The Pennsylvania State University
The Graduate School
College of Information Sciences and Technology

MACHINE LEARNING METHODS FOR BUILDING EDUCATIONAL APPLICATIONS: CONCEPT PREREQUISITE LEARNING AND AUTOMATIC DISTRACTOR GENERATION

A Dissertation in
Information Sciences and Technology
by
Chen Liang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2018

The dissertation of Chen Liang was reviewed and approved* by the following:

*Signatures are on file in the Graduate School.

Abstract

The increasing amount of education-related data provides a valuable research opportunity for developing data-driven machine learning methods for building educational applications. This dissertation investigates machine learning solutions for two educational applications: concept prerequisite learning and automatic distractor generation.

A prerequisite relation describes a fundamental directed relation among concepts in knowledge structures. The first part of this dissertation focuses on the concept prerequisite learning problem, the study of machine learning methods for automatic concept prerequisite discovery. Specifically, this dissertation explores the use of Wikipedia – the largest free online encyclopedia – for concept prerequisite learning and presents the following studies towards automatically measuring concept prerequisite relations. First, a simple but effective link-based feature, RefD, is proposed for measuring prerequisite relations among concepts. Second, how concept prerequisite relations can be recovered from university course dependencies is explored. Third, active learning of concept prerequisite learning is studied to deal with the lack of large-scale concept prerequisite labels. The dissertation explores the mathematical nature of prerequisite relation being a strict partial order and proposes an active learning framework tailored for such relation. The proposed approach incorporates relational reasoning not only in finding new unlabeled pairs whose labels can be deduced from an existing label set, but also in devising new query strategies that consider the relational structure of labels.

Multiple choice questions (MCQs) are widely used to assess students' knowledge and skills. Among all methods for creating good MCQs, finding reasonable distractors is crucial and usually the most time-consuming. The second part of this dissertation investigates automatic distractor generation (DG). In contrast with previous similarity-based methods, this dissertation presents two studies on machine learning methods for DG. The first study proposes a generative model learned from training generative adversarial nets to create useful distractors for automatically creating fill-in-the-blank questions. The second work investigates how ranking models can be used to select useful distractors for MCQs. The proposed models can learn to select distractors that resemble those in actual exam questions.

Finally, the dissertation introduces BBookX, a computer-facilitated book-creation system. Using information retrieval techniques, BBookX is designed to facilitate the online book-creation process by searching OERs. BBookX is an actual educational application where the proposed methods for concept prerequisite learning and automatic distractor generation could be applied.

Table of Contents

List of Figures **x**

List of Tables **xii**

Acknowledgments **xiv**

Chapter 1
Introduction **1**

1.1 Background 1
1.2 Concept Prerequisite Learning 2
1.2.1 RefD: A Link-based Feature for Measuring Concept Prerequisite Relations 4
1.2.2 Learning Concept Prerequisites from University Course Dependencies 4
1.2.3 Active Learning for Concept Prerequisite Learning 4
1.3 Automatic Distractor Generation for Multiple Choice Questions . . 5
1.3.1 Distractor Generation with Generative Adversarial Nets . . 6
1.3.2 Learning to Rank for Distractor Generation 6
1.4 Summary of Research Contributions 7
1.5 Structure of the Dissertation 7

Chapter 2
RefD: a Link-based Feature for Measuring Prerequisite Relations Among Concepts **9**

2.1 Introduction 9
2.2 Measuring Prerequisite Relations 10
2.3 Wikipedia-based RefD Implementation 12
2.4 Experiments 12
2.4.1 Results on the CrowdComp Dataset 13
2.4.2 Results on the Course Dataset 14

2.4.3 Parameter Analysis and Case Study . . . 16
2.5 Related Work . . . 17
2.6 Summary and Discussion . . . 19

Chapter 3
Recovering Concept Prerequisite Relations from University Course Dependencies 20
3.1 Introduction . . . 20
3.2 Problem Setup . . . 22
3.3 Our Approach . . . 23
3.3.1 Course Representation . . . 23
3.3.2 CPR-Recover Formulation . . . 23
3.4 Experiments . . . 25
3.4.1 Synthetic Dataset . . . 25
3.4.1.1 Process of Data Generation . . . 26
3.4.1.2 Baselines and Evaluation Metrics – Synthetic Data 27
3.4.1.3 Results – Synthetic Data . . . 28
3.4.2 University Course Dataset . . . 29
3.4.2.1 Data Labeling . . . 29
3.4.2.2 Baselines and Evaluation Metrics – University Course Data . . . 30
3.4.2.3 Results – University Course Data . . . 31
3.4.2.4 Case Study . . . 32
3.5 Related Work . . . 33
3.6 Summary . . . 34

Chapter 4
Investigating Active Learning for Concept Prerequisite Learning 35
4.1 Introduction . . . 35
4.2 Pool-based Active Learning . . . 36
4.2.1 Query Strategies . . . 36
4.3 Experimental Design . . . 38
4.3.1 Dataset . . . 38
4.3.2 Feature Description . . . 39
4.3.2.1 Graph-based Features (GF) . . . 39
4.3.2.2 Text-based Features (TF) . . . 40
4.4 Experimental Results . . . 42
4.4.1 Evaluation of Classification . . . 42
4.4.2 Feature Analysis . . . 43
4.4.3 Evaluation of Active Learning . . . 44

4.4.3.1 Settings 44
4.4.3.2 Results 45
4.5 Summary and Discussion 46

Chapter 5
Active Learning of Strict Partial Orders: A Case Study on Concept Prerequisite Relations 48
5.1 Introduction 48
5.2 Reasoning of a Strict Order 50
5.2.1 Preliminary 50
5.2.2 Reasoning Module for Closure Calculation 51
5.2.3 Computational Efficiency 52
5.3 Active Learning of a Strict Order 54
5.3.1 Basic Relational Reasoning in Active Learning 54
5.3.2 Query Strategies with Relational Reasoning 56
5.3.3 Bounds on the Number of Queries 57
5.4 Experiments 57
5.4.1 Experiment Results 58
5.4.1.1 Effectiveness Study 58
5.4.1.2 Efficiency Study 59
5.5 Related Work 63
5.6 Summary and Discussion 64

