordering graph*, a modification of the partial ordering graph, in order to model the vocabulary knowledge within a large online corpus without requiring any expert labor or student data. The modified model relaxes the constraints that specify whether one material is harder than another (relative difficulty) and increases the density of vocabulary knowledge structure while retaining reasonably high confidence in those “harder-than” relations.

In Chapter 4, I will demonstrate how (fuzzy) partial ordering graphs enable an accurate assessment of the student’s prior knowledge and an adaptive learning material recommendation approach that is more engaging. I will propose the *Knowledge Boundary*, the set of hardest solvable problems, to model a student’s knowledge and present an interactive algorithm to calculate a student’s knowledge boundary within a partial ordering graph. The user data of J100, an online Japanese knowledge assessment tool, verifies that this algorithm can produce accurate assessment results, and a student’s future performance on a problem can be estimated based on the distance from that problem to the knowledge boundary in the partial ordering graph. I will also introduce an adaptive learning material recommendation approach that incorporates this assessment algorithm. This approach uses a probabilistic function that balances assessment and recommendation in order to avoid an excessive amount of assessment or too much inappropriate recommendation. The experimental results of JRec, an online Japanese reading text recommendation tool, demonstrate that the adaptive recommendation approach is more engaging than the non-adaptive version, indicating that adding assessment can

¹<http://www.jlpt.jp/e/about/message.html>

significantly improve student engagement in learning material recommendation.

In Chapter 5, I will present how to analyze existing curricula and automatically synthesize curricula within the partial ordering graph. We specify a curriculum as a *learning progression*, an optimal sequence of learning materials that students can learn gradually and smoothly. By analyzing expert-designed progressions from texts and online language learning tools, we found that a good progression should have a steady pace, a good balance of introducing new knowledge and reviewing previously learned knowledge, and should build up to some harder tasks as soon as possible (“*goal-driven*”) in order for students to feel a sense of accomplishment. I will also demonstrate the possibility of synthesizing a learning progression given pacing/proportion parameters, by presenting a simple, greedy-based algorithm. Finally, I will propose a goal-based progression synthesize algorithm that builds up to harder tasks as soon as possible while helping students to review as well. The user study of Katchi, a video game that teaches basic Korean vocabulary, demonstrates that the synthesized progression is comparable to an expert-created progression in terms of both engagement and learning effectiveness.

Some of this work was originally presented in the following papers:

- *Grammatical Templates: Improving Text Difficulty Evaluation for Language Learners* [97] (in collaboration with Erik Andersen, published at The 26th International Conference on Computational Linguistics (COLING), 2016);
- *A Unified Framework for Knowledge Assessment and Progression Analysis and Design* [99] (in collaboration with Fang He and Erik Andersen, published at The ACM Conference on Human Factors in Computing Systems (CHI), 2017);
- *Adaptive Learning Material Recommendation in Online Language Education* [100] (in collaboration with Hao Wu, Ji Hun Kim and Erik Andersen, to appear at The 20th International Conference on Artificial Intelligence in Educa-

tion (AIED), 2019);

- *Goal-based Progression Synthesis in a Korean Learning Game* [98] (in collaboration with Brandon Cohen, Sixian Yi, Jung Yun Park, Nicholas Teo and Erik Andersen, to appear at The 14th International Conference on the Foundations of Digital Games (FDG), 2019).

CHAPTER 2

RELATED WORK

In this section, I will present related work on educational technology and language learning. I will first discuss previous studies on intelligent tutoring systems (section 2.1), then introduce the existing work on text difficulty evaluation (section 2.2) and the measurement of task difficulty in other educational domains such as Mathematics based on partial orderings (section 2.3). Subsequently, I will introduce several existing theories and ideas of knowledge assessment (section 2.4) and educational recommender systems (section 2.5). Finally, I will present previous work regarding curriculum design: practice problem ordering and progression synthesis (section 2.6), reinforcement scheduling (section 2.7), and engaging students in educational games (section 2.8).

2.1 Intelligent tutoring systems

There are successful adaptive learning systems such as Cognitive Tutors [6]. Some of this work has focused on language learning specifically [105, 112]. Cognitive Tutors utilize knowledge tracing [19] to track knowledge acquisition and provide tailored instruction, by tracking performance on individual production rules in a cognitive model [19, 45]. This model has been extended in several ways, including estimation of the initial probability that the student knows a skill [64], estimation of the impact of help features on probability of acquisition [9], and integrating with models of item difficulty [65]. Another method is logistic regression, which is particularly efficient for tasks involving multiple skills [108]. However, these approaches typically do not consider pacing. Instead, they continually give problems that exercise a specific production

This chapter was re-organized and rewritten based on the “related work” sections of the four papers mentioned at the end of Chapter 1.

rule until a Hidden Markov Model has achieved 95% confidence that the student has learned that rule, and then move on to the next concept. Furthermore, Cognitive Tutors are difficult to construct. It has been estimated that as much as 200-300 hours of expert design effort are required to design a single hour of content [2], although newer design techniques have reduced this to 50-100 hours [3]. To reduce the expert labor further, this thesis seeks to develop automatic ways of analyzing and recommending learning materials, and optimizing learning progressions (sequences of learning content).

2.2 Difficulty evaluation of reading texts

Text difficulty evaluation has been widely studied over the past few decades [32, 35, 42, 61, 83, 84]. Researchers have developed over 200 metrics of text difficulty [18]. For example, *Lexile* measures text complexity and readability with word frequency and sentence length [85]. *ATOS* [78] includes two formulas for texts and books, both of which take into account three variables to predict text difficulty: word length, word grade level and sentence length. *TextEvaluator* is a comprehensive text analysis system designed to help teachers and test developers evaluate the complexity characteristics of reading materials [82]. It incorporates more vocabulary features, such as meaning and word type, as well as some sentence and paragraph-level features.

Nevertheless, most of these methods provide limited consideration of grammatical difficulty, which is a major challenge for foreign language learners [14]. In fact, text readability not only depends on sentence lengths or word counts, but on ‘the grammatical complexity of the language used’ as well [79]. Based on this fact, recent readability evaluation systems improved performance by incorporating syntactic features like parse tree depth [81] and subtree patterns [36] to measure grammatical complexity. Moreover,

researchers have developed an unified framework of text readability evaluation, which combines lexical, syntactic and discourse features, and predicts readability with outstanding accuracy [70]. The relationship between text readability and reading devices was also studied in the past two years [44]. However, most of these approaches are intended for native speakers and use texts from daily news, economic journals or scientific publications, which are too hard to read for beginning and intermediate language learners. Ideally, we would have specific features and approaches for text difficulty evaluation for language learners.

Recently, language educational researchers conducted a bunch of studies on text readability evaluation for language learners in different languages, such as English, German, Portuguese and French [10, 29, 58, 91, 95, 106]. However, they use traditional syntactic features such as sentence length, part of speech ratios, number of clauses and average parse tree height, which differ from the grammatical knowledge that students actually learn in language lessons. For example, Curto et al. measured text difficulty using traditional vocabulary and syntactic features, to predict text difficulty levels for Portuguese language learners [20]. Unfortunately, 75% accuracy in 5-level classification with 52 features is not satisfactory. Instead, we extract grammatical features from *grammatical templates*, the knowledge units that language students actually learn in classes and that expert language instructors have identified and highlighted in textbooks. We also propose a novel technique that has a simpler and human-interpretable structure, uses only 5 grammatical template features, and predicts text difficulty with 87.7% accuracy in 5-level classification (section 3.1).

2.3 Trace-based partial orderings

Andersen et al. proposed a technique for automatically exploring the space of task progressions through static analysis of the procedure to be learned [4]. This technique characterizes tasks by analyzing the *execution trace* obtained by running the procedure on that task. By characterizing each task as a sequence of basic operations, one can specify a *partial ordering* that ranks the difficulty of problems. This partial ordering has been experimentally confirmed to match well with users' perceptions of difficulty in an educational algebra game [4]. The trace-based framework has been applied to math [4], video game level design [12], and teaching the Thai alphabet [5]. This work also used test-input generation tools like Pex [89] and FShell [38], which systematically create test case suites with high code coverage, to generate problems for all possible traces (within certain bounds). However, this framework cannot be applied to non-procedural topics. For example, it is unclear how to analyze the execution trace of how a human understands natural language. We build on this work by proposing a general framework of problem decomposition and organization, which can be applied to non-procedural educational domains such as language learning (Chapter 3). Within this framework, we can measure a student's ability and predict the student's performance on new problems (section 4.1).

2.4 Knowledge assessment: IRT, KST, and CAT

Item Response Theory (IRT) provides a framework for knowledge assessment [25, 27, 72]. IRT argues that the probability of a correct response to an item is a function of item parameters and individual ability [34]. Lord et al. proposed the 3PL model [56], which takes three item parameters into consideration: item difficulty, item discrimina-

tion, and the probability of guessing. A simpler model was proposed by Rasch [71], which stipulates that the probability of a correct response is determined only by the difference between the student's ability and the difficulty of the item. However, a common drawback of IRT models is that they measure student ability and item difficulty with unidimensional numeric scores [28], which does not reflect that students may find varying subsets of problems to be difficult depending on what they have mastered. We propose a novel way of measuring the difference between a student's ability and the difficulty of a problem that captures such variations (Chapter 3). We validate this model with user data collected from an online knowledge assessment platform (section 4.1).

IRT is also a crucial tool in Computerized Adaptive Testing (CAT) [92,96,101,109]. CAT uses IRT to select the items that can best discriminate examinees and updates the estimate of examinee abilities according to their responses. Both IRT and CAT characterize an item by statistically analyzing large amounts of student responses. However, this does not apply to the fresh materials in online learning due to the lack of sufficient student data. In contrast, our work measures the difficulty of online learning materials by studying the compositionality of domain knowledge and building the hierarchical knowledge structure within the corpus. By doing this, our system is able to leverage fresh learning materials from the Internet, and make appropriate recommendations for each student (Chapter 4).

Knowledge Space Theory (KST) is a well-established perspective for studying the hierarchical structure of knowledge and a powerful tool for knowledge assessment [23,24]. There are several tutoring systems based on KST, such as Alexs [28], RATH [37] and one for learning organic chemistry [88]. According to KST, a student's knowledge is represented as a knowledge state, the set of the problems that the student can solve. Problems are organized into a knowledge structure, which contains all

possible knowledge states as well as the connections between these states [1,46,47]. Researchers have proposed approaches for building knowledge structures, such as querying experts [47] and Bayesian inference [22]. Due to the complexity of representing a knowledge state, Falmagne et al. introduced the idea of a “fringe” to characterize a student’s knowledge, which can be calculated by an “entropy”-based approach [28]. We build on this work by proposing an automatic framework that can decompose a problem into its prerequisite basic skills and builds the hierarchical structure for a set of problems. Then, our framework can predict a student’s performance on new problems by measuring the relationship of the student’s ability to the problem within this structure.

2.5 Educational recommender systems

Researchers have developed many Educational Recommender Systems (ERS) based on students’ prior knowledge [17], topics of interest [39] and learning styles (e.g. verbal/visual, active/reflective) [41,50], However, most of these ERS systems are designed for formal learning scenarios, such as learning in universities. In formal learning, materials are measured and organized with well-defined structure or metadata by experts [50], and students are characterized with standard pre-assessments (for prior knowledge) [17] or pre-questionnaires (for learning preferences such as topics of interest and learning style) [39,41]. However, in informal scenarios such as online learning, a huge amount of learning materials cannot be manually structured and indexed with domain concepts and metadata (the “open corpus problem”) [11], and the modeling of students is either lacking or unstandardized [57]. This work seeks to address these issues in online learning. Our recommender system automatically organizes the learning content from the Internet into a hierarchical model and incorporates adaptive assessment into the recommender system in order to improve student engagement (section 4.2).

2.6 Practice problem ordering and progression synthesis

Researchers have discovered several principles to order practice problems or domain knowledge to introduce to students. Krashen’s Natural Order hypothesis and Input hypothesis propose that language acquisition occurs when a learner receives a language input that is “one step beyond” his/her current competency [48]. Vygotsky’s Zone of Proximal Development (ZPD) suggests that students can complete the tasks that are slightly harder than what they can finish alone with guidance [94]. Reigeluth and Stein’s Elaboration Theory stipulates that students should experience the easiest problem first, followed by gradually harder problems [76]. Li et al. conducted a case study with a machine learning agent and argues that interleaved problem orderings are more efficient than blocked orderings [51]. However, we still lack pragmatic approaches that can apply these principles in educational practice, and our work seeks to do this.

There are also studies on automatically synthesizing level progressions for educational games. Andersen et al. created hundreds of level progressions for an algebra learning game using software testing tools to generate execution traces [4]. However, the synthesized progressions were not practically tested by human players. Butler et al. proposed an automatic approach of designing game level progressions for mathematics education [12,13]. This work characterizes each problem (game level) by the n-grams in the execution trace of its solution procedure, assigns a cost to each n-gram, and selects the level with the smallest total cost as the next level. However, it fails to consider the relationship between problems or the hierarchical knowledge structure within the corpus. Additionally, it does not help students review what they have learned in the progression. We build on this work by proposing a two-step heuristic algorithm for progression synthesis. It runs a post-order Depth-First Search (DFS) in the hierarchical knowledge structure in order to introduce new problems with increasing difficulties. It also adds an

appropriate amount of reinforcement into the synthesized progression to help students review and accommodate to their desired learning pace (Chapter 5).

2.7 Forgetting curves and reinforcement scheduling

Researchers have also studied how students forget the knowledge they have learned and how to help them review appropriately. Ebbinghaus' Forgetting Curve stipulates that memory retention is exponential to the ratio between the negative of time and memory strength [26]. The Leitner System suggests that previously learned knowledge should be reviewed at increasing intervals [31]. Pavlik and Anderson examined different strategies of spaced repetition and successfully optimized schedules of reinforcement [66,67]. We build on this work by incorporating reinforcement into the synthesis of progressions of gradually increasing difficulty. We leverage these ideas to decide which problems to reinforce and when to reinforce them in our synthesized progressions.

2.8 Student engagement in educational games

Generally, people are less obliged to learn in informal learning scenarios such as playing educational games. Therefore, game designers and researchers have paid close attention to increasing student engagement and studied on multiple factors that lead students to play longer, such as the difficulty of the tasks [53, 54] and the novelty of game content [55]. Some have worked on synthesizing level progressions that better engage students [12]. This thesis builds on the former work by using student engagement, such as the number of texts read and the time played, as the primary metric to evaluate the material recommender system and the synthesized progression in user studies.

CHAPTER 3

ORGANIZING LEARNING MATERIALS

A student gains the knowledge of a language while experiencing content that is slightly more advanced than his/her current level (Krashen's *Input Hypothesis* [48], also known as "i+1" theory), and that student is also able to understand those slightly harder content with external assistance (Vogotsky's *Zone of Proximal Development* [94]). Both of these two well-established theories suggest Computer-Assisted Language Learning (CALL) systems should target those learning materials that are just beyond a student's prior knowledge. Based on this idea, a CALL system needs to first compare the relative difficulty of learning materials and judge which materials are harder (or even "just harder") than some other materials in order to organize a corpus. Previous intelligent tutoring systems normally use numeric scores to evaluate the difficulty of learning content (such as [56, 71]), and the relative difficulty can be simply judged by comparing two numbers. However, this is usually not comprehensive [28], since knowledge is not unidimensional. For example, texts of difficulty 3.1 may contain different sets of vocabulary and grammatical knowledge, and numeric scores are not able to capture this difference.

To improve on this, I will propose that the knowledge of a language is decomposable: vocabulary knowledge can be decomposed into a set of vocabulary words, and grammatical knowledge can also be decomposed into a set of *grammatical templates* (section 3.1). Based on knowledge decomposition, the difficulty of a text can be characterized as a set of its prerequisite conceptual units (vocabulary words or grammatical templates) and the relative difficulty of two texts can be judged by comparing their conceptual units. I will then introduce the *partial ordering graph* to organize the graph structure of a corpus that records all the information of relative difficulty between

materials and models the hierarchy for grammatical knowledge (section 3.2). To extend this model to vocabulary knowledge, I will relax the constraints that specify the relative difficulty and introduce the *fuzzy partial ordering graph* in order to improve the quality of the hierarchical knowledge structure (section 3.3). The (fuzzy) partial ordering graphs can be automatically created to organize a large corpus of authentic and up-to-date learning materials collected from the Internet without requiring any expert labor or student data.

Most of the work in this chapter was originally presented in *Grammatical Templates: Improving Text Difficulty Evaluation for Language Learners* (in collaboration with Erik Andersen, published at COLING'16) [97], *A Unified Framework for Knowledge Assessment and Progression Analysis and Design* (in collaboration with Fang He and Erik Andersen, published at CHI'17) [99], and *Adaptive Learning Material Recommendation in Online Language Education* (in collaboration with Hao Wu, Ji Hun Kim and Erik Andersen).

3.1 Conceptual units in grammar learning

Evaluating *text difficulty*, or *text readability*, is an important topic in natural language processing and applied linguistics [30,70,110]. A key challenge of text difficulty evaluation is that linguistic difficulty arises from both vocabulary and grammar [79]. However, most existing tools either do not sufficiently take the impact of grammatical difficulty into account [82,85], or use traditional syntactic features, which differ from what language students actually learn, to estimate grammatical complexity [29,36,81]. In fact, language courses introduce grammar constructs together with vocabulary, and grammar constructs vary in frequency and difficulty just like vocabulary [10,58,95]. Ideally, we would like to have better ways of estimating the grammatical complexity of a sentence.

To make progress in this direction, I will introduce *grammatical templates* as an important feature in text difficulty evaluation. These templates are what language teachers and linguists have identified as the most important units of grammatical understanding at different levels, and what students actually learn in language lessons. I will also demonstrate that grammatical templates can be automatically extracted from the dependency-based parse tree of a sentence.

To evaluate, I will compare the difficulty prediction accuracy of grammatical templates with existing readability features in Japanese language placement tests and textbooks. The results show that grammatical template features slightly outperform existing readability features. Moreover, adding grammatical template features into existing readability features significantly improves the accuracy by 7.4%. I will also propose a multi-level linear classification algorithm using only 5 grammatical features, and demonstrate that this simple and human-understandable algorithm effectively predicts the difficulty level of Japanese texts with 87.7% accuracy.

3.1.1 Grammatical template analysis

A key challenge in modeling text difficulty is to specify all prerequisite knowledge required for understanding a certain sentence. Traditional methods measure text difficulty mostly by evaluating the complexity of vocabulary (word count, word frequency, word type, etc.). This is effective for native speakers, who typically understand the grammar of their language but vary in mastery of vocabulary. However, these vocabulary-based methods underperform for language learners who have limited knowledge of grammar [14, 20].

To resolve this, we focus our research on grammatical difficulty. We introduce the

idea of *grammatical templates*, units of grammar that expert language instructors and linguists have identified as the most important grammatical knowledge, and are typically emphasized as key points in every textbook lesson [8, 68]. Since these grammatical templates are taught explicitly in language lessons and learned directly by language students, we believe they reflect the conceptual units of grammar more closely than parse trees.

Grammatical templates play an important role in language understanding because:

- Many grammatical templates suggest sentence structure. For example, “hardly ... when ...” in English, “nicht nur ..., sondern auch ...” (not only ... but also ...) in German, and “必ずしも ... とはいえない” (it is not necessarily the case that ...) in Japanese;
- For languages like Chinese and Japanese, lacking knowledge of some grammatical templates will cause difficulties in segmentation. For example, consider the Japanese template “...つ...つ” (two opposite behaviors occurring alternately) in the phrase “行きつ戻りつ” (to walk back and forth), and the Chinese template “越...越好” (the more ... the better) in “越早越好”(the earlier the better);
- Some grammatical templates may refer to special meanings that cannot be understood as the combination of individual words. For example, “in terms of”, “such that” in English, “mit etwas zu tun haben” (have something to do with ...) in German, and “... ことはない” (no need to ...) in Japanese.

We show some simple examples of grammatical templates for Japanese in Table 3.1¹. Line 2 shows the pronunciation of the templates, line 3 shows the translations, and

¹A long list of Japanese grammatical templates with English translations can be accessed at the JGram website: <http://www.jgram.org/pages/viewList.php>. There is also a nice and comprehensive book of Japanese grammatical templates, written by Japanese linguists, with English, Korean and Chinese translations: [90].

Template	-は	-の	-を	-ではない	-(名詞)に	-(動詞連用形)に
Pronunciation	- <i>wa</i>	- <i>no</i>	- <i>o</i>	- <i>dewa nai</i>	-(noun) <i>ni</i>	-(verb, i-form) <i>ni</i>
Translation	(topic)	(genitive)	(object)	is not	to (location)	for (purpose)
Notation	A	B	C	D	E	F

Table 3.1: Grammatical Templates in Japanese, with hyphens denoting words to be filled in. Note that some grammatical templates may impose requirements of some properties (e.g. part of speech or form) on the missing words.

Sentence	彼	は	すぐ	東京	に	到着する	
Pronunciation	kare	wa	sugu	<i>toukyou</i>	ni	touchakusuru	
Translation	he	(topic)	soon	<i>Tokyo</i>	to (location)	arrive	
Templates		A			E		
	“ he will soon arrive in Tokyo ”						
Sentence	僕	の	彼女	を	見	に	行く
Pronunciation	boku	no	kanojo	o	<i>mi</i>	ni	iku
Translation	I	(genitive)	girlfriend	(object)	<i>see</i>	for (purpose)	go
Templates		B		C		F	
	“ I go to see my girlfriend ”						
Sentence	これ	は	君	の	本	では	ない
Pronunciation	kore	wa	kimi	no	hon	dewa	nai
Translation	this	(topic)	you	(genitive)	book	is	not
Templates		A		B		D	
	“ this is not your book ”						

Table 3.2: Identified grammatical templates of Japanese sentences. In sentences, pronunciations and translations, grammatical templates are in bold. The word *toukyou* in the first sentence is a noun (Tokyo, 東京), as characterized by template E. The word *mi* (to see, 見) in the second sentence is the i-form (動詞連用形) of a verb, as required by template F.

the uppercase letters in line 4 are provided for notation. We also provide examples of how the grammar of a sentence can be described as combinations of these grammatical templates in Table 3.2.

Difficulty evaluation standard

To evaluate the difficulty of texts and grammatical templates, we follow the standard of the Japanese-Language Proficiency Test (JLPT). The JLPT is the most widely used test

for measuring proficiency of non-native speakers, with approximately 610,000 examinees in 62 countries and areas worldwide in 2011². It has five different levels, ranging from from N5 (beginner) to N1 (advanced). A summary of the levels can be found at JLPT website ³.

Grammatical template library

Due to their significance in Japanese education, grammatical templates are well-studied by Japanese teachers and researches. Grammatical templates are summarized and collected for both Japanese learners (common templates) and native speakers (templates used in very formal Japanese or old Japanese). We referenced 3 books about grammatical templates for Japanese learners [52, 80, 107], all of which divide their templates into N1-N5 levels, for generating our template library at each corresponding level.

Although not common, books may have different opinions on the difficulty of the same template. For example, an N1 template in book A may be recognized as an N2 template in book B. In order to conduct our experiments on a reliable template library, we only pick the templates recognized as the same level by at least two of the three books. For example, if both book A and C recognized template t as an N3 template, we can incorporate template t into our N3 template library. Ultimately, we collected 147 N1 templates, 122 N2 templates, 74 N3 templates, 95 N4 templates and 128 N5 templates in our library. All selected grammatical templates are stored in the format of regular expressions for easy matching in parse trees.

²<http://www.jlpt.jp/e/about/message.html>

³<http://www.jlpt.jp/e/about/levelsummary.html>

Grammatical template extraction

The framework of grammatical template extraction is shown in Algorithm 1. The program requires the dependency-based parse tree of a sentence as input, runs from bottom to top and returns a set of all identified grammatical templates $\mathbf{T}(node_0)$. Line 7 extracts the templates in the children of $node_0$ (and ignores the descendants of the children), by matching the phrase associated with the child nodes $[node_1, node_2, \dots]$ to all templates stored in terms of regular expressions in our library. The matching is based on both the structure of the phrases and the properties of the words. Line 8 shows $\mathbf{T}(node_0)$ covers all templates identified in subtrees rooted at $node_0$'s children and the templates extracted in the phrase associated with the child nodes $[node_1, node_2, \dots]$.

Require: A dependency-based parse tree of the sentence

Ensure: $\mathbf{T}(node_0)$ = set of identified grammatical templates in (sub)parse tree rooted at $node_0$.

- 1: **if** $node_0$ is leaf node **then**
- 2: return $\mathbf{T}(node_0) = \{\}$
- 3: **end if**
- 4: $node_1, node_2, \dots \leftarrow$ children of $node_0$
- 5: Calculate: $\mathbf{T}(node_1), \mathbf{T}(node_2), \dots$ // templates identified in subtrees rooted at $node_0$'s children
- 6: $\mathbf{T}_1(node_0) \leftarrow \mathbf{T}(node_1) \cup \mathbf{T}(node_2) \cup \dots$
- 7: $\mathbf{T}_2(node_0) \leftarrow$ identified templates in phrase $[node_1, node_2, \dots]$
- 8: return $\mathbf{T}(node_0) = \mathbf{T}_1(node_0) \cup \mathbf{T}_2(node_0)$

Algorithm 1: Grammatical Progression Extraction

We use Cabocha [49] for parsing Japanese sentences. This tool generates the hierarchical structure of the sentence as well as some properties (e.g. base form, pronunciation, part of speech, etc.) of each word. We execute Algorithm 1 on the parse tree to extract all identified templates of a Japanese sentence.

	N1 Texts	N2 Texts	N3 Texts	N4 Texts	N5 Texts
N1 Templates	0.902%	0.602%	0.077%	0.074%	0.056%
N2 Templates	2.077%	1.571%	1.072%	0.298%	0.056%
N3 Templates	4.070%	3.679%	1.531%	0.894%	0.222%
N4 Templates	16.635%	15.449%	13.323%	12.071%	1.832%
N5 Templates	76.316%	78.699%	83.997%	86.662%	97.834%

Table 3.3: Distribution of grammatical templates of level N1(hard)-N5(easy)

3.1.2 Statistics of grammatical templates

Corpus

We build our corpus from two sources: past JLPT exams and textbooks. The reading texts from JLPT exams are ideal for difficulty evaluation experiments since all of them are tagged authoritatively with difficulty levels, and JLPT problem sets before 2010 are publicly released⁴. We also collected reading texts from two popular series of Japanese textbooks: *Standard Japanese* [68] and *Genki* [8]. *Standard Japanese I* and *Genki I* are designed for the N5 level (the first semester) and *Standard Japanese II* and *Genki II* are designed for the N4 level (the second semester). Ultimately, our corpus consists of 220 texts (150 from past JLPT exams and 70 from textbooks), totaling 167,292 words after segmentation.

Results

For texts with different difficulties, we calculate the distribution of N1-N5 grammatical templates, which are shown in Table 3.3. We can see that N1 texts have higher portion of N1 and N2 templates than N2 texts, implying that the difficulty boosts from N2 to N1 are derived from increasing usage of advanced grammar. It is also clear that even in

⁴For example, the second exam in 2009 is published in [43].

	N1 Texts	N2 Texts	N3 Texts	N4 Texts	N5 Texts
N1 Templates	3.536	2.342	0.295	0.230	0.146
N2 Templates	8.141	6.110	4.130	0.922	0.146
N3 Templates	15.954	14.308	5.900	2.765	0.582
N4 Templates	65.214	60.081	51.327	37.327	4.803
N5 Templates	299.178	306.059	323.599	267.972	256.477

Table 3.4: Number of templates of level N1(hard)-N5(easy) per 100 sentences

the texts of advanced levels, the majority of the sentences are organized by elementary grammatical templates, and the advanced ones are only used occasionally for formality or preciseness.

We also calculate the per-100-sentence number of templates at each level, which are shown in Table 3.4. When comparing any two adjacent levels (e.g. N2 and N3), the templates at those levels or above seem to be the most significant. For instance, N1/N2 texts differ in numbers of N1 and N2 templates while they have similar numbers of N3-N5 templates, and the numbers of N1, N2 and N3 templates differentiate the N2/N3 texts while the numbers of N4 and N5 templates seem relatively similar. This phenomenon inspires us to build a simple and effective approach to differentiate the texts of two adjacent levels.

3.1.3 Difficulty level prediction

Multilevel linear classification

We differentiate two adjacent levels by looking at the knowledge ‘on the boundary’ and ‘outside the boundary’. Concretely, when judging whether a text is harder than level N_i , we consider a grammatical template as:

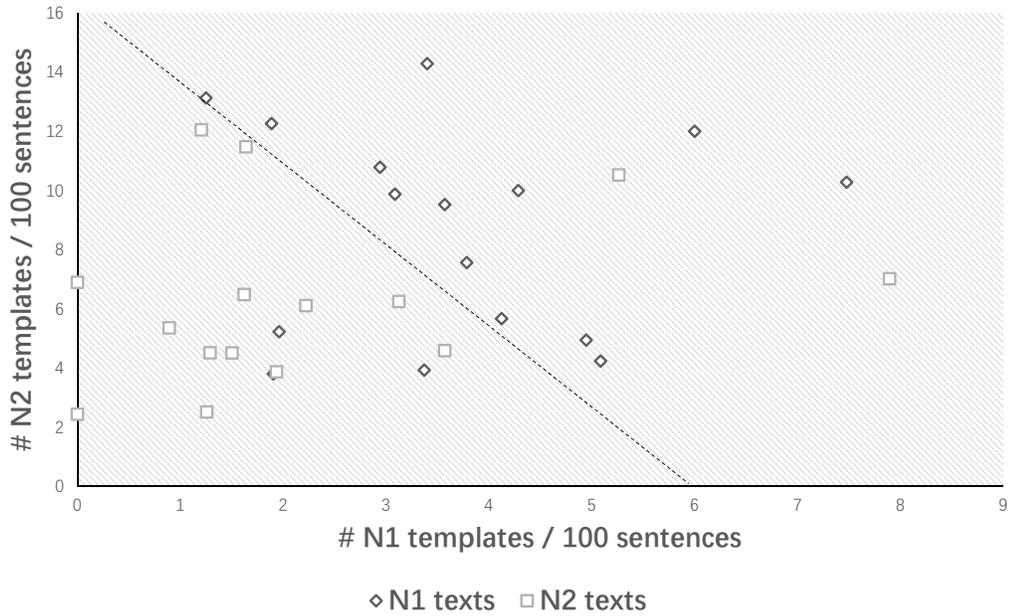


Figure 3.1: Grammatical difficulty in the N1/N2 texts

- *within the boundary*, if the template is easier than N_i (N_{i+1} to N_5);
- *on the boundary*, if the template is exactly at N_i level;
- *outside the boundary*, if the template is harder than N_i (N_1 to N_{i-1}).

We found that texts of adjacent levels are nearly linear-separable with two features: templates ‘on the boundary’ and templates ‘outside the boundary’. For example, Figure 3.1 shows how N1 and N2 texts are linearly separated based on the numbers of N1 and N2 templates: we can easily obtain a two-dimensional linear classifier separating N1 and N2 texts with 83.4% accuracy. This phenomenon is even more obvious at lower levels. Figure 3.2 shows N4 and N5 texts are almost perfectly linearly separated with two features: ‘number of N5 templates per 100 sentences’ (on the boundary) and ‘number of N1-N4 templates per 100 sentences’ (outside the boundary).

Taking advantage of this phenomenon, we build 4 linear classifiers for 4 pairs of adjacent levels. For example, the N4 classifier judges whether a text is harder than

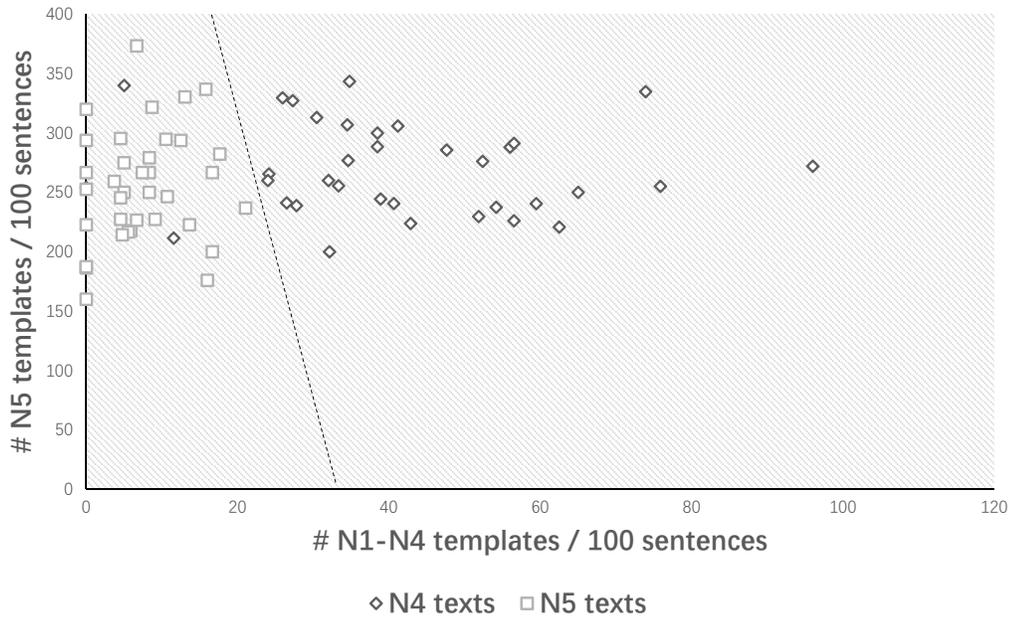


Figure 3.2: Grammatical difficulty in the N4/N5 texts

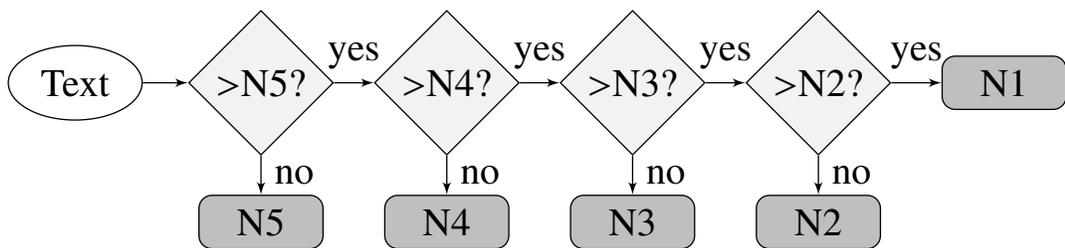


Figure 3.3: Multilevel Linear Classification (MLC). '>N5?' represents the linear classifier judging whether a text is harder than N5. The classifiers are similar for the other levels.

N4 (N1-N3). Our *Multilevel Linear Classification (MLC)* algorithm combines all 4 linear classifiers: A text is judged by the N5 classifier first. If it is no harder than N5, it will be labeled as an N5 text; otherwise, it will be passed to the N4 classifier in order to decide if it is harder than N4. The process continues similarly, until if it is judged to be harder than N2, it will be labeled as an N1 text. Figure 3.3 shows how the algorithm works.

Features

We conduct our experiments on the following 4 feature sets:

First, our *grammatical template feature set* has only 5 features:

- Average number of N1-N5 grammatical templates per sentence

We compare our work with recent readability evaluation studies [44, 70]. In our experiments, the *baseline readability feature set* consists of the following 12 features:

- Number of words in a text
- Number of sentences in a text
- Average number of words per sentence
- Average parse tree depths per sentence
- Average number of noun phrases per sentence
- Average number of verb phrases per sentence
- Average number of pronouns per sentence
- Average number of clauses per sentence
- Average cosine similarity between adjacent sentences
- Average word overlap between adjacent sentences
- Average word overlap over noun and pronoun only
- Article likelihood estimated by language model

Moreover, we combine these 12 traditional readability features with our 5 grammatical template features, forming a *'hybrid' feature set*, since we would like to see if grammatical template features are really able to improve text difficulty evaluation.

Since the text difficulty level prediction can be regarded as a special text classifica-

Feature Set (number of features)	Algorithm	Accuracy
TF-IDF Features (5100)	kNN	69.1%
	SVM	80.5%
Baseline Readability Features (12)	kNN	72.3%
	SVM	80.9%
Grammatical Template Features (5)	kNN	78.0%
	SVM	81.1%
	MLC	87.7%
Hybrid Features (17)	kNN	85.7%
	SVM	88.5%

Table 3.5: Accuracies of classifying N1-N5 texts

tion problem, we also extract *TF-IDF features* [61, 86] as an extra baseline, in order to see how general text classification techniques work on text difficulty evaluation.

Result

We test k-Nearest Neighbor and Support Vector Machines for each feature set. The implementations of these two popular classification algorithms are provided by the WEKA toolkit [33] and LibSVM [15]. The SVMs use RBF kernels [16]. We also test our Multilevel Linear Classification (MLC) algorithm on the grammatical template feature set. We use 5-fold cross validation to avoid overfitting. Table 3.5 shows the results.

Comparing the results of kNN and SVM across the four different feature sets in Table 3.5, it is clear that TF-IDF features have the largest feature set yet lowest accuracy, indicating the general word-based text classification techniques do not work well on text difficulty level prediction. Compared with baseline readability features, our grammatical template features have smaller number of features but higher accuracy (slightly higher with SVM but significantly higher with kNN). Moreover, the hybrid features, which combine baseline readability features with grammatical template features, deci-

sively outperform baseline readability features, confirming our expectation that adding grammatical template features to existing readability techniques improves text difficulty evaluation for language learners.

Additionally, our Multilevel Linear Classification algorithm achieves excellent accuracy with only 5 grammatical template features. An accuracy of 87.7%, although slightly lower than hybrid features + SVM (more features, more complexity), still significantly outperforms baseline readability techniques. In conclusion, the Multilevel Linear Classification algorithm has high accuracy, a small number of features, and a simple, human-understandable structure.

3.2 Hierarchical knowledge structure

Given that grammatical knowledge can be decomposed into a set of grammatical templates, this section will discuss how to build the hierarchical structure for grammar knowledge based on this decomposition.

3.2.1 Partial ordering graph

In order to select practice problems at an appropriate difficult level for each student, we need a hierarchical structure that encodes difficulty relationships between problems. One straightforward way to do this is to ask experts to specify these relationships. However, this becomes prohibitively difficult as the size of the problem set grows larger. Ideally, we would have automatic methods of problem organization.

In previous work [4, 13], researchers built partial ordering graphs for procedural

tasks. This work used procedural execution traces to organize content and partial orderings over those traces to create hierarchical content structures. For example, one can identify at least four basic skills that may be required to solve an integer addition problem: one-digit addition without a carry (A), one-digit addition with a carry (B), writing a carry (C), and bringing down a final carry (D). As an example, problems can be decomposed into basic skills as in Table 3.6:

Problem	2+3	15+18	93+15	298+865
Trace	A	ACB	AACD	ACBCBD

Table 3.6: Traces of Integer Addition Problems

However, in some domains, such as language learning, the target knowledge cannot easily be modeled as a single procedure. To induce hierarchical knowledge structures in such domains, we need to generalize beyond procedural domains. To do this, our framework takes advantage of *compositionality*, the idea that problems can be broken down into smaller conceptual units. This is well-studied in some semi-procedural domains such as math and language learning [4, 97].

For instance, a Japanese sentence can be decomposed into *grammatical templates*, units of grammar that expert language instructors and linguists have identified as the most important grammatical knowledge, and that students actually study in language lessons. These templates have proved to be beneficial for text difficulty evaluation for language learners (section 3.1, [97]). We found that the specific task of grammatically understanding a Japanese sentence can be decomposed into a (multi)set of grammatical templates. For example, Table 3.7 shows three Japanese sentences and their grammatical templates.

We can see that S1 has only one grammatical template: (– の). For a Japanese learner, S2 is harder than S1 since it has not only (– の), but also another template (–

S1:	私 の 先生 I of teacher “ my teacher ”
Templates:	(- の)
S2:	私 の 先生 は 忙しい I of teacher (topic) busy “ my teacher is busy ”
Templates:	(- の) (- は)
S3:	私 の 先生 の 名前 I of teacher of name “ my teacher’s name ”
Templates:	(- の) (- の)

Table 3.7: Japanese Sentences and Their Grammatical Templates

は). S3 repeats the same template (- の) twice hence is also harder than S1. These relationships cannot be captured by the partial ordering in [4] since it is unclear how to proceduralize the process of how a human understands these sentences. However, by considering multisets of concepts rather than execution traces, we can accommodate them as follows:

Definition 1 *A problem s can be decomposed into a series of concepts (basic skills) required by problem s . Since problems may require students to repeat some skills one or more times, we use a multiset of basic skills, indicated as $p(s)$, to characterize the difficulty of a problem.*

For example, for grammar learning, a text or sentence (problem s) can be decomposed into a series of grammatical templates (concepts).

Definition 2 We say problem s_1 is at least as hard as s_2 , indicated as $s_1 \geq s_2$, if and only if $p(s_1) \supseteq p(s_2)$. This implies that if a student can solve s_1 , he/she must be able to solve s_2 as well [4].

Here \supseteq denotes the superset relation between multisets. If $p(s_1) \supseteq p(s_2)$, then for any concept c that is required by problem s_2 n times, c must also be required by s_1 at least n times. For example, $AABC \supseteq ABC$ is true, while $AABCC \supseteq ABBC$ is not true since $ABBC$ has two “B”s while $AABCC$ has only one.

Definition 3 The strict partial order $s_1 > s_2$ is defined as $s_1 \geq s_2 \wedge s_2 \not\geq s_1$, which means s_1 is (strictly) harder than s_2 .

Definition 4 We say problem s_1 is directly harder than s_2 , if and only if $s_1 > s_2$ and there is no other problem s_3 such that $s_1 > s_3 > s_2$.

Using Definition 4, we build the hierarchical structure for a set of problems as follows:

Definition 5 We organize a set of problems $S = \{s_1, s_2, \dots\}$ (we call S the universal problem set) as a partial ordering graph $G = \langle S, E \rangle$, where

$$E = \{(s_i, s_j) | s_i, s_j \in S \wedge s_j \text{ is directly harder than } s_i\} \quad (3.1)$$

Namely, there is an (directed) edge from problem s_i to s_j in the partial ordering graph if and only if s_j is directly harder than s_i . An example of a partial ordering graph is shown in Figure 3.4.