Chapter 6
Automatic Distractor Generation for Multiple Choice Questions 66
6.1 Introduction 66
6.2 Related Work 67
6.2.1 Automatic Question Generation 67
6.2.2 Distractor Generation 68
6.3 Distractor Generation with Generative Adversarial Nets for Automatically Creating Fill-in-the-blank Questions 68
6.3.1 Introduction 68
6.3.2 Methods 69
6.3.2.1 Generative Adversarial Nets 69
6.3.2.2 Conditional GANs for DG 70
6.3.2.3 Context Modeling of the Question Sentence 72
6.3.2.4 Implementation Details 73
6.3.3 Experiments 73
6.3.3.1 Data Preparation 73
6.3.3.2 Experiment Settings 74

6.3.3.3 Experimental Results 75
6.3.4 Summary 76
6.4 Distractor Generation for Multiple Choice Questions Using Learning to Rank 77
6.4.1 Introduction 77
6.4.2 Learning to Rank for Distractor Generation 77
6.4.2.1 Feature-based Models 78
6.4.2.1.1 Feature Description 78
6.4.2.1.2 Classifiers 79
6.4.2.2 NN-based Models 79
6.4.2.3 Cascaded Learning Framework 81
6.4.3 Experiments 82
6.4.3.1 Datasets 82
6.4.3.2 Experiment Settings 82
6.4.3.3 Training and Implementation Details 83
6.4.3.3.1 Feature-based Models. 83
6.4.3.3.2 NN-based Models. 84
6.4.3.4 Experimental Results 84
6.4.3.4.1 First Stage Ranker 84
6.4.3.4.2 Distractor Ranking Results 85
6.4.3.4.3 Feature Analysis 86
6.4.3.4.4 Effects of Cascaded Learning 86
6.4.4 Summary 87
6.5 Discussion 87

Chapter 7
BBookX: a Computer-facilitated Book Creation Tool 88
7.1 Introduction 88
7.2 System Overview 89
7.3 Open Book Repository 89
7.3.1 Collecting Open Access Textbooks 89
7.3.2 Generating Open Books Using Wikipedia 90
7.3.3 Indexing Subsystem 92
7.4 Interactive Book Creation 93
7.4.1 User Interface 93
7.4.2 Query Subsystem 94
7.5 Field Test 96
7.6 Book Publishing 97
7.7 Related Work 97
7.8 Summary and Discussion 99

Chapter 8
Conclusion and Future Work **100**
8.1 Conclusion . . . 100
8.2 Future Work . . . 102

Appendix
Supplemental Material for Chapter 5 **104**
1 Missing Proofs . . . 104
1.1 Proof of Proposition 1 . . . 104
1.2 Proof of Proposition 2 . . . 104
1.3 Proof of Theorem 1 . . . 104
1.3.1 Well-definiteness . . . 104
1.3.2 Necessity . . . 104
1.3.3 Sufficiency . . . 105
1.4 Proof of Proposition 3 . . . 106
1.5 Proof of Proposition 4 . . . 107
1.6 Proof of Theorem 2 . . . 107
2 Experiment Environment . . . 108

Bibliography **109**

List of Figures

1.1 Concept prerequisite relations. "$A \rightarrow B$" represents that the concept A is a prerequisite of the concept B. 3

1.2 Examples of multiple choice questions. The correct answers are written in bold text. Note the second MCQ is also a fill-in-the-blank question. 6

2.1 An example of reference relations of two concepts, *Data mining* (A) and *Algorithm* (B), with a prerequisite relation. An arrow represents a reference relation. The dashed line separates two main concepts. . 11

2.2 Comparison of Precision-Recall curves of MaxEnt and *RefD* (using EQUAL and TFIDF weighting) on the Course dataset. 15

2.3 Average accuracy on two datasets with a given threshold of *RefD* using TFIDF weighting. 16

3.1 Recovering course prerequisite relations from course dependencies. . 21

3.2 Process of synthetic data generation. After a concept prerequisite graph is created, prerequisite course pairs are generated by blending prerequisite concept pairs with background words. 26

3.3 Results on two synthetic datasets. 28

3.4 Results using different numbers of pairs (k) of course dependencies. $*$ indicates that with a given k our method significantly outperforms other two methods ($p < 0.05$). 31

4.1 Comparison of different query strategies for concept prerequisite classification. 45

5.1 Following the notations in Theorem 1: (a) Black lines are pairs in H, solid lines are pairs in G, and dashed lines are pairs in G^c. The pair (a, b) in the cyan color is the pair to be labeled or deduced. (b) If $(a, b) \in G$, $\{(a, b), (e, f), (a, f), (e, b)\} \subseteq N_{(a,b)}$. (c) If $(a, b) \in G$, $\{(h, e), (h, a)\} \subseteq T_{(a,b)}$ and $\{(b, g), (f, g)\} \subseteq S_{(a,b)}$. (d) If $(a, b) \in G^c$, $\{(a, b), (a, d), (c, b), (c, d)\} \subseteq N'_{(a,b)}$. Likewise, if $\exists (x, y) \in G, \text{s.t.} (a, b) \in S_{(x,y)} \cup T_{(x,y)}$, $\{(a, b), (a, d), (c, b)\} \subseteq O_{(x,y)}$. 53
5.2 Comparison of different query strategies' AUC scores for concept prerequisite learning. 60
5.2 Comparison of different query strategies' AUC scores for concept prerequisite learning. (cont.) 61
5.3 Comparison of relations between the number of queries and the number of labeled instances when using different query strategies. . 61
5.3 Comparison of relations between the number of queries and the number of labeled instances when using different query strategies. (cont.) 62
5.4 The average runtime for calculating the new closure using Theorem 1 v.s. the size of current labeled closure $|H|$. 63

6.1 Conditional GANs for Distractor Generation. 71
6.2 Context Modeling of the Question Sentence 72
6.3 Proposed adversarial training framework for DG. 80
6.4 Recall@K for the first stage ranker. 84

7.1 BBookX system overview. 90
7.2 Example of a Computer Networking book created using Wikipedia. The left part is the table of contents of the original book. The right part is the generated table of contents. 92
7.3 Interactive User interface. 94
7.3 Interactive User interface. (cont.) 95
7.4 Example of a completed book, when embedded on a website. The first chapter is expanded, to view the links to each section. 98

List of Tables

2.1 Statistics of CrowdComp and Course Datasets 13
2.2 Comparison of out-of-domain training accuracies of a MaxEnt classifier and *RefD* using EQUAL and TFIDF weighting. MaxEnt† is the number reported by Talukdar et al. [151]. MaxEnt shows the performance of our implementation. * indicates the difference between *RefD* and MaxEnt is statistically significant ($p < 0.01$). . . 14
2.3 Comparison of in-domain training accuracies, precision, recall, and F1 measure of MaxEnt and *RefD* using EQUAL and TFIDF weighting. * indicates the improvement over MaxEnt is statistically significant ($p < 0.01$). 14
2.4 *RefD* scores between "deep learning" and the concepts linked from it. All scores are calculated by *RefD*('deep learning', concept). . . . 16

3.1 Statistics of two synthetic datasets. 27
3.2 Results on the course dataset. 31
3.3 Examples of prerequisite concept pairs we recovered, which are categorized as true prerequisite pairs (TPP) and false prerequisite pairs (FPP). Note c_s is expected to be a prerequisite of c_t. 33

4.1 Dataset statistics. 39
4.2 Results (%) for concept prerequisite relation classification. 42
4.3 Top 10 important features for each domain. 43

5.1 Summary of compared query strategies. 58
5.2 The effect of Proposition 4 on the total runtime (s) of $O_{(a,b)}$ calculation in Theorem 1 for a full round of active learning (until $\mathcal{D}_u = \emptyset$). 64

6.1 Distractor generation results. Numbers are 95% confidence intervals of percentages of generated distractors being good, fair, or bad, calculated in a leave-one-out manner. 75

6.2 Distractor generation examples for question "Changes in gene frequency over time describes the process of _____.", whose key is *Evolution*. (Legend: Good, Fair, Bad) 75
6.3 Dataset Statistics. 81
6.4 Ranking results (%) for DG on SciQ dataset. 85
6.5 Ranking results (%) for DG on MCQL dataset. 85
6.6 Top 10 important features for learning to rank distractors. 86

7.1 Performance of the candidate ranking on three different textbooks. . 91

Acknowledgments

First and foremost, I thank my advisor, Prof. C. Lee Giles for his guidance during my years at Penn State. This dissertation would not have been possible without his invaluable advice, encouragement, and motivation. I will be eternally grateful for the knowledge of research and life that he shared with me. It was an honor being a member of his lab.

I also thank Prof. Vasant G. Honavar, Prof. John Yen, Prof. Barton K. Pursel, and Prof. Rebecca J. Passonneau for their valuable help and insightful advice as members of my dissertation committee.

During my time at Penn State, I have been extremely fortunate to work with excellent collaborators and lab mates, and I am thankful for each one: Zhaohui Wu, Jianbo Ye, Wenyi Huang, Shuting Wang, Xiao Yang, Dafang He, Jian Wu, Drew Wham, Kyle Williams, Neisarg Dave, Sagnik Ray Choudhury, Rabah Al-Zaidy, Alexander G. Ororbia II, Agnese Chiatti, Kunho Kim, Lu Liu, Wenbo Guo, and Athar Sefid. It has been my pleasure working with these talented and determined colleagues.

I have made many wonderful friends at Penn State. While there are too many to mention here, but I owe special appreciation to Fei Wu, Hongjian Wang, Junpeng Qiu, Mo Yu, Yu Luo, Yang Xu, Pei Wang, Jian-Syuan Wong, Jun Xu, Cong Liao, Feng Sun, and Haoti Zhong. I will always cherish the days we shared together at the lovely small town of State College.

Finally, I extend my sincere gratitude to my parents, Hongyun Liang and Hua Yuan. Thank you for the constant love, support, and guidance that helped me through highs and lows in my life.

Dedication

I humbly dedicate this work to my parents, whose constant support and love have made it possible.

Chapter 1 Introduction

1.1 Background

There have been many computer-based learning systems from which a large amount of education-related usage data can be gathered. Examples include Learning Management Systems, Intelligent Tutoring Systems (ITS) [32], and recently developed Massive Open Online Courses (MOOCs). Their usage data provides a valuable opportunity for developing data-driven machine learning methods for building educational applications, which has gained increasing attention from computer science subfields data mining and natural language processing (NLP).

In the field of educational data mining (EDM), researchers have investigated learning methods for applications such as (i) student modeling [154], which includes performance prediction [126], engagement modeling [136], undesirable student behavior detection [57, 104], student profiling and grouping [58], etc., (ii) planning and scheduling [66], and (iii) automatic concept map construction [83]. With a special focus on utilizing text information, the NLP community has been working on educational applications including automated written response scoring/evaluation [135], tools for second and foreign language learners [153], automatic test question generation [118], educational dialog systems [98], automatic grammatical error correction [120], plagiarism detection [43], etc. The use of machine learning for building educational applications is still at an early stage, compared to the extensive research on general machine learning. Note the major venues for such studies have a relatively short history, with about 10 years for the EDM conference[1]

[1]The International Conference on Educational Data Mining

and 15 years for the NLP BEA workshop[2].

This dissertation research seeks to develop machine learning methods for two educational applications: concept prerequisite learning and automatic distractor generation. Before introducing the two tasks, we begin by defining the following general terms that will be frequently used throughout this dissertation.