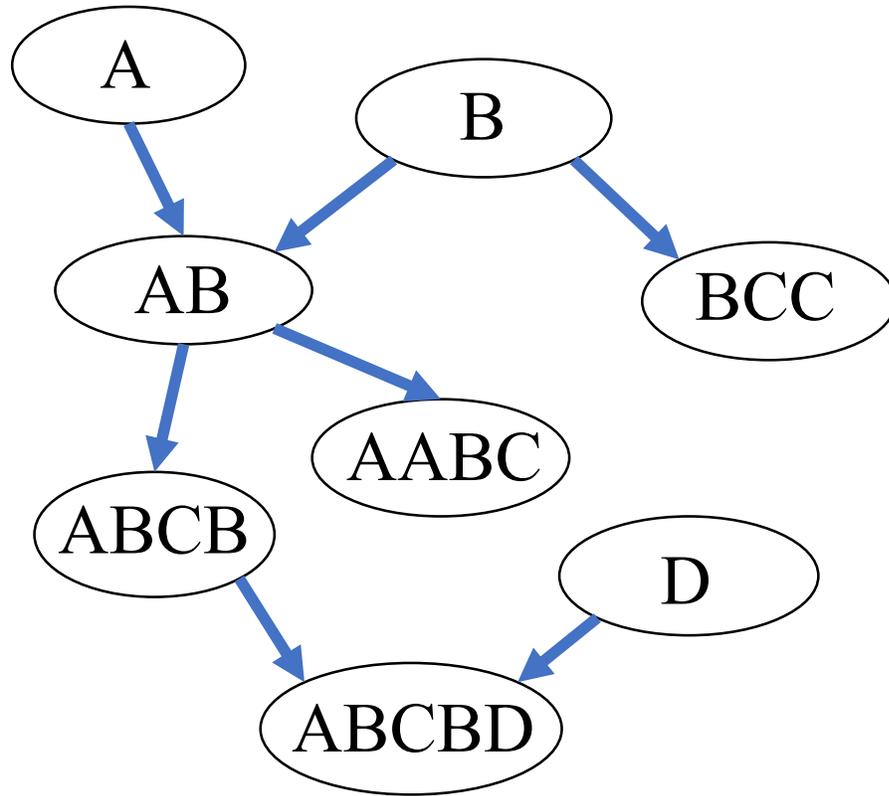


Figure 3.4: A sample partial ordering graph. Uppercase letters represent concepts and the nodes with strings represent problems. Each directed edge from problem s_i to s_j represents s_j is directly harder than s_i .

3.3 Scaling to vocabulary learning

The previous section presented a hierarchical knowledge structure for grammar knowledge using strict constraints to specify the relative difficulty between two texts (partial orderings between problems). However, this does not scale to teaching vocabulary with a large online corpus since these strict constraints yield too few edges in the structure. To this end, this section will investigate how to increase density without suffering an unacceptable loss of quality in prediction of relative difficulty. I will propose a *fuzzy partial ordering graph*, a refined hierarchical knowledge structure with relaxed constraints in order to organize a large corpus of authentic and up-to-date learning material collected

from the Internet.

3.3.1 Fuzzy partial orderings

The Internet provides a vast corpus of reading materials that are suitable for language learning. However, due to the large size and the freshness of this corpus, it is prohibitively expensive to ask experts to measure the difficulty of those materials, and data-driven techniques do not apply either due to the lack of student data. Therefore, in order to leverage learning materials from the Internet, recommender systems should be able to automatically measure the difficulty of those materials and build the hierarchical knowledge structure within the corpus. In section 3.2, I introduced a framework to do this for grammatical knowledge. In this section, I will first discuss an issue with this work that limits its application with regard to vocabulary, then address this issue and propose a refined hierarchical structure for modeling vocabulary knowledge.

In section 3.2, we used the partial ordering graph to model the relationship between reading materials and model the hierarchical structure of grammatical knowledge in a corpus [99]. Within this structure, the partial orderings are useful because they can help in the modeling of students' knowledge: a student understanding problem s implies that he/she can also understand problems easier than s (we will discuss this in details in Chapter 4). Also, this model takes advantage of *compositionality* of practice problems [97], and the order of concepts within a problem is unimportant. Therefore, it can be applied to both procedural and non-procedural educational tasks.

However, in order for the partial ordering graph to work, the hierarchical structure of domain knowledge must be “sufficiently dense”. Otherwise, the partial ordering graph will only have a small number of edges, and there will not be enough partial ordering

relations that can be used. Therefore, this model cannot be directly applied to vocabulary knowledge because vocabulary learning requires a large amount of conceptual units. For example, there are over 10,000 vocabulary words in Japanese learning whereas there are only around 500 grammatical concepts. A typical Japanese sentence may require 10-30 vocabulary words compared to only around 5 grammatical concepts. As a result, it is not common in an authentic corpus that a sentence covers all vocabulary knowledge of another sentence, and the vocabulary-based partial ordering graph will be too sparse.

To address this, we take advantage of the idea supported by existing work [48, 94] that language learners can infer the meanings of some unknown words if they understand the majority of the text, and they will accumulate language knowledge in this way. This idea inspired us to relax the partial ordering relations between two texts in order to increase the density in the vocabulary-based hierarchical knowledge structure.

Definition 6 *Problem s_1 is α -fuzzily harder than problem s_2 if s_1 covers at least a proportion α of required concepts of s_2 . Using this fuzzy partial ordering, we can also define the fuzzy partial ordering graph.*

We found that the hierarchical knowledge structure based on the fuzzy partial ordering in Definition 6 has 71% more edges than the strict version introduced in the former work [99], using fuzzy parameter $\alpha = 0.8$. As the fuzzy parameter α decreases, the number of edges in the fuzzy partial ordering graph increases exponentially (Figure 3.5). Although this relaxation increases density, it also lowers our confidence in the fuzzy partial ordering relations. If α is too small, there will be many edges in the fuzzy partial ordering graph, but our confidence in each edge (namely, the likelihood that a student understands a problem if he/she understands another problem that is fuzzily harder than it) will be too low.

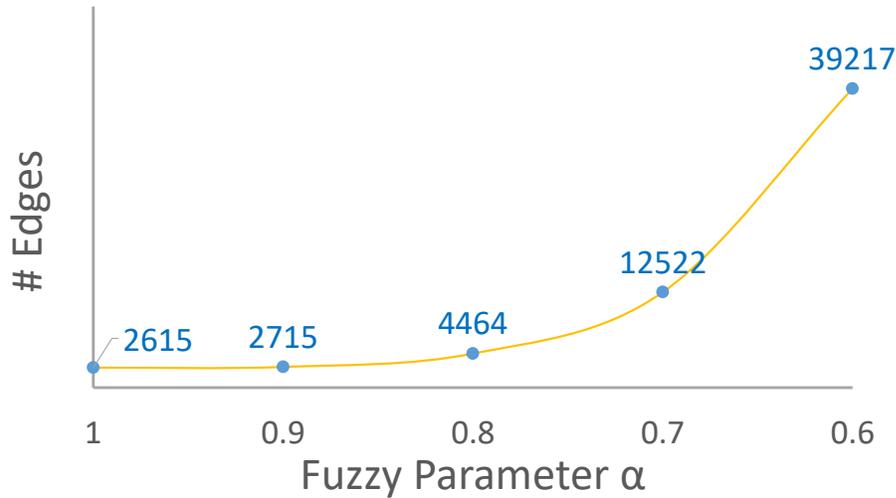


Figure 3.5: Number of edges in the fuzzy partial ordering graph over different fuzzy parameters, in our corpus of 4,269 texts. Decreasing the fuzzy parameter will increase the graph density exponentially.

This leads to a trade-off between the density of the hierarchical knowledge structure and our confidence in the (fuzzy) partial ordering relations. To identify the best fuzzy parameter for structuring vocabulary knowledge, we conducted a case study in our corpus of 4,269 Japanese texts. Examples of “fuzzily harder than” sentence pairs for fuzzy parameter $\alpha=0.9/0.8/0.7/0.6$ are listed in Table 3.8. We believe that the $\alpha = 0.9$ and $\alpha = 0.8$ values are suitable for use. In these two cases, the second sentence covers almost all the vocabulary knowledge in the first sentence. Therefore, students are very likely to understand the second sentence if they understand the first one. However, our confidence in the fuzzy partial ordering relations are too low for the $\alpha = 0.7$ and $\alpha = 0.6$ values, since in these two cases, the first sentence requires a certain amount of vocabulary knowledge that is not required by the second sentence. In this situation, we cannot be sure students who understand the second sentence will also understand the first one.

Based on these results, we used the fuzzy parameter $\alpha = 0.8$ in our vocabulary-based fuzzy partial ordering graph because the graph is sufficiently dense and our confidence in the partial ordering relations are high enough to use. However, the optimal fuzzy

α	Sample Sentence Pair
0.9	<p>席 の 数 より 客 の ほうか* 多 かった こと は、 5 回 あ り ま し た seat of number than passenger more 5 times there was</p> <p>“There were 5 times when there were more passengers than the number of seats.”</p> <p>先 月、 全 日 空 の 飛 行 機 か*、 席 の 数 より 客 が 1 人 多 い ま ま 出 発 し よ う last month ANA of flight seat of number than passenger 1 person more about to depart と し た こ と が あ り ま し た one time there was</p> <p>“Last month, there was a time when an ANA flight was about to depart but there was one more passenger than the number of seats.”</p>
0.8	<p>9 日、 この ボ ラ ン テ ィ ア に な り た い 人 た ち が 集 ま っ て、 太 田 市 で 勉 強 し ま し た 9th this volunteer become want people gather Ota(city) in studied</p> <p>“On the 9th, people who wanted to become volunteers gathered and studied in Ota.”</p> <p>集 ま っ た 人 た ち は、 あ と 2 回 勉 強 し て テ ス ト に 合 格 す る と、 病 院 な ど で 通 訳 を す る gathered people more 2 times study test pass if hospital like in interpret ボ ラ ン テ ィ ア に な り ま す volunteer become</p> <p>“The gathered people will become volunteer interpreters in places like hospitals, if they study two more times and pass the test.”</p>
0.7	<p>シ リ ア で は、 政 府 と 政 府 に 反 対 す る 人 た ち の 戦 争 が 続 い て い ま す Syria in government and government against people of war is ongoing</p> <p>“In Syria, the war between the government and the anti-government faction is still ongoing.”</p> <p>政 府 に 反 対 す る 人 た ち が た く さ ん い る ア レ ッ ポ と い う 町 に は、 政 府 の 軍 が government against people many there is Aleppo(city) in government of army 2 週 間 も 空 か ら 攻 撃 を 続 け て い ま す 2 weeks air from attack maintaining</p> <p>“In Aleppo, where there is a large anti-government faction, the government army maintained attacks from the air for two weeks.”</p>
0.6	<p>シ リ ア で は、 政 府 と 政 府 に 反 対 す る 人 た ち の 戦 争 が 続 い て い ま す Syria in government and government against people of war is ongoing</p> <p>“In Syria, the war between the government and the anti-government faction is still ongoing.”</p> <p>今 ま で の 10 年、 私 は 戦 争 が 続 い て い る 所 や 難 民 が 生 活 し て い る 所 へ now until of 10 years I war is ongoing place refugee is living place to 何 度 も 行 き ま し た for multiple times went</p> <p>“In the last 10 years, I have made multiple visits to places where a war was ongoing or refugees were living.”</p>

Table 3.8: Sample Sentence Pairs in the fuzzy partial orderings with the fuzzy parameter $\alpha = 0.9/0.8/0.7/0.6$. For each fuzzy parameter α , the second sentence is α -fuzzily harder than the first sentence. As the fuzzy parameter α decreases, our confidence in the fuzzy partial orderings (the likelihood for a student to understand the first sentence if he/she understands the second one) also drops. Text source: NHK Easy [63].

parameter α is likely different in each educational domain and needs to be empirically studied in each domain.

3.4 Conclusion

In this chapter, I discussed how to model and organize learning materials automatically. In section 3.1, I decomposed the grammatical knowledge within a text into grammatical templates, the conceptual units that identified by experts and taught to students in language lessons. I also proposed a new approach for evaluating text difficulty using these templates, which significantly improved the accuracy of text difficulty evaluation for Japanese language learning. Moreover, I introduced a simple, human-understandable, and effective text difficulty evaluation approach using only five grammatical template features. In section 3.2, I proposed the partial ordering graph to model the hierarchical knowledge structure within a corpus of learning materials based on automatic problem decomposition: a problem (or a learning material) can be characterized as a multiset of conceptual units. Within the partial ordering graph, each node represents a problem (a material) and each directed edge between two nodes represents one problem is directly harder than the other. In section 3.3, I relaxed the constraints of the “harder-than” relations such that we could have more edges in the fuzzy partial ordering graph. This increased the density of hierarchical knowledge structure and allowed for the modeling of vocabulary knowledge within a larger amount of authentic learning materials collected from the Internet.

Within the (fuzzy) partial ordering graph, we will be able comprehensively assess a student’s knowledge (section 4.1) and recommend reading texts to that student based on his/her ability (section 4.2). I will also explore the learning strategies of an expert-

created curriculum (section 5.1) and apply them into automatic curriculum design (section 5.2 and section 5.3).

In future work, we are interested in extending our work to other languages like English, and adapting grammatical templates for various languages. To achieve this, we need to itemize the grammar knowledge that students learn from language lessons. We can also develop a machine learning system that can automatically discover discriminative grammatical templates from texts. Moreover, we would like to study if the topic of a text has a considerable impact on text difficulty for language learners, just like vocabulary and grammar. Furthermore, we hope to extend our work to model and organize multimedia learning materials such that we can design language lessons with different types of materials including texts, audio, and videos.

CHAPTER 4

STUDENT ASSESSMENT AND ADAPTIVE RECOMMENDATION

Up until now, I have described how to build a partial ordering graph. In order to recommend practice problems to a student that are at the appropriate difficulty level, we need to accurately and comprehensively assess the student's knowledge. This is challenging because every student will understand a different subset of problems, and unidimensional assessment will not be comprehensive. For example, the set of mastered knowledge for different students at level N1 may be dramatically different, and a single numeric score such as "level N1" is not able to discriminate those students. Moreover, most existing content recommender systems for language learning are designed for formal learning scenarios such as universities and schools, and they make recommendations based on the student's standardized pre-assessment results. However, these systems cannot be scaled to informal learning scenarios such as online learning, where we usually do not have accurate and standardized information of a student's prior knowledge.

In this chapter, I will tackle these challenges in the perspective of students. I will primarily discuss two topics: how to comprehensively assess a student's knowledge (section 4.1) and how to recommend appropriate content for the student based on assessment results (section 4.2). Most of the work in this chapter was originally presented in *A Unified Framework for Knowledge Assessment and Progression Analysis and Design* (in collaboration with Fang He and Erik Andersen, published at CHI'17) [99] and *Adaptive Learning Material Recommendation in Online Language Education* (in collaboration with Hao Wu, Ji Hun Kim and Erik Andersen, to appear at AIED'19) [100].

4.1 Knowledge assessment

In this section, I will first leverage the partial ordering graphs and introduce the idea of the *knowledge boundary* to comprehensively characterize a student’s knowledge (section 4.1.1). I will then present an interactive approach to calculate a student’s knowledge boundary within the partial ordering graphs (section 4.1.2) and show how to use the knowledge boundary to predict a student’s future performance on a specific problem (section 4.1.3). Finally, I will evaluate these ideas in an online language assessment tool (section 4.1.4).

4.1.1 The knowledge boundary

We define the *knowledge boundary*, the “fringe” of knowledge that we use to characterize a student’s ability within the partial ordering graph¹.

Definition 7 *We measure a student’s ability with the knowledge boundary K , which is defined as the hardest problems that the student can solve. Formally, if T is the set of the problems that the student can solve, then:*

$$K = \{s | s \in T \wedge \nexists s' \in T \text{ such that } s' > s\} \quad (4.1)$$

Consider the partial ordering graph in Figure 4.1 as an example. Assume a student can solve problems A, AB, B, AABC and BCC (the green nodes in the graph), and cannot solve ABCB, D or ABCBD (the red nodes in the graph). Then the knowledge boundary consists of only two problems: AABC and BCC, since there are no other

¹Knowledge Space Theory defined the fringe of knowledge as a simpler representation of a student’s knowledge state. However, we define the knowledge boundary in a partial ordering graph, which provides a way to measure how far away a problem is from what a student currently knows.

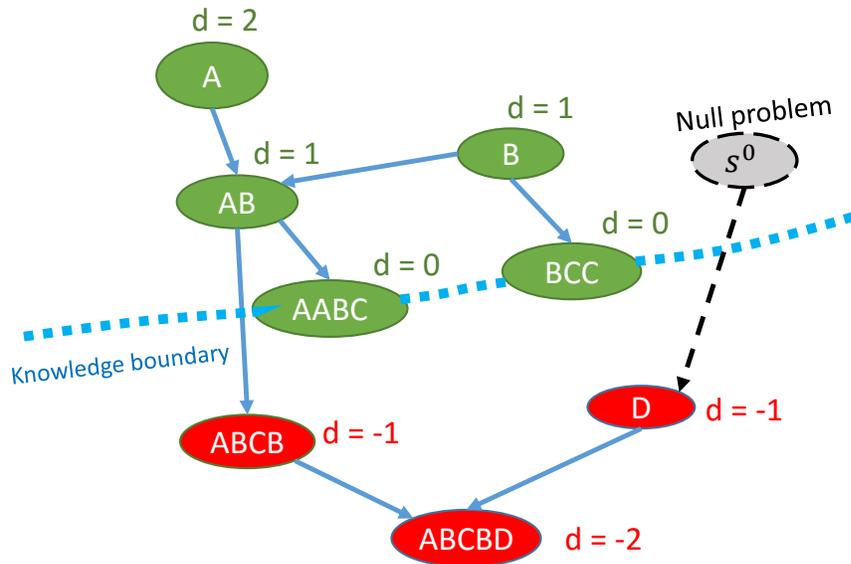


Figure 4.1: A Student’s Knowledge boundary in the partial ordering graph. Uppercase letters represent concepts and the nodes with strings represent problems. The student can solve the green problems but cannot solve the red ones. The distances from each problem to the knowledge boundary are indicated as d . The null problem s^0 is an auxiliary problem that is added to help us compute the distances.

“green” problems that are harder than AABC or BCC. The knowledge boundary does not include A, AB or B since there is a “green” problem AABC that is harder than all of them.

4.1.2 Calculating the knowledge boundary

We present a graph coloring algorithm of calculating a student’s knowledge boundary in the partial ordering graph. This algorithm is based on two properties of the partial ordering: if a student can solve problem s , he must be able to solve any problem s' that is at most as hard as s ($s \geq s'$); if a student cannot solve problem s , he must not be able to solve any problem s' that is at least as hard as s ($s' \geq s$). For example, if a student can

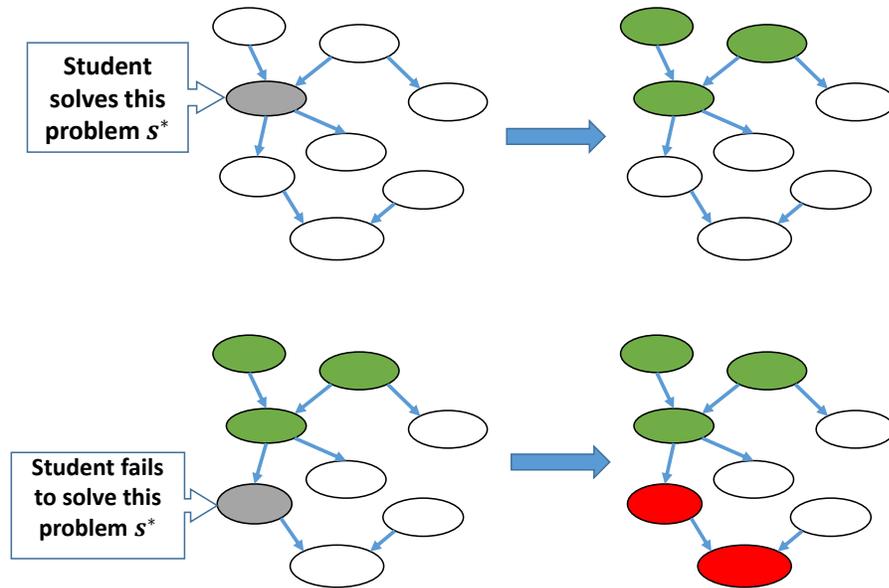


Figure 4.2: Coloring the partial ordering graph based on the student's response. If the student solves s^* , all the nodes that are at most as hard as s^* (including s^* itself) will be colored 'solvable' (green); if the student fails to solve s^* , all the nodes that are at least as hard as s^* (including s^* itself) will be colored 'unsolvable' (red).

solve problem AB, he/she can also solve problems A and B; if a student cannot solve problem ABCB, then he/she cannot solve problem ABCBD either.