Definition 1 (Concept)**.** *A concept is defined as a general idea of something. To be concrete, each concept in this dissertation corresponds to a Wikipedia entity/title. For example, "Machine learning", "Data mining", "Natural language processing" are all concepts.*

Definition 2 (Prerequisite of a Concept)**.** *A prerequisite of a concept C is a concept that is necessary to learn before one can proceed to understand C. For example, "Linear algebra" is a prerequisite of "Deep learning".*

Definition 3 (Distractor)**.** *A distractor is an alternative answer used to sidetrack students from the correct answer. Distractors are usually a part of multiple choice questions. For example, "ADP" could be a distractor for the question, "A compound which is found in all living cells and play a key role in energy transformations is ____," of which "ATP" is the correct answer.*

1.2 Concept Prerequisite Learning

A *prerequisite* relation describes a fundamental directed connection among concepts in knowledge structures. Following the learning order that is consistent with the underlying prerequisite relationship is crucial to successful and effective teaching and learning processes. For the example shown in Figure 1.1, learning the concept "Hidden Markov Model" requires first understanding prerequisites such as "posterior probability" and "maximum likelihood". Identifying prerequisite concepts is crucial for a variety of other educational applications such as curriculum planning [5] and intelligent tutoring systems [7]. It can be especially useful for online learning where students face a large amount of educational resources. For example, prerequisite information can be extremely helpful for students in MOOCs, who are typically faced with hundreds of course choices. Because each university creates its own

[2]The Workshop on Innovative Use of NLP for Building Educational Applications

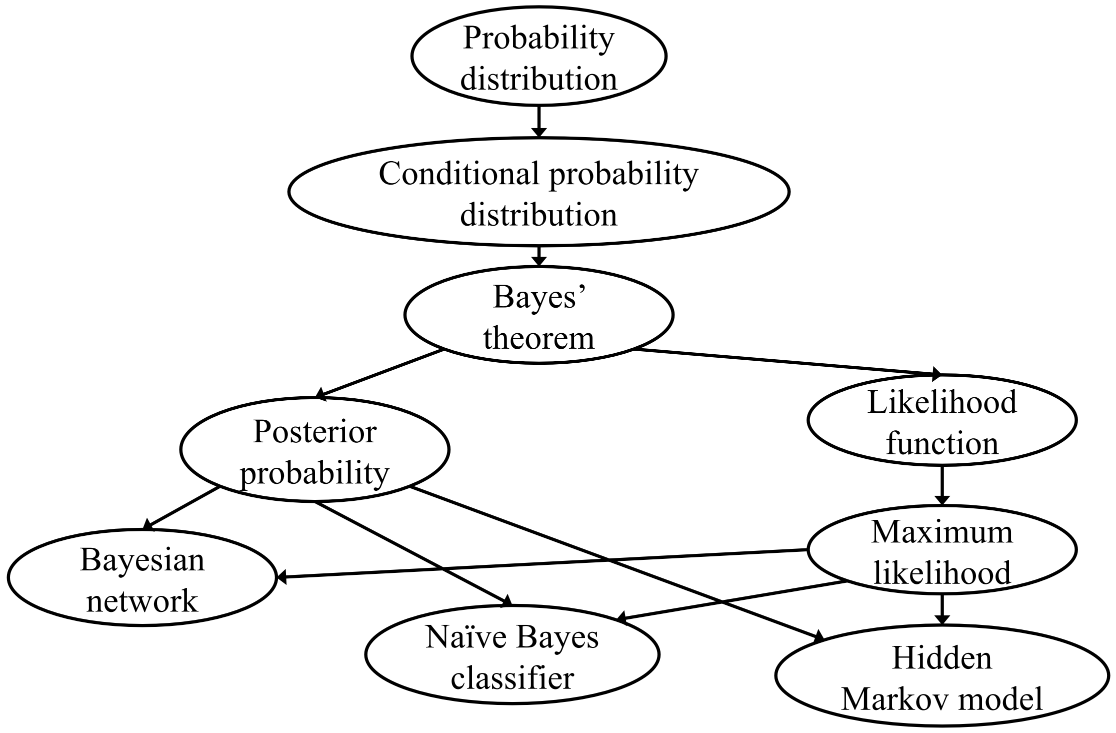


Figure 1.1. Concept prerequisite relations. "$A \to B$" represents that the concept A is a prerequisite of the concept B.

MOOCs and puts them on different MOOC platforms, there are usually no readily identifiable prerequisite relations among courses from different universities or across different platforms. In addition, manually organizing prerequisites from thousands of MOOCs would be too time-consuming. This challenge motivates the need for automatic prerequisite relation discovery methods.

The first part of this dissertation focuses on solving the *concept prerequisite learning problem* [151], the study of machine learning methods for automatic concept prerequisite discovery. Specifically, the dissertation focused on the concept prerequisite learning problem defined as follows:

Definition 4 (Concept Prerequisite Learning Problem.)**.** *Given a pair of concepts (A, B), predict whether A is a prerequisite of B.*

Concept prerequisite learning is is a binary classification problem. Here, cases where B is a prerequisite of A and where no prerequisite relation exists are both considered negative.

A possible solution for developing scalable methods for automatic prerequisite discovery is to develop or integrate approaches that automatically infer such prerequisites from the increasing amount of digital educational data. Available data sources include knowledge bases, student assessment data, text books, course materials, etc. This dissertation explores the use of Wikipedia – the largest free online encyclopedia – for concept prerequisite learning and presents three studies towards automatically measuring concept prerequisite relations, which are summarized as follows.

1.2.1 RefD: A Link-based Feature for Measuring Concept Prerequisite Relations

As a semantic relation, concept prerequisite relation has not been well studied in computational linguistics. The dissertation proposes a simple link-based feature, namely *reference distance* (**RefD**) [90], that effectively models the relation by measuring how differently two concepts refer to each other. Evaluations on two datasets that include seven domains show that a Wikipedia-based RefD implementation outperforms existing supervised learning-based methods.

1.2.2 Learning Concept Prerequisites from University Course Dependencies

Besides using knowledge bases such as Wikipedia, this dissertation also investigates how to recover concept prerequisite relations from course dependencies [94]. An optimization-based framework is proposed to address the problem. The first real dataset for empirically studying this problem is created, which consists of the listings of computer science courses from 11 U.S. universities and their concept pairs with prerequisite labels. Experiment results on a synthetic dataset and the real course dataset both show that the proposed method outperforms existing baselines.

1.2.3 Active Learning for Concept Prerequisite Learning

A major obstacle to extracting concept prerequisite relations at scale is the lack of large-scale labels to enable effective data-driven solutions. This dissertation presents the first study [93] to investigate the applicability of active learning to concept prerequisite learning. We propose a novel set of features tailored for prerequisite classification and compare the effectiveness of four widely used query strategies. Experimental results for domains including data mining, geometry, physics, and precalculus show that active learning can be used to reduce the amount of training data required. Given the proposed features, the query-by-committee strategy outperforms other compared query strategies.