At the start of the algorithm, all the problems (nodes) in the partial ordering graph are uncolored. The algorithm asks the student whether he/she can solve some problem s^* . If the student can solve s^* , all the nodes that are at most as hard as s^* (including s^* itself) will be colored 'solvable'; if the student cannot solve s^* , all the nodes that are at least as hard as s^* (including s^* itself) will be colored 'unsolvable'. Figure 4.2 shows how this coloring process works.

The algorithm repeatedly selects an uncolored problem s^* from the partial ordering graph, asks the student to solve it, and then updates the coloring of the graph based on the response. This is a greedy algorithm designed to minimize the number of problems that must be given to the student. Formally, if n_s^+ denotes the number of the uncolored

problems that are at most as hard as s , and n_s^- denotes the number of the uncolored problems that are at least as hard as s , then we can maximize the number of problems that can be colored based on the student’s response by greedily selecting a problem s^* as follows:

$$s^* = \arg \max_{s \text{ is uncolored}} \min(n_s^+, n_s^-) \quad (4.2)$$

4.1.3 Predicting performance on specific problems

Distance to knowledge boundary

In order to recommend problems at appropriate difficulty levels to the students, we need to predict their performance on problems. Existing IRT studies have proposed several popular models stipulating how student performance is related to student ability [56, 71]. However, they measure a student’s ability and the difficulty of a problem using unidimensional numeric scores, which is incomprehensive [28].

Ideally, prediction of a student’s performance would utilize multidimensional metrics to measure the distance between a problem and what a student already knows. The key technical challenge in the design of multidimensional metrics is that it is impossible to measure this distance without taking into account the hierarchical structure of the problem space. In our framework, we can measure this as the distance from the problem to the knowledge boundary in the partial ordering graph. We use signed numbers to distinguish which ‘side’ of the knowledge boundary a problem is on: problems ‘inside’ the boundary (which the student can solve) have positive distances while problems ‘outside’ the boundary (which the student cannot solve) have negative distances. Using this distance, we can leverage IRT models to predict a student’s performance on new problems.

Here, we give the definition of this distance together with the examples in Figure 4.1. Assume we have the universal problem set S , and a student can solve a subset of problems T . For example, in Figure 4.1, $S = \{A, AB, B, AABC, BCC, ABCB, D, ABCBD\}$, $T = \{A, AB, B, AABC, BCC\}$, and the knowledge boundary $K = \{AABC, BCC\}$. We calculate the distance from any problem $s \in S$ to the knowledge boundary K , indicated as $d(s, K)$, following the steps below:

Step 1: Calculate distances for problems on the boundary For any problem s such that $s \in K$, $d(s, K) = 0$.

For example, the distance of problems AABC and BCC is 0.

Step 2: Calculate distances for problems inside the boundary For any problem s such that $s \in T - K$, based on the definition of K , there must be one or more $s' \in K$ such that $s' \geq s$ (otherwise, s should be contained in K), and

$$d(s, K) = \min_{s' \in K} dis(s, s') \quad (4.3)$$

where $dis(s, s')$ indicates the length of the shortest directed path from s to s' in the partial ordering graph. If there is no directed path from s to s' , $dis(s, s') = \infty$. Note that if $s' \geq s$, then there must exist at least one directed path from s' to s .

For example, the distance of problem AB is 1 since problem AABC, which is on the knowledge boundary, is directly harder than AB. Similar for problem B (BCC is directly harder than B). The distance of problem A is 2 since the shortest directed path from A to any problem on the knowledge boundary (which is $A \rightarrow AB \rightarrow AABC$) has length 2.

Step 3: Calculate distances for problems outside the boundary In a hierarchical knowledge structure, it follows intuitively that problems that are further away from the

boundary will be more difficult for the student. Therefore, for any problem s such that $s \in S - T$, we define the distance $d(s, K)$ to be the shortest directed path from any problem in T to s . Note that this distance also has a teaching interpretation: if easier problems should always be taught before harder problems [76], then this distance also measures the number of problems that need to be taught before teaching s .

Since there are some basic problems that have no problems easier than them, and thus have no incoming edges in the partial ordering graph, there is not always a path from a problem in T to s . For instance, there is no directed path from any problem in T to problem D in Figure 4.1. To resolve this, we add a *null problem* s^0 , the pseudo problem with no prerequisite concepts, to T . For any problem $s \in S - T$, if there is no other problem s' such that s is directly harder than s' (namely, s has no incoming edges), we add an edge (s^0, s) to the partial ordering graph.

Now there is at least one directed path from the null problem s^0 or some problem in T to s . We can define the distance as:

$$d(s, K) = - \min_{s' \in T \cup \{s^0\}} dis(s', s) \quad (4.4)$$

Note that $d(s, K)$ is negative if and only if s is outside the knowledge boundary.

For example, the distance of problem ABCB is -1 since it is directly harder than the “green” problem AB. Problem D has no incoming edges in the partial ordering graph, hence we add an edge from the null problem s^0 to D, and the distance of problem D is -1 since the path $s^0 \rightarrow D$ has length 1. Lastly, the distance of problem ABCBD is -2. Actually, there are two shortest paths with length 2: $AB \rightarrow ABCB \rightarrow ABCBD$ and $s^0 \rightarrow D \rightarrow ABCBD$.

The metric of distance is dependent on the density of the partial-ordering graph. This is inevitable since the measurement is based on the hierarchical structure of the problem

space. We believe that for most well-defined problem spaces, it is a reasonable assumption that the partial ordering graph will be sufficiently dense. We will demonstrate this metric works well for a well-built Japanese language learning corpus in the next section.

In the later part of this chapter, we will use d to denote $d(s, K)$ for convenience.

Adapted Rasch model

In this section, we describe how we can adapt existing unidimensional IRT models to build a multidimensional metric that leverages the partial ordering graph and the distance d calculated in the previous section in order to predict student performance. One of the most famous models of IRT, the Rasch model [71], stipulates that a student's performance P is a function of the difference between the student's ability θ and the problem's difficulty b :

$$P(\theta, b) = \frac{e^{\theta-b}}{1 + e^{\theta-b}} \quad (4.5)$$

In the Rasch model, $\theta - b$ measures the difference between the student's ability and the difficulty of the problem. In our framework, we use d to measure this, hence we want to replace $\theta - b$ with d . It is also common to add a *discrimination* parameter a , which represents the degree to which the task discriminates between students [21]. In addition, we have found in practice that we need to add an additional parameter c , which measures how 'comfortable' the students feel with the problems on the knowledge boundary (any problem s such that $s \in K$). Ideally, students should be able to solve the problems on the boundary. However, in reality, students may still experience some difficulty with these problems. We can thus replace $\theta - b$ with $ad + c$.

This brings us to an adapted Rasch model, which stipulates how student performance

P is related to the distance d :

$$P(d) = \frac{e^{ad+c}}{1 + e^{ad+c}} \quad (4.6)$$

In the next section, we will demonstrate that this adapted model nicely fits the data collected from our knowledge assessment platform.

4.1.4 Evaluation

In this section, we will evaluate our calculation of the knowledge boundary and the distance presented in the previous section by applying it to a Japanese language learning domain.

J100: A language assessment platform

J100 is a language assessment platform that evaluates how well a user understands the grammatical knowledge in *Genki I* [8] (JLPT level N5²). It is designed for Japanese beginners who have learned Japanese for less than 1 year. As the platform starts, users view 15-18 Japanese sentences as well as corresponding vocabulary explanations and sentence translations³. The J100 platform will ask users to judge how well they understand each sentence. All of the test sentences were collected from *Genki I*. Figure 4.3 shows a screenshot of the J100 platform.

J100 has 2 stages: *assessment* and *evaluation*. In the assessment stage, users view 10 sentences and respond whether they understand each sentence ('Yes' or 'No'). Using

²Japanese Language Proficiency Test (JLPT) has 5 levels: from N1 (advanced) to N5 (beginner). The JLPT levels are summarized here: <http://www.jlpt.jp/e/about/levelsummary.html>

³The vocabulary explanations are provided to users since J100 focuses on grammatical knowledge only, while users are not recommended to read sentence translations unless they are not sure whether their understanding of the sentence is correct.



Vocabulary

撮る(とる)	to take (pictures)
写真(しゃしん)	photograph, picture
お寺(おてら)	temple

Translation

I took many pictures at the temple.

Sentences are selected from: Banno, E., Ikeda, Y., Ohno, Y., and Shinagawa, C. (2011). Genki I: An Integrated Course in Elementary Japanese. Tokyo: The Japan Times.

Figure 4.3: Screenshot of J100 Platform (Assessment Stage)

the graph-coloring algorithm, we calculate a user's knowledge boundary based on his or her responses in this stage. In the evaluation stage, users view 5-8 additional sentences and are asked to indicate how well they understand each sentence ('Yes', 'Almost', 'Somewhat', 'Little', or 'No'). Sentences used in the assessment stage are not repeated in the evaluation stage. The responses in the evaluation stage are used to validate the knowledge boundary calculated in the assessment stage.

Reddit deployment

We recruited users through the Japanese Learning Sub-reddit⁴. The deployment was very successful: 847 users finished the J100 “test”, and our post received an up-vote/down-vote score of 145 (for comparison, the other posts on the same page have an average score of 12). We received about 50 comments from J100 users, and most of the comments are supportive “*This is ideal. Especially good for learners like myself who tend to forget bits and details of older lessons, and just good in general for testing your progress.*” Many users even expressed a future interest in our platform and requested that we make it work for higher levels (N4-N1): “*My friends love this. If you could make some intuitive tests for N4-N1 I’d pay money if I could take tests that could also adapt to my level. Making me learn little by little what I need improving on.*”

Validating the knowledge boundary and distance metric

In order to validate our calculation of the knowledge boundary and the distance metric, we will demonstrate that the user data collected from J100 nicely fits the adapted Rasch Model. We calculate a user’s knowledge boundary K based on his/her responses to the problems in the assessment stage, and for each problem s responded in the evaluation stage, we measure the distance from the problem s to the knowledge boundary K . The user responses in the evaluation stage can be regarded as users’ self-estimation of their performance, and we score the five possible responses uniformly from 1 to 0:

Response	Yes	Almost	Somewhat	Little	No
Score	1	0.75	0.5	0.25	0

Table 4.1: Scoring User Responses in the Assessment Stage of J100

⁴www.reddit.com/r/LearnJapanese

Distance	User Response	
	Average	StdDev.
-4	0.125	0.11547
-3	0.279412	0.329654
-2	0.638037	0.40631
-1	0.861589	0.266999
0	0.913887	0.198634
1	0.940882	0.171181
2	0.959838	0.144577
3	0.983871	0.11543

Table 4.2: The average and standard deviation of J100 user responses, grouped by the calculated distance from the knowledge boundary. Note that the standard deviation of the user responses is maximized where the distance is at some point between -2 and -3, indicating that users are most unsure of their ability at this distance.

We would like to test if the relationship between the distance and user response matches the adapted Rasch model (Equation 4.6). If it does match, then this result will verify our calculation of the knowledge boundary as an effective measurement of a student’s knowledge. Additionally, we can utilize the distance from a problem to the student’s knowledge boundary to predict the student’s performance on that problem.

We calculate each user’s knowledge boundary after the assessment stage, and calculate the distance to each problem given in the evaluation stage to that knowledge boundary. Then we group these problems by these distances and calculate the average and standard deviation of the user responses when given problems with the same distance (Table 4.2).

We use non-linear regression to estimate the parameters a and c , and show the results in Figure 4.4. The results suggest that the adapted Rasch model fits our user data fairly well ($R^2 = 0.992$). There are some imperfections in the top-right corner of the figure, where users tended to indicate less confidence in their understanding of the sentence than would have been expected. This may be because users find it hard to judge whether

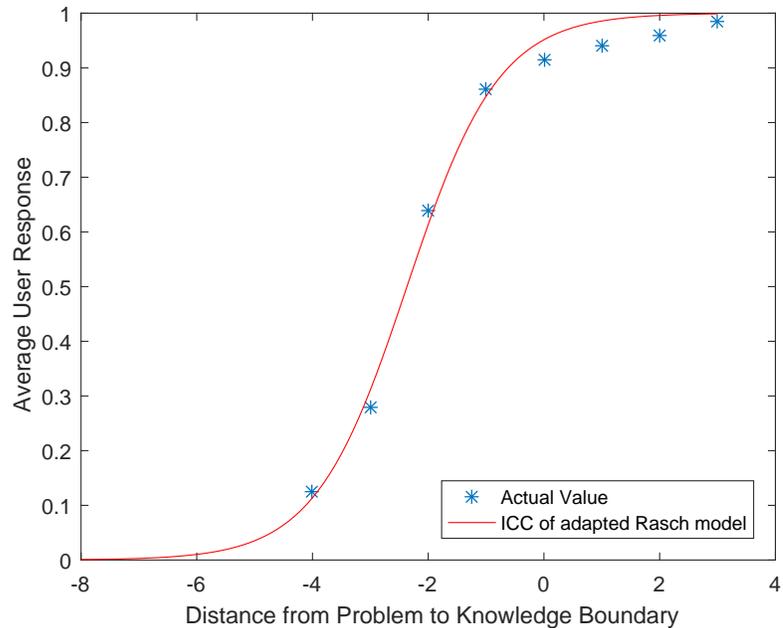


Figure 4.4: The adapted Rasch model nicely fits the J100 user data. The blue data points show the average user response to problems with the same distance from the knowledge boundary, and the red curve is the *Item Characteristic Curve* (ICC) calculated based on the adapted Rasch model.

they ‘understand’ or ‘almost understand’ a sentence, and may have slightly underestimated their ability. It could also result from the gap between the scores of ‘Yes’ (1) and ‘Almost’ (0.75) - if a user wanted to indicate something in between, such as 0.9, no such response was available.

There is another interesting phenomenon: the standard deviation of the user responses (in Table 4.2) is maximized where the distance is at some point between -2 and -3, and the slope of the curve in Figure 4.4 is also maximized where the distance is at some similar point between -2 and -3. According to IRT, the problems that are at the difficulty level that corresponds to the maximum slope of the curve are the most discriminative, and the J100 data shows that users are most unsure of their ability at this distance. This makes sense and is further evidence that the adapted Rasch model fits the

J100 user data well.

In conclusion, these results validate that our framework can efficiently measure students' ability with the knowledge boundary and predict student performance on specific problems. Since our formalism is inspired by both Item Response Theory (IRT) and Knowledge Space Theory (KST), the idea of the knowledge boundary and the distance may contribute to the unification of these two theories.

4.2 Incorporating assessment into recommendation

In order for students to be engaged, they need to experience learning materials at the right difficulty level. Although we have seen existing educational recommender systems that recommend learning materials based on student ability, most of these systems characterize each student by standardized pre-assessment results, such as in standard language placement tests [17, 69]. However, in online learning, where pre-assessment results are usually unavailable, we still lack an effective approach to recommend learning materials that automatically assesses and adapts to each student's prior knowledge. Ideally, recommender systems need to carefully balance the trade-off between assessment and recommendation: in order for recommendations to be appropriate, the system needs to accurately assess each student; however, excessive assessment can potentially harm engagement because students might need to respond to too many problems that are far outside of their comfort zone.

To this end, this work seeks to incorporate an appropriate amount of knowledge assessment into learning material recommendation. In this section, I will first present an adaptive heuristic to assess a student's prior knowledge faster (section 4.2.1), as well as a recommendation heuristic that selects learning materials that are just beyond

his/her knowledge within the (fuzzy) partial ordering graph (section 4.2.2). I will then propose a hybrid approach that uses a probabilistic function to balance the assessment and recommendation heuristics to improve student engagement in online learning (section 4.2.3). Finally, I will evaluate this approach in an online Japanese text recommendation tool (section 4.2.4).

4.2.1 Adaptive assessment heuristic

To recommend learning materials that adapt to each student’s prior knowledge, we follow a typical interaction process in adaptive education systems [99, 109]: the system keeps selecting the next problem (learning material) to present to a student and updating the model of the student’s knowledge based on his/her response. In the last section, we introduced a framework for modeling a student’s knowledge in the hierarchical knowledge structure. This framework characterizes a student’s knowledge by monitoring whether he/she can solve each problem in the library. With the help of partial orderings between problems, the assessment algorithm can infer the student’s performance on some problems without presenting them. To be more specific, if the student can solve problem s_1 , he/she can also solve problems that are easier than s_1 ; if the student cannot solve problem s_2 , he/she cannot solve problems that are harder than s_2 either.

Building on this framework, we propose an adaptive assessment heuristic to select the next problem in the (fuzzy) partial ordering graph.

The (Adaptive) Assessment Heuristic: Select the problem that maximizes the *expected* amount of information gained on the student’s prior knowledge. Formally, the assessment heuristic selects the problem s^* such that:

$$s^* = \arg \max_s [p_s n_s^+ + (1 - p_s) n_s^-] \quad (4.7)$$

where p_s indicates the probability that the student can solve s . If the student can solve s , n_s^+ represents how many problems we know that he/she can solve. Otherwise, if the student cannot solve s , n_s^- represents how many problems we know that he/she cannot solve. Both n_s^+ and n_s^- include s itself and exclude the problems we already know the student can/cannot solve before presenting s .

The probability p_s can be estimated in a straightforward way:

$$p_s = N^+ / (N^+ + N^-) \quad (4.8)$$

where N^+ and N^- denote the number of presented problems that the student can and cannot solve.

Note that the assessment heuristic in Equation (4.7) is different from the heuristic in Equation (4.2) in the last section. The new heuristic in Equation (4.7) incorporates the probability p_s and calculates the *expected* amount of information gained on the student's prior knowledge, while the previous heuristic only focuses on the information gained in the lesser of the two cases where the student can/cannot solve the problem. By doing this, the new heuristic adapts to students at the extremes of ability levels much faster. For instance, for a very good student that can solve 9 out of 10 problems presented to him/her, the new assessment heuristic will start to select the hardest problems in our library from the fifth problem, while the previous heuristic will always select the problems with intermediate difficulty.

4.2.2 ZPD-based recommendation heuristic

Vygotsky's Zone of Proximal Development (ZPD) stipulates that a student can solve the problems just beyond his/her knowledge with guidance, and a good teacher/tutor system should recommend those problems to the student. Based on this theory, we

propose the recommendation heuristic to select the next problem in the (fuzzy) partial ordering graph.

The (ZPD-based) Recommendation Heuristic: Select the problem that is directly harder than some problem that the student can solve. Since we believe that students are more engaged while solving a problem relevant to their experience, if there are multiple problems satisfying this requirement, pick the one that is *most relevant* to the student prior knowledge.

Here the relevance of a problem to the student’s prior knowledge can be measured by counting the “harder than” relations between that problem to any problem that the student can solve within the hierarchical knowledge structure. Practically, the relevance is measured as the number of edges from that problem’s node to any solvable problem’s node in the (fuzzy) partial ordering graph.

4.2.3 Balancing assessment and recommendation

Both assessment and recommendation heuristics are for selecting the next problem to present to students. The difference between them is that the assessment heuristic searches the whole knowledge structure to extract more information about a student’s knowledge, while the recommendation heuristic only selects the problems that are just outside the “boundary” of the set of problems that the student has correctly answered.

Our system uses a probabilistic function to balance the assessment and recommendation heuristics. To select the next problem, our system chooses the assessment heuristic with probability

$$p = \#Prob/M \tag{4.9}$$

and chooses the recommendation heuristic with probability $1 - p$. Here $\#Prob$ represents the number of the problems that the student has experienced, regardless of whether he/she has solved those problems. M is a pre-set parameter that controls how fast our system transitions from assessment-favoring to recommendation-favoring. It also indicates that our system will always choose the recommendation heuristic after the student has experienced M problems.

This function ensures that our system favors the assessment heuristic at the beginning in order to gain more information about a student's knowledge. As the student experiences more problems, and the model of student's knowledge gets more comprehensive and convincing, our system tends to make more recommendations in the student's ZPD.

4.2.4 Evaluation

We evaluate our adaptive learning material recommender system in *JRec*, a Japanese reading text recommendation tool. Our corpus of 380 articles was collected from NHK Easy [63], a Japanese news website for language learners. In order to accommodate beginners, our tool split those articles into 4,267 sentences and paragraphs such that students do not have to read the whole article. Afterwards, it analyzed the hierarchical structure of vocabulary knowledge in the corpus and built a fuzzy partial ordering graph. When using this tool, users are directed to an NHK Easy webpage, read a recommended text (a paragraph or a sentence), and respond whether or not they understand it. Our tool highlights the recommended text and grays out the rest of the webpage. Figure 4.5 shows a screenshot of JRec. We released our tool in the Japanese Learning Sub-reddit [75] and recruited 368 users in three days.