Mathematically, a prerequisite relation is a type of strict partial order, a mathematical structure commonly seen in relational data. As a follow-up work for the abovementioned study, this dissertation proposes an active learning framework [95] for mining such relations subject to a strict order. The proposed approach incorporates relational reasoning not only in finding new unlabeled pairs whose labels can be deduced from an existing label set, but also in devising new query strategies that consider the relational structure of labels. Experiments on concept prerequisite relations show the proposed framework can substantially improve the classification performance with the same query budget compared to other baseline approaches.

1.3 Automatic Distractor Generation for Multiple Choice Questions

Multiple choice questions (MCQs) are widely used as an assessment of students' knowledge and skills. An MCQ consists of three elements: (i) *stem*, the question sentence; (ii) *key*, the correct answer; and (iii) *distractors*, alternative answers used to sidetrack students from the correct answer. See Figure 1.2 for examples of MCQs. Among all methods for creating good MCQs, finding reasonable distractors is crucial and usually the most time-consuming. The second part of this dissertation investigates automatic *distractor generation* (DG), i.e., generating distractors given the stem and the key to the question. DG is a crucial step for multiple choice question generation (MCQG) because one of its main challenges is the generation of "good" distractors which can distinguish knowledgeable test takers from less knowledgeable ones, in the sense that the question becomes more effective at testing a student's knowledge.

Most existing methods for DG are based on semantic similarities [2, 54, 80]. Distractors are selected from a ranked list based on a weighted combination of

1. What is the least dangerous radioactive decay?

   (a) Beta decay **(b) Alpha decay** (c) Zeta decay (d) Gamma decay

2. If extension in spring is proportional to load applied then material obeys _____.

   (a) Gravitational law (b) Newton's law **(c) Hooke's law** (d) Charles's law

Figure 1.2. Examples of multiple choice questions. The correct answers are written in bold text. Note the second MCQ is also a fill-in-the-blank question.

different similarity metrics, where the weights are determined by heuristics. In contrast with previous similarity-based methods, this dissertation presents two studies on machine learning methods for DG, which are summarized below.

1.3.1 Distractor Generation with Generative Adversarial Nets

We propose a generative model learned from training generative adversarial nets (GANs) to create useful distractors for automatically creating fill-in-the-blank questions [92]. Our method utilizes only context information and does not use the correct answer, which is completely different from previous ontology-based or similarity-based approaches. Trained on the Wikipedia corpus, the proposed model can predict Wiki entities as distractors. Our method is evaluated on two biology question datasets collected from Wikipedia and actual college-level exams. Experimental results show that our context-based method achieves comparable performance to a frequently used word2vec-based method for the Wiki dataset. In addition, we propose a second-stage learner to combine the strengths of the two methods, which further improves the performance on both datasets, with 51.7% and 48.4% of generated distractors being acceptable.

1.3.2 Learning to Rank for Distractor Generation

We investigate how machine learning models, specifically ranking models, can be used to select useful distractors for MCQs [91]. Our proposed models can learn to select distractors that resemble those in actual exam questions, which is different from most existing unsupervised ontology-based and similarity-based methods. We empirically study feature-based and neural net based (NN-based) ranking models with experiments on the recently released SciQ dataset and our MCQL dataset. Experimental results show that feature-based ensemble learning methods (random forest and LambdaMART) outperform both the NN-based method and unsupervised baselines. These two datasets can serve as benchmarks for distractor generation.

1.4 Summary of Research Contributions

To summarize, the goal of this dissertation is to design machine learning methods tailored for two educational applications: concept prerequisite learning and automatic distractor generation for multiple choice questions. The main research contributions include:

- A simple but effective link-based feature for measuring concept prerequisite relations. [90]
- A novel optimization-based method to learn concept-level prerequisite relations from course dependencies. [94]
- The first study of active learning for the concept prerequisite learning problem. [93]
- The first attempt to design active learning query strategies tailored for strict partial orders. The proposed methods are applied to concept prerequisite learning and appear to be successful on data from educational domains. [95]
- The first application of GANs to automatic distractor generation. [92]
- Supervised learning to rank methods for automatic distractor generation. [91]

1.5 Structure of the Dissertation

This dissertation presents the research described above. Specifically, Chapter 2 introduces a simple but effective link-based feature, RefD, for measuring concept prerequisite relation, and Chapter 3 proposes an optimization framework to learn concept prerequisites from university course dependencies. Chapter 4, then, investigates active learning for concept prerequisite learning. Chapter 5 explores the mathematical nature of prerequisite relation being a strict partial order and proposes an active learning framework tailored for such relation. The focus shifts to distractor generation in Chapter 6, which presents two studies on supervised learning methods for automatic distractor generation, and Chapter 7, which introduces BBookX, an educational application where the proposed methods for distractor generation and concept prerequisite learning would potentially be useful. Lastly, Chapter 8 concludes the dissertation and discusses possible future work.

Chapter 2 RefD: a Link-based Feature for Measuring Prerequisite Relations Among Concepts

2.1 Introduction

What should one know/learn before starting to learn a new area such as "deep learning"? A key for answering this question is to understand what a *prerequisite* is. A prerequisite is usually a concept or requirement before one can proceed to a following one. And the prerequisite relation exists as a natural dependency among concepts in cognitive processes when people learn, organize, apply, and generate knowledge [82]. While there has been serious effort in understanding prerequisite relations in learning and education [12, 123, 156], it has not been well studied as a semantic relation in computational linguistics, where researchers focus more on lexical relations among lexical items [113] and fine-grained entity relations in knowledge bases [116].

Instead of treating it as a relation extraction or link prediction problem using traditional machine learning approaches [151, 169], we seek to better understand prerequisite relations from a perspective of cognitive semantics [34]. Partially motivated by that to understand a concept, one needs to understand all the related concepts, we propose a metric that measures prerequisite relations based on a simple observation of human learning. When learning concept A, if one needs to refer to concept B for a lot of A's related concepts but not vice versa, B would more likely be a prerequisite of A than A of B. Specifically, we model a concept in a vector space using its related concepts and measure the prerequisite relation between two concepts by computing how differently the two's related concepts refer to each other, or *reference distance* (**RefD**).

Our simple metric **RefD** successfully reflects some properties of the prerequisite relation such as asymmetry and irreflexivity; and can be properly implemented for various applications using different concept models. We present an implementation of the metric using Wikipedia by leveraging the links as reference relations among concepts; and present a scalable prerequisite dataset construction method by crawling public available university course prerequisite websites and mapping them to Wikipedia concepts. Experimental results on two datasets that include seven domains demonstrate its effectiveness and robustness on measuring prerequisites. Surprisingly, our single metric based approach significantly outperforms baselines which use more sophisticated supervised learning. All the datasets are publicly available upon request.

Our main contributions include:

- A novel metric to measure the prerequisite relation among concepts that outperforms existing supervised learning baselines.
- A new dataset containing 1336 concept pairs in Computer Science and Math.

2.2 Measuring Prerequisite Relations

Our goal is to design a function $f : \mathcal{C}^2 \rightarrow \mathcal{R}$ that maps a concept pair (A, B) to a real value that measures the extent to which A requires B as a prerequisite, where $\mathcal{C}$ is the concept space. How should a concept be represented in $\mathcal{C}$? Since one cannot understand a concept without access to all essential knowledge related to it, such knowledge can be actually viewed as a set of related concepts. Thus, a concept could be represented by its related concepts in $\mathcal{C}$. For example, the concept "deep learning" may be represented by concepts such as "machine learning", "artificial neural network", etc.

Compared to prerequisites, a more common and observable relation among concepts is a *reference*, which widely exists in various forms such as hyperlinks, citations, notes, etc. Although a single evidence of reference does not indicate a

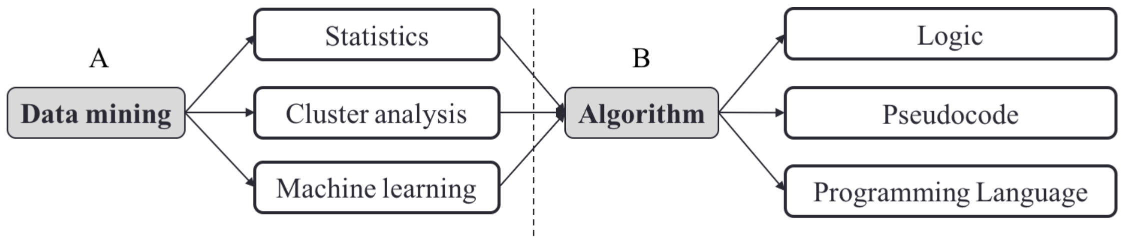


Figure 2.1. An example of reference relations of two concepts, *Data mining* (A) and *Algorithm* (B), with a prerequisite relation. An arrow represents a reference relation. The dashed line separates two main concepts.

prerequisite relation, a large number of such evidences might make a difference. For example, if most related concepts of A refer to B but few related concepts of B refer to A, then B is more likely to be a prerequisite of A, as shown in Figure 2.1. In order to measure prerequisite relations, we propose a *reference distance* (**RefD**), which is defined as

$$RefD(A,B) = \frac{\sum_{i=1}^{k} r(c_i,B)\cdot w(c_i,A)}{\sum_{i=1}^{k} w(c_i,A)} - \frac{\sum_{i=1}^{k} r(c_i,A)\cdot w(c_i,B)}{\sum_{i=1}^{k} w(c_i,B)} \tag{2.1}$$

where $\mathcal{C} = \{c_1, ..., c_k\}$ is the concept space; $w(c_i, A)$ weights the importance of c_i to A; and $r(c_i, A)$ is an indicator showing whether c_i refers to A, which could be links in Wikipedia, mentions in books, citations in papers, etc.

$RefD$ enables several useful properties for the prerequisite relation: 1) normalized: $RefD(A,B) \in [-1,1]$; 2) asymmetric: $RefD(A,B)$+$RefD(B,A)$=0, which means if A is a prerequisite of B then B is not a prerequisite of A; and 3) irreflexive: $RefD(A,A)$=0, which means A is not a prerequisite of itself. To capture all three possible prerequisite relations between a concept pair, *RefD* is expected to satisfy the following constraints:

$$RefD(A,B) \in \begin{cases} (\theta, 1], & \text{if B is a prerequisite of A} \\ [-\theta, \theta], & \text{if no prerequisite relation} \\ [-1, -\theta), & \text{if A is a prerequisite of B} \end{cases}$$

where θ is a positive threshold.

Equation 2.1 provides a general framework to calculate *RefD*. In practice, we need to specify the concept space $\mathcal{C}$, the weight w, and the reference indicator function r.

2.3 Wikipedia-based RefD Implementation

We now implement *RefD* using Wikipedia. As a widely used open-access encyclopedia, Wikipedia provides relatively up-to-date and high quality knowledge and has been successfully utilized as explicit concepts [48]. Moreover, the rich hyperlinks created by Wiki editors provide a natural way to calculate the reference indicator function r.

Specifically, the concept space $\mathcal{C}$ consists of all Wikipedia articles. $r(c, A)$ represents whether there is a link from Wiki article c to A. For $w(c, A)$, we experiment with two methods:

- *EQUAL*: A is represented by the concepts linked from it ($L(A)$) with equal weights.

$$w(c, A) = \begin{cases} 1 & \text{if } c \in L(A) \\ 0 & \text{if } c \notin L(A) \end{cases}$$

- *TFIDF*: A is represented by the concepts linked from it with TFIDF weights.

$$w(c, A) = \begin{cases} tf(c, A) * \log \frac{N}{df(c)} & \text{if } c \in L(A) \\ 0 & \text{if } c \notin L(A) \end{cases}$$

  where $tf(c, A)$ is the number of times c being linked from A; N is the total number of Wikipedia articles; and $df(c)$ is the number of Wikipedia articles where c appears.

2.4 Experiments

In order to evaluate the proposed metric, we apply it to predicting prerequisite relations in Wikipedia, i.e., whether one article in Wikipedia is a prerequisite of another article. Given a pair of concepts (A,B), we predict whether B is a prerequisite of A or not. Both pairs where A is a prerequisite of B and pairs where no prerequisite relation exists will be viewed as negative examples.