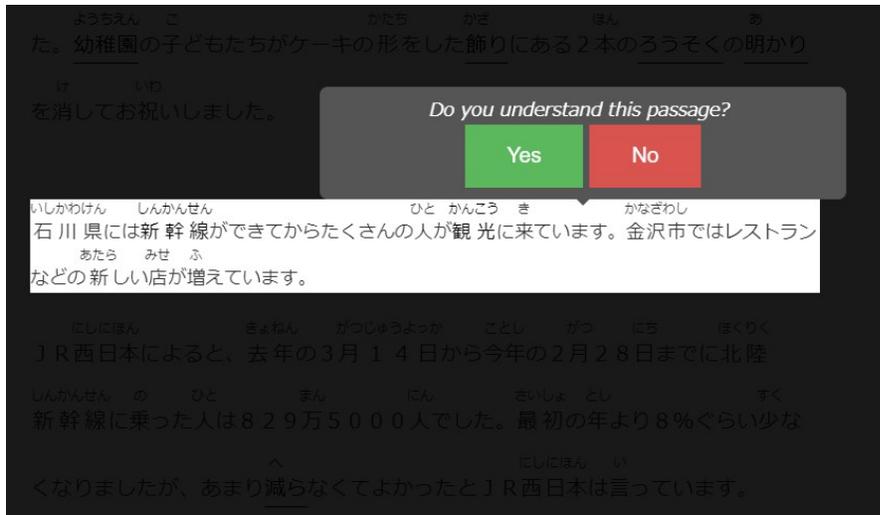


Figure 4.5: Screenshot of *JRec*, a Japanese reading text recommendation tool. When using this tool, users are directed to an NHK Easy webpage [63], read a recommended text, and respond whether or not they understand it. Our tool highlights the recommended text and grays out the rest of the webpage.

Adding adaptivity improved engagement significantly

In *JRec*, we tested four different versions: 1) adaptive recommendation (which balances recommendation and assessment as we discussed in the last section⁵) and 2) non-adaptive recommendation (with no assessment incorporated), as well as 3) assessment-only, and 4) random selection as additional baselines. We particularly wanted to see if adaptive recommendation is more engaging than non-adaptive recommendation, since this would demonstrate that incorporating adaptive assessment can enhance learning material recommendation.

In order to measure engagement, we recorded the number of texts each user read before leaving. 131 randomly selected users used adaptive recommendation (A.R.), 91 users used non-adaptive recommendation (N.R.), 115 users used assessment-only (A.O.)

⁵We used $M = 50$ in Equation (4.9) to balance assessment and recommendation.

Comparison	Results
Adaptive Recommendation. vs Non-adaptive Recommendation.	$p = .035, Z = 2.109$
Assessment-Only vs Adaptive Recommendation	$p = .766, Z = 0.298$
Assessment-Only vs Non-adaptive Recommendation	$p = .022, Z = 2.287$
Random vs Non-adaptive Recommendation	$p = .547, Z = 0.603$
Assessment-Only vs Random	$p = .294, Z = 1.049$
Adaptive Recommendation vs Random	$p = .389, Z = 0.861$

Table 4.3: We ran Wilcoxon Rank-sum tests for all pairs of our four groups: Adaptive Recommendation (A.R.), Non-adaptive Recommendation (N.R.), Assessment-Only (A.O.) and Random (Rand.). The difference between adaptive recommendation and non-adaptive recommendation was statistically significant ($p = .035$).

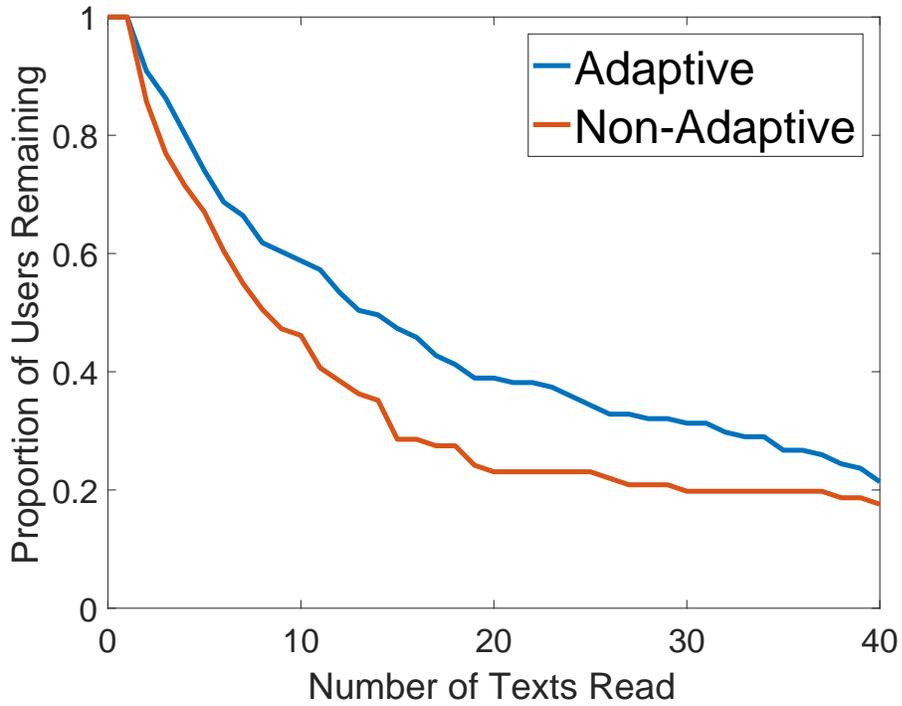


Figure 4.6: Proportion of users remaining after reading certain amount of texts. We observed that users in the adaptive recommendation group read 62.5% more texts than those in the non-adaptive recommendation group, which indicates that incorporating adaptive assessment significantly improved student engagement in learning material recommendation.

and 31 users used the random algorithm (Rand.).⁶ Since our data was not normally distributed, we ran Wilcoxon Rank-sum tests for all pairs of the four groups (Table 4.3). We observed that the median user in the adaptive recommendation group ($Median = 13$) read 62.5% more text than those in the non-adaptive recommendation group ($Median = 8$), and the difference between these two groups was statistically significant ($p = .035$), which indicates that adaptive recommendation led users to read more texts than non-adaptive recommendation. Figure 4.6 shows the proportions of users remaining after reading certain amounts of texts in the adaptive recommendation and the non-adaptive recommendation group. These results demonstrate that incorporating adaptive assessment can significantly enhance learning material recommendation in online learning. In addition, the median user in the assessment-only group read 12 texts, which was also significantly more than that in the non-adaptive recommendation group ($p = .022$). The median user in the random group read 8 texts and we did not find a statistically significant difference compared to the other three groups, possibly because the random group had too few users.

4.3 Conclusion

In this chapter, I discussed how to multidimensionally assess a student's prior knowledge and how to incorporate assessment into recommendation in order to make accurate recommendations and increase student engagement in online language learning. In section 4.1, I introduced the *knowledge boundary*, the set of the hardest problems that a student can solve, to model the student's knowledge multidimensionally within the hierarchical knowledge structure (namely, partial ordering graphs). The knowledge bound-

⁶Users were assigned to these four conditions at a ratio of 3:3:3:1, respectively. Since the tool only recorded when a user responded to a text, the number of recorded users in each group differs somewhat from the expected ratio. This may be because some users quit before responding to the first problem.

ary can be automatically calculated through an interactive algorithm that communicates with students, and can be used to predict a student's future performance on a specific problem. These ideas were evaluated in J100, an online Japanese knowledge assessment tool, and the results verified that this interactive algorithm can effectively assess a student's prior knowledge within the partial ordering graph and the prediction of student performance is accurate according to the Rasch model. In section 4.2, I revised the assessment algorithm to make it more efficient and incorporated it into a ZPD-based recommendation heuristic using a probabilistic function to balance assessment and recommendation. Based on the user evaluation of JRec, an online Japanese text recommendation tool, the adaptive recommendation (with assessment incorporated) achieved significantly better student compared to the non-adaptive recommendation (without assessment incorporated).

In future work, we would like to explore ways to assess a wider range of language skills, not only grammatical and vocabulary knowledge, but the listening, reading or speaking skills as well. This will require a hierarchical structuring for multimedia corpora. We are also interested in how the balance of assessment and recommendation affects the student's learning effectiveness and engagement. Furthermore, we hope to apply the assessment and recommendation approaches mentioned in this chapter not only to computer-assisted language learning, but also to other educational domains such as learning programming languages, mathematics, or even general knowledge.

CHAPTER 5

LEARNING PROGRESSION ANALYSIS AND DESIGN

Another key challenge in Computer-Assisted Language Learning is to automatically design curricula. In this work, a curriculum is specified as a *learning progression*, an optimal sequence of tasks or learning materials that help people practice and master new skills smoothly while attracting them to learn longer. Generally, tasks in a good progression should have gradually increasing difficulty, such that students can practice easier tasks before harder tasks [76]. Since hand-crafting such progressions can be time-consuming [2, 3], and may not be scaled to different user-specific progression parameters, researchers have recently studied several principles and approaches to synthesize progressions automatically: Andersen et al. synthesized level progressions for an algebra learning game based on the analysis and generation of the *execution traces* of solving algebra equations [4]. Butler et al. developed a feature-based system to design levels for a fraction game by assigning different weights to the skills or skill combinations in *solution procedures* and minimizing the total cost [12, 13]. However, we still lack practical approaches to characterize existing handcrafted language learning curricula and apply the wisdom of experts to synthesize progressions for language learning.

In this chapter, I will try to bridge this gap by incorporating the general principles identified in expert-designed progressions into automatic progression design in order to improve online language learning games and tools. In section 5.1, I will first explore general characteristics of good learning progressions by analyzing expert-designed learning progressions from language learning textbooks and online tools, and define a parameter space to characterize and compare different progressions in the partial ordering graphs. I will then present a simple greedy algorithm in section 5.2 to demonstrate that progressions can be automatically synthesized according to those defined param-

ters. In section 5.3, I will focus on an important pattern of expert progressions: *goal-drivenness*, which stipulates that a good learning progression should build up to some hard tasks as soon as possible to help students feel a sense of accomplishment. Based on this idea, I will propose a goal-based progression synthesis algorithm that is capable of creating synthesized progressions comparable to expert-created progressions in terms of both learning effectiveness and engagement, when applied to a Korean learning game.

Most of the work in this chapter was originally presented in *A Unified Framework for Knowledge Assessment and Progression Analysis and Design* (in collaboration with Fang He and Erik Andersen, published at CHI'17) [99] and *Goal-based Progression Synthesis in a Korean Learning Game* (in collaboration with Brandon Cohen, Sixian Yi, Jung Yun Park, Nicholas Teo and Erik Andersen, to appear at FDG'19) [98].

5.1 Progression analysis of expert-designed curricula

Good progressions enhance students' engagement of the curriculum [51, 77], and our work aims to discover general characteristics of good progressions. In this section, we will leverage our framework to analyze educational progressions, and introduce two significant features of a good progression: composition and pace. We study existing curricula and demonstrate some striking similarities in expert-designed progressions.

5.1.1 Composition: the balance of learning and review

When designing an educational progression, a key consideration is how much time one should spend reinforcing previously-introduced knowledge as opposed to introducing new knowledge. Another consideration is whether the progression should grow in com-

plexity by combining concepts together in new ways. Ideally, we would be able to define a progression parameter space that can concisely capture these important aspects.

In our framework, for any problem s that the student is about to learn, we classify s , with respect to the current knowledge boundary K , as:

reinforcement, if $\exists s' \in K$ s.t. $s' \geq s$;

recombination, if $\nexists s' \in K$ s.t. $s' \geq s$, and for any concept c that is required by s , $\exists s' \in K$ s.t. s' also requires concept c .

introduction, if there exists some concept c s.t. c is required by s but is not required by any problem in K .

In other words, the next problem in the progression will be classified as reinforcement if the student has learned some problem that is at least as hard as it, as introduction if it requires any new concepts (basic skills) that the student has never learned, and as recombination if it combines together previously learned concepts in a new way.

For example, if the current knowledge boundary is:

$$K = \{A, BCD, AFFD, BBEECC\}$$

The following problems will be classified as:

reinforcement: $A, BD, AFD, BBEC$

recombination: $AB, BFE, BBEECC, AFDD$

introduction: $G, BHD, BBEGC, GH$

We can think of reinforcement as the review of learned knowledge, introduction as

pure learning, and recombination as a mixture of both. The designer’s strategy of balancing learning and review in an educational progression is revealed by the proportions of these three types, which we refer to as *progression composition*.

5.1.2 Pace: the growth rate of knowledge

We would like to know if the difficulty of lessons (chapters, units, etc.) in a progression grows at a consistent speed such that students can learn smoothly. Instead of numerically measuring the difficulty of each lesson, we calculate the *knowledge size*, the total number of problems that have been introduced up until each lesson. This can be measured as the number of the problems in the universal problem set¹ S that are classified as reinforcement by the student’s knowledge boundary K after mastering all the knowledge from the start to the current lesson. The knowledge size $|K|$ is therefore the number of such problems:

$$|K| = |\{s \in S \wedge \exists s' \in K \text{ s.t. } s' \geq s\}| \quad (5.1)$$

The *pace* is the ratio between knowledge growth (change of knowledge size $\Delta|K|$) and time. Assuming that the number of problems that students learn is directly proportional to the amount of expended time, we can use the length of the lesson $|L|$, which measures the number of practice problems in it, to represent time. Hence, the pace of lesson i can be measured as:

$$Pace_i = \frac{|K_i| - |K_{i-1}|}{|L_i|} \quad (5.2)$$

where $|K_i|$ and $|K_{i-1}|$ indicate the knowledge sizes after lesson i and lesson $i - 1$.

For example, assume the knowledge size after lesson 5 is 110, and the knowledge

¹The universal problem set S contains all the practice problems we have for an educational task.

Progression Composition of Japanese Textbooks (Unit 1-6)

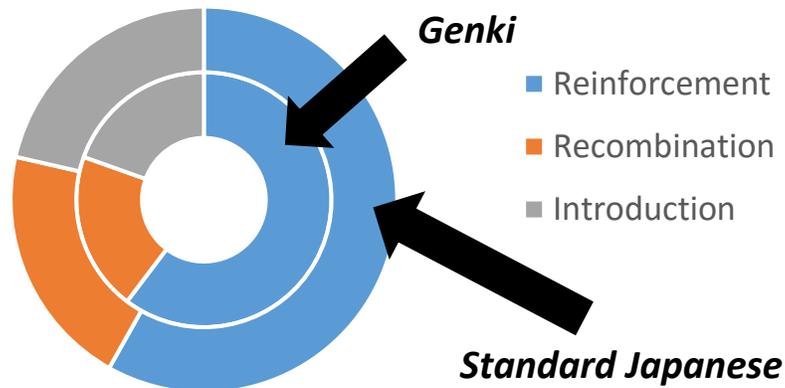


Figure 5.1: Progression composition of two Japanese textbooks: *Standard Japanese* and *Genki*. The proportions are fairly similar. Note that it usually takes a student 2 semesters to learn units 1-6.

size after lesson 6 is 134. If lesson 6 has 8 problems, then the pace of lesson 6 is $(134 - 110)/8 = 3$.

5.1.3 Studies on textbooks

To validate our work, we study two popular Japanese textbooks: *Standard Japanese* [68] and *Genki* [8]. Note that in language learning, ‘problems’ are the practice sentences that are introduced throughout the curriculum. Our goal is to find out if the progressions of both textbooks have similar pace and composition.

Both textbook progressions have similar composition

Figure 5.1 shows the progression compositions of the first 6 units (which usually takes a student two semesters to learn) in these two Japanese textbooks. Clearly, both text-

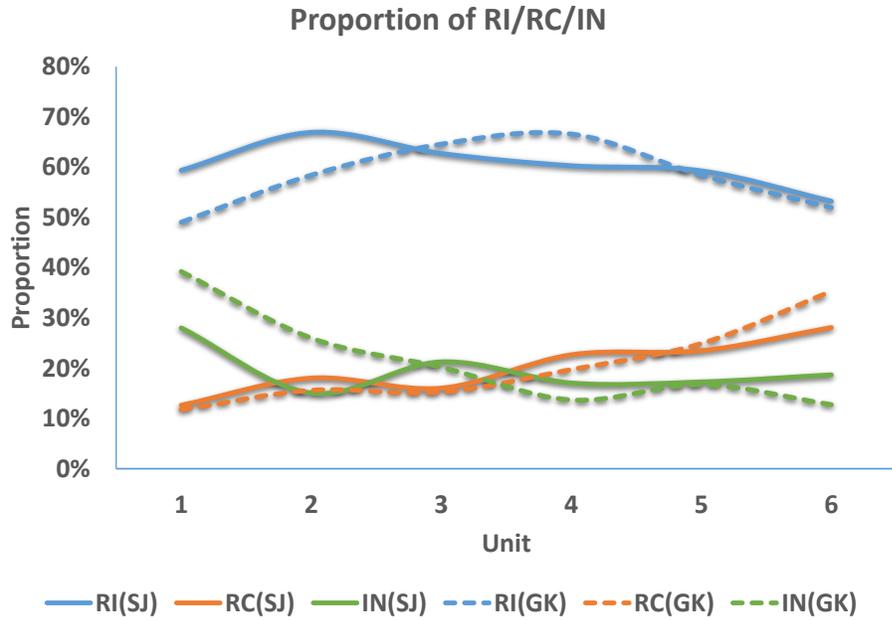
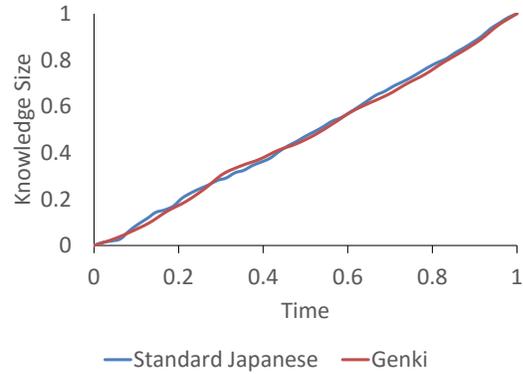


Figure 5.2: Proportion of reinforcement(RI), recombination(RC) and Introduction(IN) in two series of Japanese textbooks: *Standard Japanese* (SJ) and *Genki*(GK). Here x-axis represents the learning progress: unit 1-6, and y-axis demonstrates the proportions.

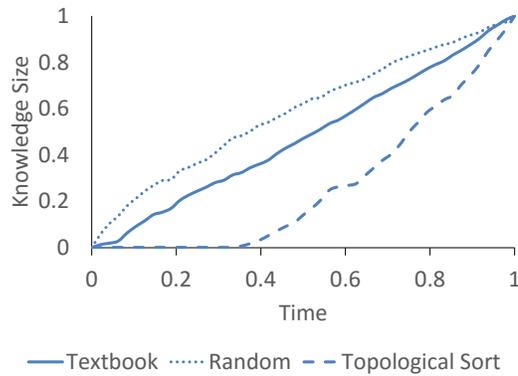
book progressions have a similar composition, possibly revealing that the experts who designed these two books arrived at similar conclusions.

To analyze how the progression composition changes over time, we calculate the proportions of reinforcement, recombination and introduction of each unit of both textbooks. For the proportions of unit i , we generate the knowledge boundary K_{i-1} based on unit 1 to unit $i - 1$, and classify all sentences in unit i based on K_{i-1} .

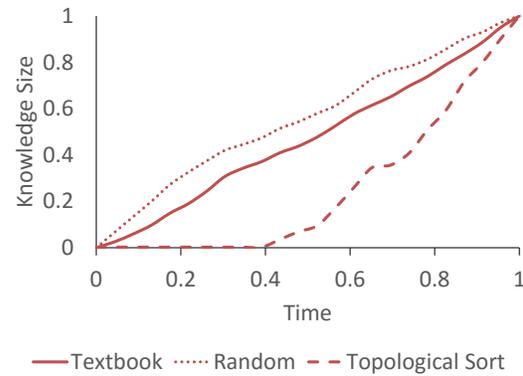
Figure 5.2 shows how the composition of these two textbook progressions changes over time. Clearly, there are some striking similarities between the curves of reinforcement, recombination and introduction: for both series of books, the proportion of reinforcement increases to 70% then drops to 50%, the proportion of recombination increases from 10% to 30%-35%, and the proportion of introduction decreases from 30%-



(a) Paces of Textbook Progressions: *Standard Japanese* and *Genki*.



(b) Standard Japanese Texts: Paces of the textbook progression, and the progressions generated by the random algorithm and topological sort.



(c) Genki Texts: Paces of the textbook progression, and the progressions generated by the random algorithm and topological sort.

Figure 5.3: Paces of various progressions. Here, the x-axis represents the normalized learning time (namely, the normalized number of sentences learned), and the y-axis corresponds to the normalized knowledge size. (a) shows that both textbooks introduce the same amount of knowledge over the same period of time, which indicates that they use a steady pace. (b) and (c) compare the textbook progression to other baseline progressions. When compared with the textbook progression, random progressions are too difficult at the beginning, and progressions generated by a topological sort are too easy at the beginning.

40% to 15%-20%. This validates the usefulness of characterizing progression composition, as there may be general principles of balancing learning and review that can be learned from existing textbooks.

Both textbook progressions have a similar, steady pace

Figure 5.3(a) shows the paces of both textbooks. Clearly, they both introduce the same amount of knowledge over the same period of time, which demonstrates they use a steady pace. This fits our intuition that learning is most comfortable with a steady pace.