Dataset	Domain	# Pairs	# Prerequisites
CrowdComp	Meiosis	400	67
	Public-key Cryp.	200	27
	Parallel Postulate	200	25
	Newton's Laws	400	44
	Global Warming	400	43
Course	CS	678	108
	MATH	658	75

Table 2.1. Statistics of CrowdComp and Course Datasets

RefD is tested on two datasets: CrowdComp dataset [151] and a Course prerequisite dataset collected by us. We compare *RefD* with a Maximum Entropy (MaxEnt) classifier which exploits graph-based features such as PageRank scores and content-based features such as the category information, whether a title of concept is mentioned in the first sentence of the other concept, the number of times a concept is linked from the other, etc. [151]. All experiments use a Wikipedia dump of Dec 8, 2014.

2.4.1 Results on the CrowdComp Dataset

The CrowdComp dataset was collected using Amazon Mechanical Turk by Talukdar et al. [151]. It contains binary-labeled concept pairs from five different domains, including meiosis, public-key cryptography, the parallel postulate, Newton's laws of motion, and global warming. The label of the prerequisite relation for each pair is assigned using majority vote. Details of the dataset are shown in Table 2.1.

Following Talukdar et al. [151], we evaluate different methods in a "leave one domain out" manner, where data from one domain is used for testing and data from other four for training. Classes in the training and testing set are balanced by oversampling the minority class. Table 2.2 lists the accuracies of different methods. In terms of average performance, *RefD* achieves comparable average accuracy as MaxEnt. When TFIDF is used to calculate w, *RefD* performs better than MaxEnt. Also we notice that our implementation of MaxEnt classifier achieves higher accuracy than reported in the original paper, which may be due to the difference between Wiki dumps used. In addition, we can see that there are large differences in performance across different domains, which is mainly due to two

Domain	**MaxEnt**†	**MaxEnt**	**EQUAL**	**TFIDF**
Meiosis	51	60.2	53	55.7
Public-key Cryp.	67.1	60.3	55.1	57.7
Parallel Postulate	64.7	73.6	70.5	67.9
Newton's Laws	53.9	57.7	63.7	64.6
Global Warming	56.8	50.0	57.4	60.1
Average	58.7	60.4	60.0*	**61.2***

Table 2.2. Comparison of out-of-domain training accuracies of a MaxEnt classifier and *RefD* using EQUAL and TFIDF weighting. MaxEnt† is the number reported by Talukdar et al. [151]. MaxEnt shows the performance of our implementation. * indicates the difference between *RefD* and MaxEnt is statistically significant ($p < 0.01$).

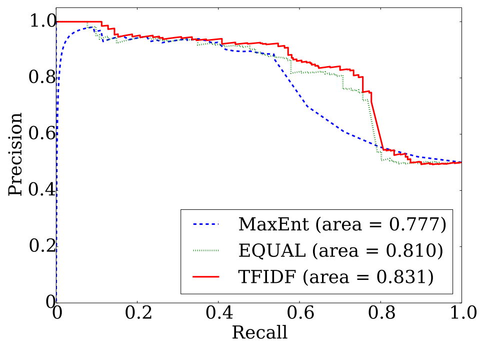
Method	**CS**				**MATH**			
	A	P	R	F	A	P	R	F
MaxEnt	72.8	87.6	53.2	66.1	69.0	78.1	53	63.1
EQUAL	76.4*	80.4	69.9	74.7*	**73.9***	78.4	67.3	**71.9***
TFIDF	**77.1***	82.3	69.1	**75.1***	70.3*	76.3	60.1	66.7*

Table 2.3. Comparison of in-domain training accuracies, precision, recall, and F1 measure of MaxEnt and *RefD* using EQUAL and TFIDF weighting. * indicates the improvement over MaxEnt is statistically significant ($p < 0.01$).

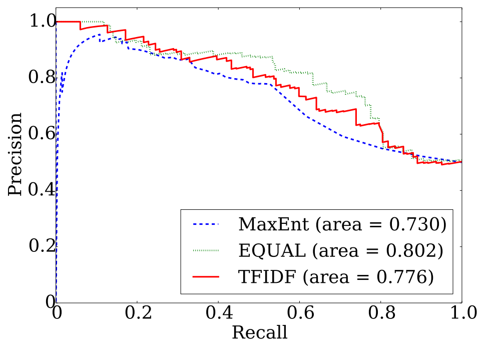
reasons. First, the coverage of Wikipedia for different domains may vary a lot. Some domains are more popular and thus edited more frequently, leading to a better quality of articles and a more complete link structure. Second, since the ground-truth labels are collected by crowdsourcing and there is no guarantee for workers' knowledge about a certain domain, the quality of labels for different domains varies.

2.4.2 Results on the Course Dataset

We also built a Course dataset with the help of information available on a university's course website containing prerequisite relations between courses. For example, "CS 331 Data Structures and Algorithms" is a prerequisite for "CS 422 Data mining". We get the prerequisite pairs by crawling the website and linking the course to Wikipedia using simple rules such as title matching and content similarity. In order to get negative samples, we randomly sample 600 pairs using concepts appearing



(a) CS



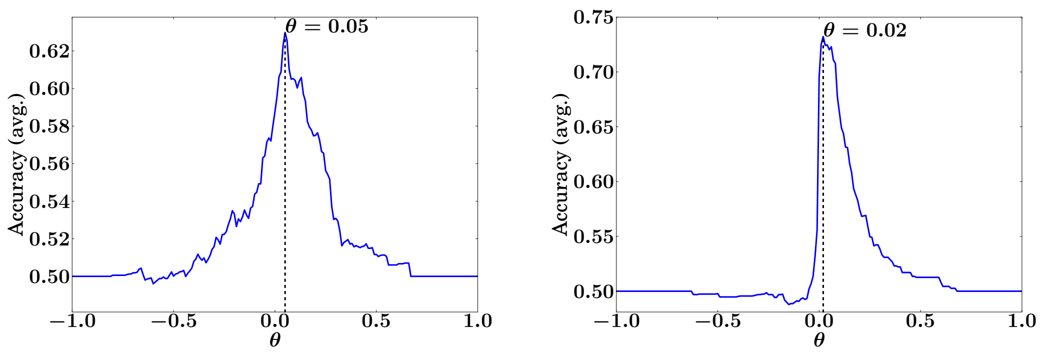
(b) MATH

Figure 2.2. Comparison of Precision-Recall curves of MaxEnt and *RefD* (using EQUAL and TFIDF weighting) on the Course dataset.

in the prerequisite pairs. All pairs are then checked by two domain experts by removing pairs with incorrect labels. Table 2.1 lists the information of the dataset.