To test if other reasonable progressions also have this property, we generate two alternate progressions by shuffling the sentences. The first alternate progression is generated randomly, and the second follows a topological sort. The topological sort ensures that any sentence s is put before all sentences that are harder than s . We compare the textbook progressions with the progressions generated by the random algorithm and topological sort (Figure 5.3(b) and Figure 5.3(c)). The random progression is too difficult at the beginning because hard and easy sentences are equally likely to be selected. This may cause students to become discouraged by hard sentences at the beginning. On the other hand, the topological sort is too easy at the beginning because all of the easy sentences are put first. This results in an increase of difficulty that is initially too slow. As can be seen from our analyses of these two textbooks, steady pacing may be an important characteristic of a good progression.

5.1.4 Studies on online language learning tools

We also analyze the progressions used by two online Spanish learning tools: Duolingo² and Language Zen³. We observed that some of the interesting differences in these two language learning progressions resulted from how they introduce vocabulary. Therefore, instead of using grammatical templates for Spanish, we treated each unique word as its

²<https://www.duolingo.com/course/es/en/Learn-Spanish-Online>

³www.languagezen.com

Progression Composition of Online Spanish Tutoring Systems (First 30 Problems)

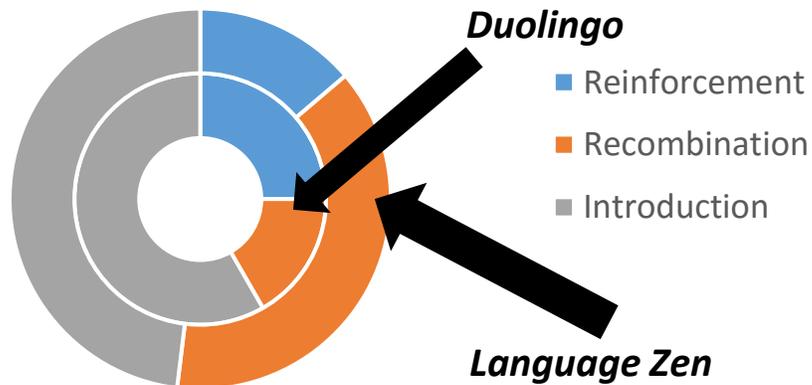


Figure 5.4: Progression composition of two online Spanish learning tools: Duolingo and Language Zen. Compared with Duolingo, Language Zen has less reinforcement and less introduction, but much more recombination. Note that it usually takes a student a couple of days to learn 30 problems.

own concept, and calculated the knowledge boundary and progression parameters using vocabulary.

The progression compositions of the first 30 problems (which usually take a student a couple of days to learn) introduced in these two tools are shown in Figure 5.4. Compared with Duolingo, Language Zen has less reinforcement and less introduction, but much more recombination. It has been claimed⁴ that Language Zen is 1.5 times faster than Duolingo. We speculate that the higher proportion of recombination may contribute to this difference.

One may also notice that Figure 5.1 has a higher proportion of reinforcement and a lower proportion of introduction than Figure 5.4. There are three factors we believe that could contribute to these differences. One factor is the learning method. Textbooks are

⁴<http://elmodenafrontline.com/8825/feature/language-zen-the-new-and-coming/>

typically used in classroom learning and are thus not necessarily intended to be the only learning material that students are using. They are typically combined with lectures and assignments. However, online language learning tools such as Duolingo and Language Zen are more self-contained. Therefore, these textbooks may have a higher proportion of reinforcement (review) than these online tutoring systems.

The second factor is the difference of the learning period. The texts in Figure 5.1 are usually taught over two semesters, while Figure 5.4 only shows the sentences that are taught in the first couple of days of the online learning tools. The differences in progression composition in the two figures imply that these expert-designed progressions tend to focus more on introduction than reinforcement at the very beginning of the progression. This is also consistent with Figure 5.2, where the proportion of introduction is the highest at the beginning then drops afterwards.

The third factor is that Figure 5.1 examines grammar and Figure 5.4 examines vocabulary, which could imply that grammar needs more reinforcement than vocabulary.

5.2 Parameter-based progression synthesis

We have discussed two features of an educational progression: composition and pace. In this section, we will demonstrate that educational progressions can be automatically synthesized according to specific composition and pace parameters. To be more precise, a progression can be characterized as three numeric parameters: overall pace (*pace*), proportion of reinforcement (*ri*), and proportion of introduction (*in*). The proportion of recombination (*rc*) is redundant since $ri + rc + in = 1$.

We use a greedy algorithm to synthesize progressions. The algorithm starts with

an empty progression, repeatedly selects the next problem that minimizes the following error function and appends it to the progression:

$$Error = (pace - pace^*)^2 + (ri - ri^*)^2 + (in - in^*)^2 \quad (5.3)$$

where $pace, ri, in$ are the actual progression characteristics, and $pace^*, ri^*, in^*$ are the desired progression parameters. Note that there may be multiple available problems with minimal error. In that case, our algorithm randomly selects one from them as the next problem.

This greedy algorithm does not always generate progressions with the exact desired parameters. However, it runs very fast. For a universal problem set of 25 problems, it can synthesize over 100 progressions (with 10 problems each) in one second. Therefore, we can run this greedy algorithm many times in order to synthesize progressions with desired characteristics.

Table 5.1 shows four examples of synthesized progressions. We extracted the following parameters from the *Genki* progression: $pace = 0.465, ri = 0.582, in = 0.214$. We then used these parameters (approximately) to synthesize the ‘*Genki* Simulation’ progression. We also generated three other progressions with tailored parameter settings, which are biased towards reinforcement, recombination and introduction. These results demonstrate that we can not only utilize the principles of expert progression to design and synthesize progressions with parameters that are good for most students, but also tailor progressions for students with specific preferences.

Progression Name	Desired Progression Parameters	Synthesized Progression
<i>Genki</i> Simulation	$pace = 0.5, ri = 0.6, in = 0.2$	DG B DG DG DG BG BG B BD B
Reinforcement-biased	$pace = 0.3, ri = 0.7, in = 0.2$	C C C C C A C A AC A
Recombination-biased	$pace = 1.2, ri = 0.2, in = 0.3$	G G AB BG AG BG EF BE AE BF
Introduction-biased	$pace = 1.6, ri = 0.2, in = 0.5$	EF EF AB BF ABC AF CDE AF BCD DG

Table 5.1: Synthesized progressions with desired parameters. The ‘*Genki* Simulation’ progression is synthesized with parameters extracted from the textbook. The other three progressions are generated with specific parameter settings. This demonstrates the potential of our progression synthesis framework to map characteristics of one progression onto another, and to create progressions that are tailored to the needs of an individual student.

5.3 Goal-based progression design

Engagement comes from not only a steady increase of difficulty but also a feeling of accomplishment after completing hard tasks. Shigeru Miyamoto, designer of the famous video game *Mario*, suggested that people need to have “a sense of satisfaction of completing something” in order to be engaged by a video game (video [93], 0:51). For this reason, many game progressions have “periods of intensity” and “periods of rest”, allowing players to use newly-acquired skills to complete difficult challenges and feel a sense of accomplishment. These “periods of intensity” may occur in “boss levels” that are significantly harder. However, previous progression synthesis work in educational games has not paid much attention to these intentional variations in difficulty that help create a sense of accomplishment.

Therefore, we focus our work on *goal-based* progression synthesis for language learning games. We believe that one important characteristic of a good progression is *goal-drivenness*: the progression should build up to complex goals quickly. Although students are not initially prepared to tackle hard tasks, ideally the progression should build up to them as quickly as possible such that players can feel a sense of accom-

	Progression	Translation
1	tengo	I have
2	tengo	I have
3	más	(any) more
4	tengo más	I have more
5	dos	two
6	tengo dos	I have two
7	de	than
8	tengo más de dos	I have more than two

Table 5.2: First eight tasks of Language Zen [111], an online Spanish learning tool (as of December 2018).

plishment. This is a pattern we have observed in the Language Zen [111], an online Spanish learning tool. Table 5.2 shows the progression (first eight tasks) of Language Zen. Clearly, this progression tries to build as quickly as possible to the sentence “tengo más de dos”. It does this by breaking the sentence into smaller valid phrases (“tengo más” and “tengo dos”), and builds up to these phrases by introducing vocabulary words individually (“tengo”, “más”, “dos” and “de”). We would like to apply this pattern to automatic progression synthesis.

Another common drawback in previous work is that their synthesized progressions did not intentionally help people review previously learned knowledge. Students need to practice their newly acquired skills, and a good progression should schedule reinforcement for those skills appropriately. Moreover, we also need to consider that students may prefer different progression lengths: some students may favor a shorter and faster learning pace while others may prefer longer progressions with more reinforcement. Ideally, progression synthesis algorithms should be able to create progressions of different lengths in order to accommodate different student preferences. Although we don’t intend to tailor progression length for each individual player since adaptation is not one of the contributions of this section, we still want to find an approach to synthesize a goal-based progression of a pre-specified length.

In this section, we propose an automatic approach for synthesizing goal-based progressions in educational games for language learning. This algorithm builds on previous work of hierarchical knowledge structuring and models all game levels in a directed graph based on their relative difficulty. To help students complete the hardest levels (the “learning goals”), the algorithm runs a post-order Depth-First Search (DFS) to generate a fast-paced, reinforcement-free progression in which students can achieve smaller sub-goals as soon as possible, so that they can feel a sense of accomplishment while progressing to the final goals. The algorithm also adds an appropriate amount of duplicate levels to the reinforcement-free progression based on existing theories of forgetting curves [26] and reinforcement scheduling [31,66,67] in order to help students review the knowledge that they may have forgotten. The amount of review levels can be controlled by the pre-specified progression length.

In order to evaluate our novel progression synthesis approach, we designed *Katchi*, a Korean language learning puzzle game that teaches basic vocabulary such as colors, numbers, shapes and conjunctions. *Katchi* is specifically designed to be highly parameterizable, so that all aspects of each puzzle can be controlled by a progression control algorithm. Using *Katchi*, we conducted an evaluation with 248 participants, and the results demonstrated that our synthesized progression performed similarly to an expert-designed progression in terms of both the total time played, a key metric of engagement, and the amount of knowledge learned, our metric of learning effectiveness. This indicated that our algorithm is capable of automatically synthesizing goal-based progressions that are comparable to manually created progressions. Moreover, further analysis of the *Katchi* user data demonstrated the potential of modeling the student’s desired learning pace and tailoring progressions accordingly.

5.3.1 The two-step heuristic approach

There are multiple characteristics of a progression that make it engaging. One is *pacing*: the difficulty should increase proportionally with the learner’s increasing skill. We hypothesize that another important characteristic is *goal-drivenness*: the progression should build up to a complex goal quickly. Learners want a return on their investment: although they are not initially prepared to tackle complex problems, ideally the progression should build up to them as quickly as possible such that learners can feel a sense of accomplishment. This is a pattern that we have observed in the progression (first eight tasks) of Language Zen [111] (Table 5.2). As we discussed at the beginning of this section, the progression of Language Zen builds up to a hard sentence (the final learning goal) by breaking it into smaller valid phrases (sub-goals) and individual vocabulary words (atomic conceptual units). By doing this, students can complete some sub-goals and feel a sense of accomplishment before they reach the final goal.

We would like to apply this pattern to automatically generate level progressions for educational games. In this section, we propose a two-step heuristic algorithm for synthesizing level progressions that build up towards a specific learning goal. Our algorithm first generates a fast, reinforcement-free progression using post-order Depth-First Search (DFS) [102], then adds reinforcement levels to it in order to generate progressions of the pre-specified length. The rest of this section will elaborate on these two steps.

Generating reinforcement-free progressions

To help students complete a specific goal level s^* , we need to gradually introduce levels that are easier than s^* . Let V be the set of levels that includes s^* and those easier than s^* .

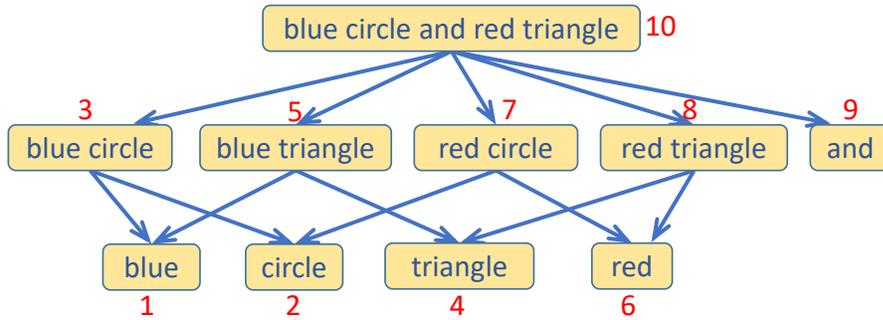


Figure 5.5: A sample partial ordering graph in goal-based progression synthesis. Each node represents a level containing a specific set of required concepts. Each directed edge represents a “harder than” relation between two levels. The red numbers next to each node denote the traversal order of each level in the post-order DFS, namely the order of each level in the reinforcement-free progression.

Our goal-based progression synthesis algorithm needs to study the relationship among levels in V , such that it can decide in which order to introduce each of those levels.

To this end, we model the hierarchical structure of our level pool V with a *partial ordering graph* [99]: each node represents a level in V , and each edge connects two nodes (levels) if one level is “harder than” another. Here level p_1 is “harder than” level p_2 if and only if p_1 covers all the required concepts in p_2 . For example, the level “two red triangles” is harder than the level “two triangles” and the level “red triangle”, so there should be an edge from “two red triangles” to “two triangles” and an edge from “two red triangles” to “red triangle”. Figure 5.5 shows an example of this partial ordering graph. Note that this is a rooted graph [103], meaning that there is at least one directed path from the root s^* to each node in this graph.

We generate a reinforcement-free level progression using post-order DFS. Here “reinforcement-free” means there are no duplicate levels in the progression and students will only experience new levels. The post-order DFS runs as follows: when visiting a (sub)graph rooted at node s , it first recursively visits the subgraphs rooted at each un-

visited child of s , then visits node s itself. Each node will be visited only once. For example, the post-order DFS traversal order of the partial ordering graph in Figure 5.5 is: *blue, circle, blue circle, triangle, blue triangle, red, red circle, red triangle, and, blue circle and red triangle.*

The reinforcement-free progression generated by post-order DFS is topologically sorted, which means that harder levels will always be introduced after easier levels [4, 76]. However, progressions without reinforcement are usually too fast⁵, since students lack the chance to review previously learned knowledge. We will discuss how to slow down reinforcement-free progressions in the next section.

Adding reinforcement

Students may forget some knowledge they have learned, hence we need to add reinforcement to the reinforcement-free progressions in order to help students review. Since the amount of reinforcement is controlled by the pre-specified progression length (learning pace), the remaining challenge of scheduling reinforcement in a progression is to decide which levels to reinforce and when to reinforce them.

Ebbinghaus modeled the student's memory with the *Forgetting Curve* [26], which stipulates that the memory retention R is exponential to memory strength S and time t :

$$R = e^{-\frac{t}{S}} \quad (5.4)$$

To calculate the memory retention of a level in the progression, we let the memory strength S refer to how many times the level has been reinforced so far. Note that a level can be reinforced by itself or other levels that are harder. Time t refers to the number of

⁵In goal-based progression synthesis, *progression length* and *learning pace* are equivalent if the goal is fixed: a long progression implies a slow learning pace and a short progression implies a fast learning pace.

Require: reinforcement-free progression P_0 and
minimum memory retention parameter $minR$

Ensure: longer and slower progression P

- 1: **Initialization:** $P \leftarrow []$, $pointer \leftarrow 1$
- 2: **while** $pointer \leq \text{length of } P_0$ **do**
- 3: $s \leftarrow$ the problem in P with the least memory retention
- 4: **if** $P \neq []$ **and** memory retention of $s < minR$ **then**
- 5: Append s to P
- 6: **else**
- 7: Append $P_0[pointer]$ to P , $pointer \leftarrow pointer + 1$
- 8: **end if**
- 9: Update memory retention for all problems in P
- 10: **end while**
- 11: **return** target progression P

Algorithm 2: Scaling a reinforcement-free progression into a longer and slower progression according to parameter $minR$

levels completed in the progression since the level was reinforced most recently up til now. For example, assume the progression is currently A, B, AB, C, BC, AC (each letter represents a conceptual unit and each string represents a level containing those units). For level B, memory strength S is 3 (B, AB and BC reinforced B three times) and time t is 1 (BC reinforced B most recently), hence the memory retention of B is $e^{-1/3}$.

In order to accommodate different desired progression lengths, we did not use fixed reinforcement intervals in our algorithm. Instead, we set a parameter of minimum memory retention $minR$ in progression synthesis: if the memory retention of a previously introduced level is smaller than $minR$, then we believe the student has forgotten how to solve that level. This idea is inspired by existing work in spaced repetition [31, 66, 67]. However, as far as we know, our work is the first to incorporate reinforcement scheduling techniques in the synthesis of level progressions that gradually increase in difficulty.

Algorithm 2 elaborates how we scale the reinforcement-free progression P_0 into a longer and slower progression P given the minimum memory retention parameter $minR$.

	Reinforcement-free progression	Target Progression	
		$minR = e^{-3}$	$minR = e^{-2}$
1	red	red	red
2	circle	circle	circle
3	white	white	white
4	red circle	red circle	<i>red</i>
5	white circle	white circle	<i>circle</i>
6	triangle	triangle	<i>white</i>
7	red triangle	red triangle	red circle
8	white triangle	<i>red circle</i>	white circle
9		<i>white circle</i>	triangle
10		white triangle	<i>red circle</i>
11			<i>white circle</i>
12			<i>triangle</i>
13			red triangle
14			white triangle

Table 5.3: Sample Results of Algorithm 2, which scales a reinforcement-free progression into longer and slower progressions given the minimum memory retention parameter $minR$. Levels in italics are added by the algorithm to help student review. The parameter $minR$ controls the progression length (learning pace): the larger the parameter $minR$, the longer the target progression.

This algorithm starts with an empty progression (line 1) and then adds levels to the target progression P (recall that the order of levels in P is the order in which we will introduce them to students). During this process, the algorithm also keeps track of the memory retention of all levels that have already been introduced in P (line 9) in order to decide which levels to reinforce and when to reinforce them. If at any time there exists at least one previously introduced level with memory retention smaller than $minR$ (line 4), then the algorithm adds the level with the least memory retention such that students can review that level (line 5). Otherwise, it sequentially adds the level at the tail of the reinforcement-free progression P_0 in order to introduce a new level (line 7). Table 5.3 shows sample results of this algorithm.

Generally, the larger the minimum memory retention $minR$, the more likely our al-

gorithm recognizes that students have forgotten some level. Therefore, the algorithm reinforces that level and the target progression P becomes longer. To generate progressions with the desired length, we can simply use a binary search to find the best parameter $minR$ that generates an optimal progression with length closest to the desired amount.

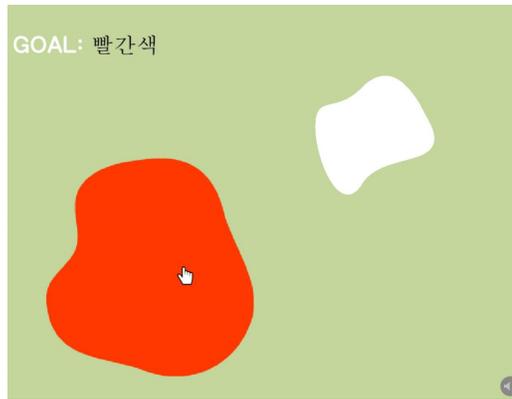
This two-step algorithm can be scaled up to the cases when the learning goal is to solve multiple levels instead of only one. We can simply add a pseudo-root node that links to all goal levels in the partial ordering graph. By doing this, we can transform a multi-goal case into a unique-goal case and our algorithm will work naturally.

5.3.2 Evaluation

Katchi

In order to evaluate different progressions, we designed *Katchi*, a Korean learning puzzle game that teaches basic Korean vocabulary such as colors, numbers, shapes and conjunctions. Each level in *Katchi* helps students understand a Korean phrase such as “one green triangle and two red circles”. Generally, *Katchi* helps students practice the skills of comprehension (“input”) and construction (“output”) of the Korean language. To do this, *Katchi* has two types of levels: *selection levels*, in which the player must choose the matching object from a set of objects matching the phrase shown on the top-left of the screen (Figure 5.6(a)), and *construction levels*, in which the player is asked to construct a phrase describing an object or set of objects by dragging-and-dropping the correct Korean word(s) from a bank of words (Figure 5.6(b)).

Katchi is designed to be highly parameterizable so that a progression control al-



(a) A selection Level: players need to either select the correct graphic object based on the given Korean word(s) (“red”) on the top-left of the screen.



(b) A construction Level: players need to drag-and-drop the correct Korean word(s) to describe the target graphic objects (“one white square”) shown in the middle of the screen.

Figure 5.6: Screenshots of *Katchi*, a Korean learning puzzle game that teaches basic vocabulary such as colors, numbers, shapes and conjunctions. There are two types of game levels in *Katchi*: *selection levels*, in which players need to either select the correct graphic object based on the given Korean word(s); *construction levels*, in which players need to drag-and-drop the correct Korean word(s) to describe the target objects.

gorithm can specify all aspects of each puzzle game level: the Korean phrase that the level corresponds to, the number of distractor objects in selection levels or the number of distractor words in construction levels, the number of hints (slots where the correct words are already filled in and locked, e.g. the last slot on dark-gray background in Figure 5.6(b)), and the assigned difficulty score.

We wrote a algorithm to generate levels for *Katchi*. The game is built around a specific grammatical structure, namely, noun phrases that build up to adjective, noun, number, conjunction, adjective, noun, number. For example, “three triangles” or “two green triangles” or “one green triangle and two red circles”. The game also includes a set of vocabulary words that correspond to each of these categories. For example, the color (adjective) category includes “green”, “red”, and “white”, and the shape (noun) category includes “triangle”, “circle” and “square”. To generate levels for *Katchi*, we

enumerate all possible combinations of adjectives, nouns, numbers, etc. that fit within the noun phrase grammatical structure, as well as different numbers of distractors and hints. For the set of words and the grammatical structure specified above, this algorithm produces a pool of 852 levels. A valid *Katchi* progression can be created by selecting a certain amount of levels from this level pool and arranging them appropriately.

We have successfully deployed *Katchi* on both Reddit [73] and Newgrounds [62]. We will discuss the evaluation of our goal-based progression synthesis algorithm using *Katchi* in the next section.

Experiment

In order to evaluate whether our goal-based progression synthesis algorithm can automatically produce progressions that are comparable to human-designed progressions, we applied our algorithm to synthesize a progression for *Katchi* and compared it to an expert-crafted progression with the same length and the same goals. We specified the goals as the following four phrases:

- *two red triangles and two green circles*
- *three red triangles and three white triangles*
- *one white square and two green squares*
- *two white triangles and three green squares*

Note that in Korean, nouns generally do not have plural forms. For readability, we add plurals to the English translation in this chapter although in Korean there is actually no difference between *triangle* and *triangles*.

In order to help students understand these four goal phrases, a Korean native speaker and one of the designers of *Katchi* crafted a manual progression with 54 game levels. To compare, we used our progression synthesis algorithm to generate an automatic progression with 54 levels that also build towards these four goal phrases. Our algorithm considers levels to be variations of the same problem if they require students to utilize the same set of Korean vocabulary, even if they vary in terms of the number of incorrect “distractor” words to choose from. However, our algorithm always puts the easier variations before the harder variations in the synthesized progression when integrating into the game. We present the expert progression and the synthesized progression in Table 5.4. We noticed a clear difference between two progressions: our synthesized progression introduces more complex levels early on and starts to introduce (hard) goal levels in the middle phases, whereas the expert progression tends to introduce easy levels at the start and all (hard) goal levels at the end.

Comparing synthesized and expert progressions

We hope that our algorithm can automatically generate progressions that are comparable to the expert-designed progression in terms of engagement. To validate this, we compared the total time users played our game for the two progressions. We recruited 248 users from the Language Learning and the Korean Sub-reddits [73, 74]. 122 randomly selected users played the expert-designed progression and 126 users played our synthesized progression. Since our data was not normally distributed, we used the Wilcoxon-Kruskal-Wallis test. We found the median time played is 42 seconds for the synthesized progression and 45 seconds for the expert-designed progression. These results are visualized in Figure 5.7. The synthesized progression performed similarly to the expert-designed progression based on engagement: the median time played was only 7% lower

	Expert Progression	Synthesize Progression
1	red	one
2	white	white
3	green	square
4	red	white square
5	white	one white square
6	green	green
7	red and white	green square
8	white and green	one green square
9	red and green	two
10	white	two white squares
11	green	two green squares
12	red	white and green
13	green	one white square
14	circle	white square and green square
15	triangle	one white square and two green squares
16	square	red
17	triangle	triangle
18	circle	red triangle
19	triangle	two red triangles
20	square	circle
21	triangle	red circle
22	circle	two red circles
23	circle and triangle	two white squares and one green square
24	triangle and square	green triangle
25	circle and square	two green triangles
26	red circle	green circle
27	red circle	two red triangles
28	red circle	two green circles
29	red triangle	red and green
30	green circle	two red circles
31	green triangle	circle and triangle
32	white triangle	red triangle and green circle
33	white square	two green triangles
34	one	two red triangles and two green circles
35	two	white triangle
36	one	two white triangles
37	two	three
38	three	three white triangles
39	two	one white square and two green squares
40	two	three white squares
41	three	three green triangles
42	one red square	two red circles and two green triangles
43	one white circle	three green squares
44	one green triangle	two white triangles
45	two white circles	triangle and square
46	one green circle	three white triangles
47	red triangle and white triangle	white triangle and green square
48	white square and green circle	three white squares
49	red circle and green triangle	three green triangles
50	two white squares	two white triangles and three green squares
51	two red triangles and two green circles	three red triangles
52	three red triangles and three white triangles	red and white
53	one white square and two green squares	red triangle and white triangle
54	two white triangles and three green squares	three red triangles and three white triangles

Table 5.4: Expert progression and synthesized progression in *Katchi*. Goal levels are in bold. Note that the expert progression puts all goal levels at its end whereas synthesized progression introduces goal levels starting at level 15 and reinforces them afterwards.

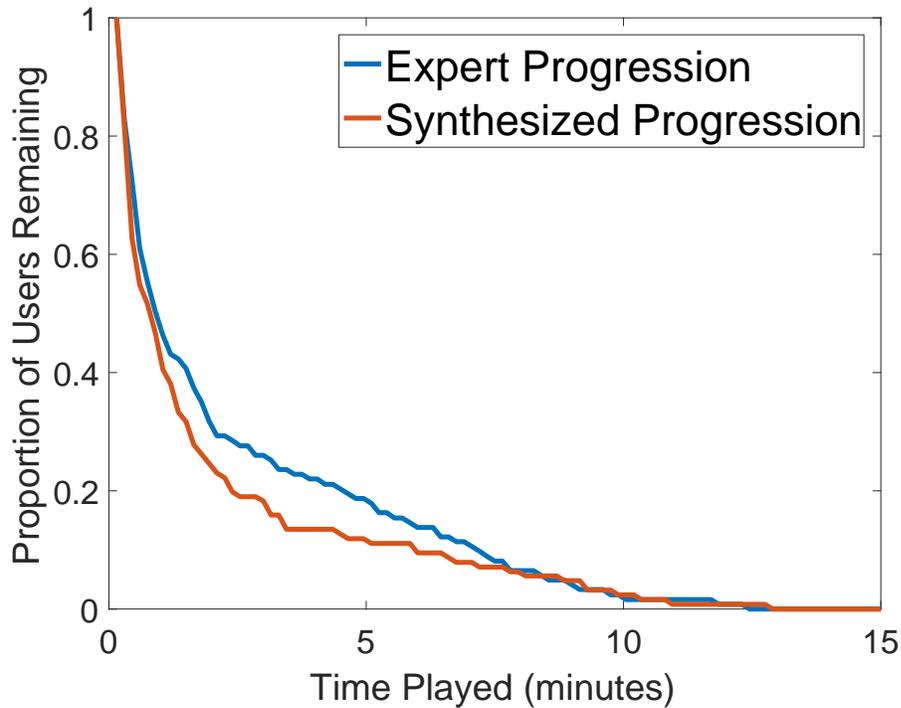


Figure 5.7: Proportion of users remaining after playing our game for certain amounts of time. We did not find a statistically significant difference between the two progressions. It appears that the synthesized progression engaged players slightly less than the expert progression, but they are quite comparable. The median player in the synthesized progression played for 42 seconds and the median player in the expert-designed progression played for 45 seconds.

for the synthesized progression than the expert-design progression and this difference was not statistically significant ($p = .185$, $r = .084$).

We also hope that our synthesized progression is comparable to the expert designed progression in terms of learning effectiveness. To validate this, we compared the total amount of knowledge users learned after playing our game for the two progressions. We use *knowledge size*, a concept defined in Equation (5.1), to measure how much a student has learned after finishing certain amount of levels: *a student's knowledge size is defined as the number of tasks that either have been completed or are easier than some completed tasks*. In other words, if a students has learned and finished the

level “two white square”, then the level “white square” will also be considered solvable by the student. We present the student’s knowledge size for the two progressions in Figure 5.8. The Wilcoxon-Kruskal-Wallis test suggested that the two progressions were quite similar in terms of knowledge size: the difference between the two progressions was not statistically significant ($p = .472$, $r = .046$). The median knowledge size is higher for the synthesized progression (Median = 5.5) than the expert progression (Median = 5).

We noticed that our synthesized progression performed worse than the expert progression in the middle stage of the game. For example, Time Played between 3 and 6 minutes in Figure 5.7 and Student’s Knowledge Size between 10 to 20 in Figure 5.8. We suspect that this is because our synthesized progression was too hard for some intermediate students. Our algorithm always builds up towards goal levels as fast as possible, yet this might be too aggressive for some students. In the future, we hope to refine our algorithm and generate progressions that are slightly less goal-driven. Ideally, our algorithm would take “degree of goal-drivenness” as an extra parameter and generate multiple progressions to accommodate different student preferences.

This work improves over existing work in automatically generating progressions that are comparable to expert-designed progressions [12] in three aspects. First, existing work only looks at the sequential information of each problem and neglects the structural information contained within the problem library, whereas our progression synthesis algorithm utilizes the hierarchical knowledge structure of the problem library.

In addition to the expert-designed progression and synthesized progression discussed above, our experiment simultaneously tested another synthesized progression that was slightly longer (67 levels instead of 54). We did not include this in our analysis because comparison of progressions with different lengths is not the main focus of this project. This third progression performed somewhat worse than the other two (median time played was 34 seconds and median knowledge size was 4).

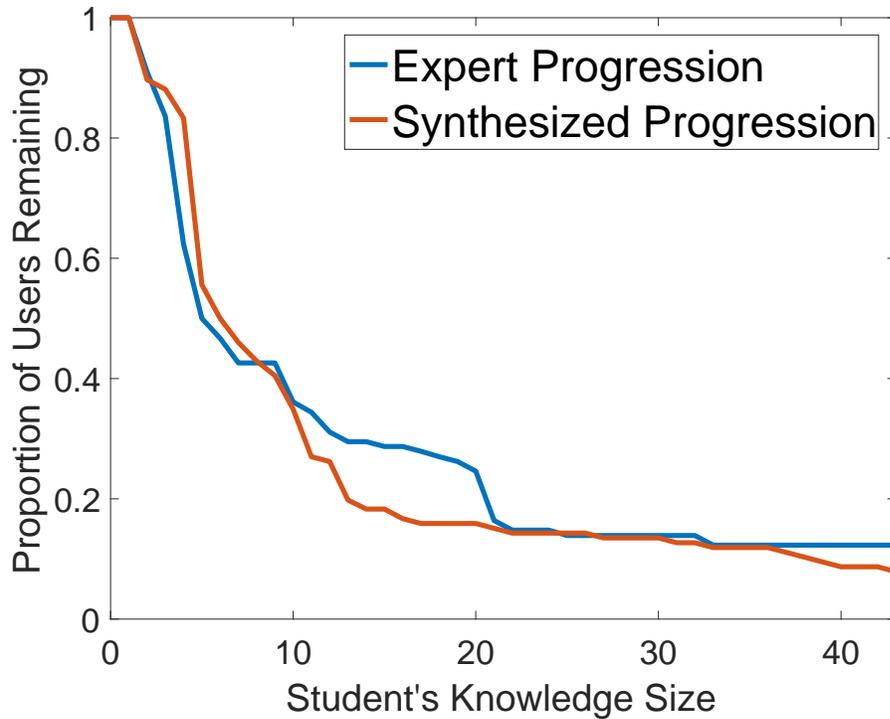


Figure 5.8: Proportion of users remaining after learning certain amounts of knowledge. We use *knowledge size*, number of levels that either have been completed or are easier than some completed levels, to measure the amount of knowledge learned. There is no statistically significant difference between the two progressions. The proportion of players with knowledge size between 10 and 20 is higher for the expert progression, while the two curves are quite similar otherwise. The median knowledge size is 5.5 for the synthesized progression and 5 for the expert-designed progression.

Second, our work applies *goal-drivenness*, a principle we observed in expert-authored progressions in order to automatically synthesize “general” progressions that are good for most students. Last, and most importantly, our algorithm incorporates reinforcement into the synthesized progressions such that students can review the knowledge they have learned.

Correlating time spent per level with quitting early

Our long-term goal is to vary the progression pace in order to accommodate students' different preferred learning paces and maximize engagement for each student. Inspired by work in dynamic difficulty adjustment [40, 87], we wanted to see whether there are early markers of players who might be more likely to quit, possibly because the pace of our synthesized progression is too hard or too easy. This would allow us to focus future interventions on those populations. Therefore, we studied the connections between student performance and engagement in *Katchi*. There were 31 players that finished at least 13 levels, which is 25% of the total game. For these players, we calculated the average time spent on each level.

Figure 5.9 shows these results. We observe that 11 out of the 12 players who finished at least 90% of the game spent 7-12 seconds on each level (green zone). There are 6 players who spent less than 7 seconds per level and left before 90% of the game (red zone). We suspect they quit early due to our progression being too fast for them and we should tailor a longer progression in order to accommodate their slower learning pace. On the other hand, 6 players spent more than 12 seconds per level and left before 90% of the game (orange zone). In this case, we suspect they left early due to our progression being too slow for them, and we should generate a shorter progression adapting to their faster learning pace. The remaining 7 players spent the typical time of 7-12 seconds per level but still left early, which is probably due to the lack of general interest in our game.

These results demonstrate a potential of modeling the student's preferred learning pace and tailoring progressions accordingly. We will explore this further in future work.

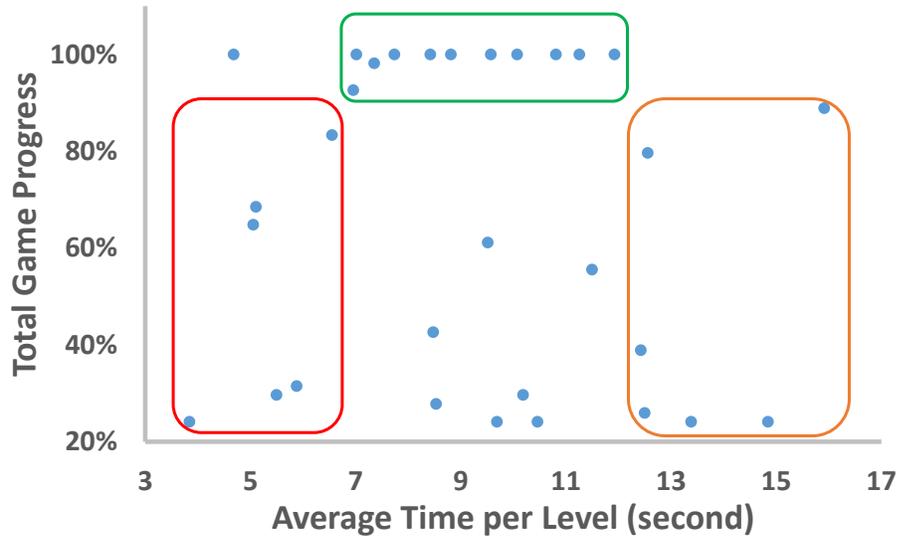


Figure 5.9: Total game progress over different average times spent on each level. Each blue plot represents a player. Typically, players who finished at least 90% of the game (green zone) spent 7-12 seconds on each level. Players who spent less than 7 seconds (red zone) or more than 12 seconds (orange zone) tended to leave early before 90% of the game, which probably indicates that we should generate a shorter/longer progression to accommodate their faster/slower learning pace respectively.

5.4 Conclusion

In this chapter, I studied expert-created learning progressions extracted from language learning textbooks and online tools, and discovered several of their characteristics: a steady pace, a certain proportion of learning new knowledge and reviewing previously learning knowledge, as well as the goal-drivenness. I presented a framework within the hierarchical knowledge structure (partial ordering graphs) in which existing progressions can be characterized with pacing and proportion parameters (section 5.1), and new progressions can be created based on a set of (reasonable) user-specific values of these parameters (section 5.2). Furthermore, I proposed a two-step heuristic algorithm to synthesize goal-based learning progressions with the user-specific length (section 5.3). I applied this progression synthesis algorithm into a Korean learning puzzle game and the

user study suggested that this algorithm can generate synthesized progressions comparable to expert progressions in terms of both student engagement and learning effectiveness. Moreover, further analysis of the user data demonstrated the potential of modeling the student's desired learning pace and tailoring progressions accordingly.

We hope these ideas will enable a science of progression analysis, in which sequencing and pacing parameters are extracted from progressions across a wide variety of topics to identify the best principles. In future work, we would like to refine our algorithm and incorporate extra parameters (such as a numeric parameter “degree of goal-drivenness”) in order to better accommodate different student preferences. Furthermore, we hope to build a tutoring system that automatically detects the student's prior knowledge and desired progression parameters, and designs adaptive and personalized progressions in real time.

CHAPTER 6

CONCLUSION

This thesis presents multiple frameworks, models and approaches for computer-assisted language learning. I showed that computers can take over several critical tasks in language education, which can potentially reduce the cost of human labor and make language learning more affordable and more engaging. Specifically, I showed how to build a hierarchical structure to organize the grammatical and vocabulary knowledge within a corpus of learning materials, and within this structure, how computers can accurately assess a student's knowledge, recommend learning materials, analyze existing expert-designed curricula, and synthesize progressions automatically. In the previous three chapters, I elaborated how computers can assist language learning in three perspectives: materials, students, and curricula. I will now revisit these ideas:

In Chapter 3, I proposed the *partial ordering graph* to hierarchically model the domain knowledge within a corpus of learning materials. This model is constructed based on the compositionality of each problem and the relative difficulty between pairs of problems. Specifically, I demonstrated that this model works for two important types of knowledge in language learning: grammar and vocabulary. I leveraged *grammatical templates* as the conceptual units to decompose the grammatical knowledge within a given text such that we can measure the grammatical difficulty of a text using a multiset of grammatical templates. Experimental results on the JLPT corpus indicated that these grammatical templates can be very useful for the prediction of text difficulty. I also relaxed the constraints that specify the relative difficulty between two texts, so as to ensure sufficient density in the hierarchical structure of vocabulary knowledge. The statistics on a large online corpus indicated that a *fuzzy partial ordering graph* is significantly denser than its strict version using a reasonable fuzzy parameter while the quality of the

fuzzy partial orderings is still acceptable according to our case study.

In Chapter 4, I presented an algorithm that interactively assesses a student's language knowledge, grammar or vocabulary, within the (fuzzy) partial ordering graph. I proposed the *knowledge boundary*, the set of hardest solvable problems, to model a student's knowledge and hypothesized that a student's performance on a specific problem can be predicted based on the distance between that problem to the student's knowledge boundary. The user data of *J100*, an online Japanese knowledge assessment tool, verified this hypothesis and indicated that this algorithm can efficiently measure the student's ability. Additionally, I also incorporated this assessment algorithm into the ZPD-based learning material recommendation heuristic. The balance of assessment and recommendation is carefully controlled by a probabilistic function in order to keep students from getting bored by either excessive assessment or too many inaccurately recommended materials. The experimental results of *JRec*, an online Japanese reading text recommendation tool, demonstrated that users in the adaptive recommendation group read significantly more texts than those in the non-adaptive group, which suggested that adding assessment significantly improved student engagement in reading material recommendation.

In Chapter 5, I studied on multiple expert-designed progressions (sequences of learning materials or practice problems) extracted from both traditional textbooks and online language learning tools. The study of those expert-designed progressions demonstrated that a good learning progression should have a steady pace, certain proportions of Reinforcement/Recombination/Introduction problems, and should ideally be goal-driven. Then I introduced a parameter-based algorithm to synthesize progressions with the given pacing/proportion parameters, indicating that the expert wisdom extracted from existing progressions can be incorporated into automatic progression synthesis. Further-

more, I proposed a two-step algorithm to generate goal-based learning progressions with specific lengths. These synthesized progressions build up towards some harder problems (learning goals, like “boss levels” in games) such that students can feel a sense of accomplishment while helping students to review what they have potentially forgotten throughout the learning process. The user study of Katchi, an online Korean learning game, suggested that the synthesized progression performed similarly compared to an expert-crafted progression with the same length, indicating that our algorithm is capable of synthesizing goal-based learning progressions that are comparable to expert-designed progressions.

6.1 Future work

In the future, we hope to refine and optimize our models to better characterize materials, students and curricula. We also hope to extend our theories and apply our approaches to a wider range of educational scenarios. Here are some research topics that we should explore further:

Multimedia Learning Materials Contemporary language education utilizes not only texts, but also audios and videos as learning content. There are multiple reasons to incorporate multimedia materials in language learning. First, some students may favor visual or oral content over textual content, and we need to prepare various types of materials to accommodate their preferences. Also, there are topics in language learning that are much better presented in the form of audio or video, such as “how to make a phone call” and “how to buy groceries”, thus adding audio and video materials can help students

I would like to thank Erik Andersen for his advice on the unfinished or ongoing projects mentioned in this section. I also want to express my appreciation to my collaborators for their work on these projects: Liye Zhong (Multimedia Learning Materials), Tong Mu, Emma Brunskill (Probabilistic Student Modeling within Hierarchical Knowledge Structure), and Sixian Yi (Explore More Parameters for Progressions).

grow language skills in a wider range of practical scenarios. Furthermore, incorporating multimedia materials increases the variability of the learning process. Students are less likely to get bored of repetitive and similar content and potentially get more engaged, if the tutoring system alternately recommends textual, audio and video materials to them.

Therefore, we hope to build a multimedia language tutoring system in future work that can prepare curricula containing all three types of media: texts, audios and videos. This is complicated because materials of different media contain different sets of language knowledge. Some knowledge is shared among materials of different media while others are distinct. For example, vocabulary knowledge is shared by all three media, pronunciation knowledge is shared by audio and video, and subtitles can only be found in videos. Ideally, the tutoring system would be able to organize both the common and distinct language skills among multimedia materials within the same model, such that the assessment of common language skills can be shared, and the tutoring system can move back and forth between distinct language skills across multimedia materials.

Probabilistic Student Modeling within Hierarchical Knowledge Structure In our current student model, each task or material will eventually be labeled with either “solvable” or “unsolvable” after assessment. However, this is not accurate since a student may fail to solve a problem even if the solution is within his/her ability, and conversely may “solve” a problem by making a guess even if he/she has not mastered all the prerequisite knowledge. To improve this, we could model the *likelihood* that a student can solve a problem and build a probabilistic student model within the hierarchical knowledge structure (partial ordering graph). We have implemented this model using some advanced techniques and sophisticated tools such as multi-armed bandit and ZPDES [60] and we are still working on a refined partial ordering graph that is capable of modeling student knowledge in a more accurate way.

Explore More Parameters for Progressions We hope to model a learning progression through a larger set of parameters and build a more comprehensive framework to characterize learning progressions. For instance, instead of classifying an educational task as either reinforcement, recommendation or introduction, we could take a close look at each prerequisite concept of the task and quantify its proportion of reinforcement/recommendation/introduction. Another parameter we could explore is the “degree of goal-drivenness”, which we could use to balance the trade-off between two heuristics: “going wide” and “going deep”.

Corpus-wide Progression Synthesis In order to engage language students over a long-term period, we plan to synthesize learning progressions for large authentic corpora. For example, we would like to build the progression for all NHK Easy articles in the past few months. It is worth mentioning that synthesizing a corpus-wide progression is different from synthesizing a parameter-based or goal-based progression, since we need to arrange all articles in a corpus and the synthesized progression will be much longer, making it even more necessary to help students review. Moreover, we need to take into consideration that students may read different numbers of texts each day, hence the synthesized progression should ideally accommodate different learning speeds.

There are two goals we want to reach while building corpus-wide progression: First, the progression should introduce new vocabulary smoothly. Namely, we want to minimize the number of new vocabulary words in each article. Second, we should timely help students to review the knowledge they may have forgotten. Specifically, if the student needs to reinforce a previously learned word at some point of the progression, then we should present an article with that word as quickly as possible.

BIBLIOGRAPHY

- [1] Dietrich Albert and Theo Held. Establishing knowledge spaces by systematical problem construction. In *Knowledge structures*, pages 81–115. Springer, 1994.
- [2] Vincent Aleven, Bruce M McLaren, Jonathan Sewall, and Kenneth R Koedinger. The cognitive tutor authoring tools (CTAT): preliminary evaluation of efficiency gains. In *International Conference on Intelligent Tutoring Systems*, pages 61–70. Springer, 2006.
- [3] Vincent Aleven, Bruce M McLaren, Jonathan Sewall, and Kenneth R Koedinger. A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education*, 19(2):105–154, 2009.
- [4] Erik Andersen, Sumit Gulwani, and Zoran Popovic. A trace-based framework for analyzing and synthesizing educational progressions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 773–782. ACM, 2013.
- [5] Andersen, Erik. *Bpan Yaa*. <https://www.cs.cornell.edu/~eland/games/thai/>.
- [6] John R Anderson, Albert T Corbett, Kenneth R Koedinger, and Ray Pelletier. Cognitive tutors: Lessons learned. *The journal of the learning sciences*, 4(2):167–207, 1995.
- [7] Anki. *Anki - powerful, intelligent flashcards*, 2019.
- [8] Eri Banno, Yoko Ikeda, and Yutaka Ohno. *GENKI: An Integrated Course in Elementary Japanese*. Japan Times and Tsai Fong Books, 2011.
- [9] Joseph E Beck, Kai-min Chang, Jack Mostow, and Albert Corbett. Does help help? introducing the bayesian evaluation and assessment methodology. In *International Conference on Intelligent Tutoring Systems*, pages 383–394. Springer, 2008.
- [10] Carl Blyth. A constructivist approach to grammar: Teaching teachers to teach aspect. *The Modern Language Journal*, 81(1):50–66, 1997.
- [11] Peter Brusilovsky and Nicola Henze. Open corpus adaptive educational hypermedia. In *The adaptive web*, pages 671–696. Springer, 2007.

- [12] Eric Butler, Erik Andersen, Adam M Smith, Sumit Gulwani, Zoran Popovic, and WA Redmond. Automatic game progression design through analysis of solution features. In *Proc. of the SIGCHI Conf. on Human Factors in Computing (CHI 2015)*, 2015.
- [13] Eric Butler, Adam M Smith, Yun-En Liu, and Zoran Popovic. A mixed-initiative tool for designing level progressions in games. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 377–386. ACM, 2013.
- [14] Jamie Callan and Maxine Eskenazi. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Proceedings of NAACL HLT*, pages 460–467, 2007.
- [15] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.
- [16] Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. Training and testing low-degree polynomial data mappings via linear SVM. *The Journal of Machine Learning Research*, 11:1471–1490, 2010.
- [17] Chih-Ming Chen, Shih-Hsun Hsu, Yi-Lun Li, and Chi-Jui Peng. Personalized intelligent m-learning system for supporting effective english learning. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 6, pages 4898–4903. IEEE, 2006.
- [18] Kevyn Collins-Thompson and James P Callan. A language modeling approach to predicting reading difficulty. In *HLT-NAACL*, pages 193–200, 2004.
- [19] Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.
- [20] Pedro Curto, Nuno Mamede, and Jorge Baptista. Assisting European Portuguese teaching: Linguistic features extraction and automatic readability classifier. In *Computer Supported Education*, pages 81–96. Springer, 2015.
- [21] Rafael Jaime De Ayala. *The theory and practice of item response theory*. Guilford Publications, 2013.
- [22] Michel C Desmarais, Peyman Meshkinfam, and Michel Gagnon. Learned stu-

- dent models with item to item knowledge structures. *User Modeling and User-Adapted Interaction*, 16(5):403–434, 2006.
- [23] Jean-Paul Doignon and Jean-Claude Falmagne. Spaces for the assessment of knowledge. *International journal of man-machine studies*, 23(2):175–196, 1985.
- [24] Jean-Paul Doignon and Jean-Claude Falmagne. *Knowledge spaces*. Springer Science & Business Media, 2012.
- [25] Fritz Drasgow and Charles L Hulin. Item response theory. *Handbook of industrial and organizational psychology*, 1:577–636, 1990.
- [26] Hermann Ebbinghaus. *Memory: A contribution to experimental psychology*. University Microfilms, 1913.
- [27] Susan E Embretson and Steven P Reise. *Item response theory*. Psychology Press, 2013.
- [28] Jean-Claude Falmagne, Eric Cosyn, Jean-Paul Doignon, and Nicolas Thiéry. The assessment of knowledge, in theory and in practice. In *Formal concept analysis*, pages 61–79. Springer, 2006.
- [29] Thomas François and Cédric Fairon. An AI readability formula for French as a foreign language. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 466–477. Association for Computational Linguistics, 2012.
- [30] Glenn Fulcher. Text difficulty and accessibility: Reading formulae and expert judgement. *System*, 25(4):497–513, 1997.
- [31] Robert Godwin-Jones. Emerging technologies. *Language learning & technology*, 2010.
- [32] Itziar Gonzalez-Dios, María Jesús Aranzabe, Arantza Díaz de Ilarraza, and Haritz Salaberri. Simple or complex? assessing the readability of Basque texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 334–344, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.
- [33] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

- [34] Ronald K Hambleton, Hariharan Swaminathan, and H Jane Rogers. *Fundamentals of item response theory*, volume 2. Sage, 1991.
- [35] Julia Hancke, Sowmya Vajjala, and Detmar Meurers. Readability classification for German using lexical, syntactic, and morphological features. In *COLING*, pages 1063–1080, 2012.
- [36] Michael Heilman, Kevyn Collins-Thompson, and Maxine Eskenazi. An analysis of statistical models and features for reading difficulty prediction. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pages 71–79. Association for Computational Linguistics, 2008.
- [37] Cord Hockemeyer, Theo Held, and Dietrich Albert. Rath-a relational adaptive tutoring hypertext www-environment based on knowledge space theory, 1997.
- [38] Andreas Holzer, Christian Schallhart, Michael Tautschnig, and Helmut Veith. Fshell: Systematic test case generation for dynamic analysis and measurement. In *International Conference on Computer Aided Verification*, pages 209–213. Springer, 2008.
- [39] Ching-Kun Hsu, Gwo-Jen Hwang, and Chih-Kai Chang. Development of a reading material recommendation system based on a knowledge engineering approach. *Computers & Education*, 55(1):76–83, 2010.
- [40] Robin Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, pages 429–433. ACM, 2005.
- [41] Gwo-Jen Hwang, Han-Yu Sung, Chun-Ming Hung, and Iwen Huang. A learning style perspective to investigate the necessity of developing adaptive learning systems. *Educational Technology & Society*, 16(2):188–197, 2013.
- [42] Shoaib Jameel, Xiaojun Qian, and Wai Lam. *N*-gram fragment sequence based unsupervised domain-specific document readability. In *Proceedings of COLING 2012*, pages 1309–1326, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.
- [43] Japan Educational Exchanges, Services, and Japan Foundation. *The 2009-2 Japanese Language Proficiency Test Level 1 and 2: Questions and Correct Answers*. Bonjinsha Inc., 2010.
- [44] A-Yeong Kim, Hyun-Je Song, Seong-Bae Park, and Sang-Jo Lee. Device-

- dependent readability for improved text understanding. In *EMNLP*, pages 1396–1404, 2014.
- [45] Kenneth R Koedinger, John R Anderson, William H Hadley, and Mary A Mark. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education (IJAIED)*, 8:30–43, 1997.
- [46] Mathieu Koppen. Extracting human expertise for constructing knowledge spaces: An algorithm. *Journal of mathematical psychology*, 37(1):1–20, 1993.
- [47] Mathieu Koppen and Jean Paul Doignon. How to build a knowledge space by querying an expert. *Journal of Mathematical Psychology*, 34(3):311–331, July 1990.
- [48] Stephen D Krashen. *The input hypothesis: Issues and implications*. Addison-Wesley Longman Ltd, 1985.
- [49] Taku Kudo and Yuji Matsumoto. Japanese dependency analysis using cascaded chunking. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–7. Association for Computational Linguistics, 2002.
- [50] Annabel Latham, Keeley Crockett, and David McLean. An adaptation algorithm for an intelligent natural language tutoring system. *Computers & Education*, 71:97–110, 2014.
- [51] Nan Li, William W Cohen, and Kenneth R Koedinger. Problem order implications for learning transfer. In *Intelligent Tutoring Systems*, pages 185–194. Springer, 2012.
- [52] Wenzhao Liu and Hiroshi Ebihara. *New JLPT N1 Grammar Description*. East China University of Science and Technology Press, 2012.
- [53] Yun-En Liu, Christy Ballweber, Eleanor O’rourke, Eric Butler, Phonraphee Thummaphan, and Zoran Popović. Large-scale educational campaigns. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 22(2):8, 2015.
- [54] Derek Lomas, Kishan Patel, Jodi L Forlizzi, and Kenneth R Koedinger. Optimizing challenge in an educational game using large-scale design experiments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 89–98. ACM, 2013.

- [55] J Derek Lomas, Kenneth Koedinger, Nirmal Patel, Sharan Shodhan, Nikhil Poonwala, and Jodi L Forlizzi. Is difficulty overrated?: The effects of choice, novelty and suspense on intrinsic motivation in educational games. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 1028–1039. ACM, 2017.
- [56] Frederic M Lord. *Applications of item response theory to practical testing problems*. Routledge, 1980.
- [57] Nikos Manouselis, Hendrik Drachslar, Riina Vuorikari, Hans Hummel, and Rob Koper. Recommender systems in technology enhanced learning. *Recommender systems handbook*, pages 387–415, 2011.
- [58] Javier Valenzuela Manzanares and Ana María Rojo López. What can language learners tell us about constructions? *APPLICATIONS OF COGNITIVE LINGUISTICS*, 9:197, 2008.
- [59] Mnemosyne. *Welcome to the Mnemosyne Project*, 2019.
- [60] Tong Mu, Shuhan Wang, Erik Andersen, and Emma Brunskill. Combining adaptivity with progression ordering for intelligent tutoring systems. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, page 15. ACM, 2018.
- [61] Jessica Nelson, Charles Perfetti, David Liben, and Meredith Liben. Measures of text difficulty: Testing their predictive value for grade levels and student performance. *Council of Chief State School Officers, Washington, DC2012*, 2012.
- [62] Inc Newgrounds. *Newgrounds.com — Everything, By Everyone*, 2018. <https://www.newgrounds.com>.
- [63] NHK. *NEWS WEB EASY*, 2019. www3.nhk.or.jp/news/easy/.
- [64] Zachary Pardos and Neil Heffernan. Navigating the parameter space of Bayesian Knowledge Tracing models: Visualizations of the convergence of the Expectation Maximization algorithm. In *Educational Data Mining 2010*, 2010.
- [65] Zachary A Pardos and Neil T Heffernan. KT-IDEM: Introducing item difficulty to the knowledge tracing model. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 243–254. Springer, 2011.
- [66] Philip I Pavlik and John R Anderson. Practice and forgetting effects on vocabu-

- lary memory: An activation-based model of the spacing effect. *Cognitive Science*, 29(4):559–586, 2005.
- [67] Philip I Pavlik and John R Anderson. Using a model to compute the optimal schedule of practice. *Journal of Experimental Psychology: Applied*, 14(2):101, 2008.
- [68] People’s Education Press. *Standard Japanese of China-Japan Exchanges for Beginners*. Mitsumura Tosho Publishing Co.Ltd., 2013.
- [69] Ildikó Pilán, Elena Volodina, and Lars Borin. Candidate sentence selection for language learning exercises: from a comprehensive framework to an empirical evaluation. *CoRR*, abs/1706.03530, 2017.
- [70] Emily Pitler and Ani Nenkova. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the conference on empirical methods in natural language processing*, pages 186–195. Association for Computational Linguistics, 2008.
- [71] Georg Rasch. *Probabilistic models for some intelligence and attainment tests*. ERIC, 1993.
- [72] Mark Reckase. *Multidimensional item response theory*, volume 150. Springer, 2009.
- [73] Reddit. *Korean*, 2018. <https://www.reddit.com/r/Korean/>.
- [74] Reddit. *language learning*, 2018. <https://www.reddit.com/r/languagelearning/>.
- [75] Reddit. *Learn Japanese*, 2019. <https://www.reddit.com/r/LearnJapanese/>.
- [76] C Reigeluth and R Stein. Elaboration theory. *Instructional-design theories and models: An overview of their current status*, pages 335–381, 1983.
- [77] Charles M Reigeluth. *Instructional design theories and models: An overview of their current status*. Routledge, 2013.
- [78] Renaissance Learning. *ATOS analyzer*, 2016.

- [79] Jack C Richards and Richard W Schmidt. *Longman dictionary of language teaching and applied linguistics*. Routledge, 2013.
- [80] Hitoko Sasaki and Matsumoto Kiko. *Japanese Language Proficiency Test N1 GRAMMAR Summary*. World Publishing Corporation, 2010.
- [81] Sarah E Schwarm and Mari Ostendorf. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 523–530. Association for Computational Linguistics, 2005.
- [82] Kathleen M Sheehan, Irene Kostin, Diane Napolitano, and Michael Flor. The TextEvaluator tool: Helping teachers and test developers select texts for use in instruction and assessment. *The Elementary School Journal*, 115(2):184–209, 2014.
- [83] Manjira Sinha, Tirthankar Dasgupta, and Anupam Basu. Influence of target reader background and text features on text readability in Bangla: A computational approach. In *COLING*, pages 345–354, 2014.
- [84] Manjira Sinha, Sakshi Sharma, Tirthankar Dasgupta, and Anupam Basu. New readability measures for Bangla and Hindi texts. In *Proceedings of COLING 2012: Posters*, pages 1141–1150, Mumbai, India, December 2012. The COLING 2012 Organizing Committee.
- [85] Malbert Smith III, Anne Schiano, and Elizabeth Lattanzio. Beyond the classroom. *Knowledge Quest*, 42(3):20, 2014.
- [86] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.
- [87] Katharina Spiel, Sven Bertel, and Fares Kayali. "Not Another Z Piece!": Adaptive Difficulty in TETRIS. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 5126–5131, New York, NY, USA, 2017. ACM.
- [88] Mare Taagepera and S Noori. Mapping students' thinking patterns in learning organic chemistry by the use of knowledge space theory. *J. Chem. Educ.*, 77(9):1224, 2000.
- [89] Nikolai Tillmann and Jonathan De Halleux. Pex–white box test generation for.

- net. In *International conference on tests and proofs*, pages 134–153. Springer, 2008.
- [90] Miyamoto Atsushi Tomomatsu Etsuko and Waguri Masako. *Essential Japanese Expression Dictionary: A Guide to Correct Usage of Key Sentence Patterns (New Edition)*. ALC Press, 2010.
- [91] Sowmya Vajjala and Detmar Meurers. On improving the accuracy of readability classification using insights from second language acquisition. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 163–173. Association for Computational Linguistics, 2012.
- [92] Wim J van der Linden and Cees AW Glas. *Computerized adaptive testing: Theory and practice*. Springer, 2000.
- [93] Vox. How the inventor of mario designs a game. <https://www.youtube.com/watch?v=K-NBcP0YUQI>, January 2017.
- [94] Lev Semenovich Vygotsky. *Mind in society: The development of higher psychological processes*. Harvard university press, 1980.
- [95] Renee Waara. Construal, convention, and constructions in L2 speech. *Cognitive linguistics, second language acquisition and foreign language pedagogy*, pages 51–75, 2004.
- [96] Howard Wainer and Robert J Mislevy. Item response theory, item calibration, and proficiency estimation. *Computerized adaptive testing: A primer*, pages 65–102, 1990.
- [97] Shuhan Wang and Erik Andersen. Grammatical templates: Improving text difficulty evaluation for language learners. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics*, pages 1692–1702, December 2016.
- [98] Shuhan Wang, Brandon Cohen, Sixian Yi, Jung Yun Park, Teo Nicholas, and Erik Andersen. Goal-based progression synthesis in a Korean learning game. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*, 2019.
- [99] Shuhan Wang, Fang He, and Erik Andersen. A unified framework for knowledge assessment and progression analysis and design. In *Proceedings of the 2017 CHI*

- Conference on Human Factors in Computing Systems*, pages 937–948. ACM, 2017.
- [100] Shuhan Wang, Hao Wu, Ji Hun Kim, and Erik Andersen. Adaptive learning material recommendation in online language education. In *In The 20th International Conference on Artificial Intelligence in Education (AIED), 2019*, 2019.
- [101] David J Weiss and G Kingsbury. Application of computerized adaptive testing to educational problems. *Journal of Educational Measurement*, 21(4):361–375, 1984.
- [102] Wikipedia. *Depth-first search*, 2018. https://en.wikipedia.org/wiki/Depth-first_search.
- [103] Wikipedia. *Rooted graph*, 2018. https://en.wikipedia.org/wiki/Rooted_graph.
- [104] Piotr Wozniak. Optimization of learning. Master’s thesis, University of Technology in Poznan, 1990.
- [105] Ruth Wylie, Kenneth Koedinger, and Teruko Mitamura. Analogies, explanations, and practice: examining how task types affect second language grammar learning. In *International Conference on Intelligent Tutoring Systems*, pages 214–223. Springer, 2010.
- [106] Menglin Xia, Ekaterina Kochmar, and Ted Briscoe. Text readability assessment for second language learners. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 12–22. Association for Computational Linguistics, 2016.
- [107] Xiaoming Xu and Reika. *Blue Book All-in-one: JLPT N1-N5 Grammar*. East China University of Science and Technology Press, 2015.
- [108] Yanbo Xu and Jack Mostow. Using logistic regression to trace multiple sub-skills in a dynamic bayes net. In *Educational Data Mining 2011*, 2010.
- [109] Lihua Yao. Multidimensional CAT item selection methods for domain scores and composite scores: Theory and applications. *Psychometrika*, 77(3):495–523, 2012.
- [110] Mostafa Zamanian and Pooneh Heydari. Readability of texts:: State of the art. *Theory and Practice in Language Studies*, 2(1):43, 2012.

- [111] Language Zen. *The Fastest Way to Learn Spanish*, 2016. www.languagezen.com, as of July 2016.
- [112] Helen Zhao, Kenneth Koedinger, and John Kowalski. Knowledge tracing and cue contrast: Second language English grammar instruction. In *Proceedings of the 35th Annual Conference of the Cognitive Science Society*, 2013.