Evaluation uses in-domain 5-fold cross-validation and classes are balanced by oversampling the minority class. Table 2.3 lists the performance comparison of different methods on accuracy, precision, recall and F1 score. We can see that *RefD* outperforms MaxEnt in terms of accuracy, recall, and F1 score on both CS and MATH domain. Because MaxEnt relies on many features but there are only limited distinct positive samples in the dataset, it is more likely to overfit the training data, which leads to high precision but low recall on test set. In order to better compare precision and recall, we plot the Precision-Recall curves of different methods, as shown in Figure 2.2. *RefD* shows a clear improvement in the area under the Precision-Recall curve.

Comparing two weighting methods, we find that TFIDF performs slightly better than EQUAL on CS while EQUAL has higher scores than TFIDF on MATH. Since how to compute w in *RefD* is a crucial problem, our ongoing work is to explore more sophisticated semantic representations to measure prerequisite relations. A natural extension to the two simple methods here is to represent a concept using WordNet [113], Explicit Semantic Analysis [48], or Word2vec embeddings [112]. Incorporating these representations may improve the performance of *RefD*.



(a) CrowdComp (b) Course

Figure 2.3. Average accuracy on two datasets with a given threshold of *RefD* using TFIDF weighting.

Concept	*RefD*	Concept	*RefD*	Concept	*RefD*
Deep belief network	-0.38	List of Nobel laureates	0.009	Machine learning	0.32
Neocognitron	-0.28	Neural development	0.009	Artificial neural network	0.31
Word embedding	-0.24	Watson (computer)	0.003	Artificial intelligence	0.15
Vanishing gradient problem	-0.22	Self-organization	8e-5	Algorithm	0.14
Feature learning	-0.17	Language model	-0.004	Statistical classification	0.13

Table 2.4. *RefD* scores between "deep learning" and the concepts linked from it. All scores are calculated by *RefD*('deep learning', concept).

2.4.3 Parameter Analysis and Case Study

Since using *RefD* to predict prerequisites requires setting a threshold θ, we also investigate the relation between the threshold and the performance of prediction, as shown in Figure 2.3. We can see that a threshold of 0.05 for *RefD* using TFIDF achieves the highest average accuracy on the CrowdComp dataset while a threshold of 0.02 works the best for Course dataset. Empirically we find that a threshold between 0.02 and 0.1 yields a good performance for prerequisite prediction task.

We further explore the performance of *RefD* through a case study for the concept "deep learning" (denoted as c'). Specifically, for any concept c linked from c' we calculate *RefD*(c', c). Table 2.4 lists the *RefD* scores for different concepts using *EQUAL* weighting. The concepts on the left have negative *RefD* scores with high absolute values, which means that "deep learning" is a prerequisite of them. Meanwhile concepts on the right have high positive *RefD* scores, which means that "deep learning" requires knowing them first. For example, people may first need to have some knowledge of "machine learning", "artificial intelligence" and "algorithm" in order to learn “deep learning”. Also we notice that concepts in the middle have *RefD* scores which are very close to 0, showing that there is no prerequisite relations between these concepts and “deep learning”. However, since our *RefD* implementation is based on Wikipedia, it might not give an accurate measure for concepts if they have no Wikipedia articles or their articles are too short to provide an encyclopedic coverage, such as “discriminative model” and “feature engineering”.

Please note that our Wikipedia-based implementation is computationally efficient especially after precomputing weights and references and can be easily incorporated as a feature into existing supervised learning based methods.

2.5 Related Work

Design of data-driven methods for automatically discovering prerequisite relations has been explored in multiple works. Established methods in educational data mining have been developed based on the automatic analysis of the assessment data acquired by students’ performance [28, 144, 156]. Such methods require that the association between test items and handcrafted knowledge components is set before hand and are not applicable for processing a large concept set. Liu et al. [100] studied learning-dependency between knowledge units using classification where a knowledge unit is a special text fragment containing concepts. We focus on more general prerequisite relations among concepts. As the largest open knowledge base, Wikipedia has also been studied to find prerequisite relations among concepts. [5, 151, 159], where both Wikipedia article contents and their link structures are utilized. Talukdar and Cohen [151] applied a Maximum Entropy classifier to predict prerequisite structures in Wikipedia using various features such as a random walk with restart score and PageRank score. Using Wikipedia, Wang et al. [159] designed a joint framework for key concept extraction and prerequisite identification to extract concept maps from textbooks. Instead of doing feature engineering, we here propose to measure prerequisite relations using a single metric. Yang et al. [169] proposed Concept Graph Learning to induce relations among concepts from prerequisite relations among courses, where the learned concept prerequisite relations are implicit and thus can not be evaluated. Our method is more interpretable for measuring prerequisite relations. Gordon et al. [53] proposed a information-theoretic metric to capture concept dependencies in a scientific corpus. Their method relies on topic modeling techniques and requires human annotaions of latent topics to make the result interpretable. Pan et al. [125] propose to include other features such as video references and sentence references for learning prerequisite relations among concepts in MOOCs.

Our work is closely related to the study of semantic relations. One related line of research is semantic hierarchy learning, where the goal is to automatically discover the hypernym-hyponym relation, or the "is-a" relation. The problem has been extensively studied. Established methods are usually based on lexical patterns [60, 109, 140], distributional similarity [79, 85], semantic word embeddings [46], web mining [47], etc. Despite being similar, prerequisite relation and "is-a" relation are different. For example, "linear algebra" is a prerequisite of "machine learning" but there is no hypernym-hyponym relation.

Another related research topic is knowledge graph completion. The concept prerequisite relation could be viewed as one type of relations among entities in the knowledge graph. The main task of knowledge graph completion is link prediction, whose goal is to predict new relations between entities on a knowledge graph by investigating existing relations of the graph [122]. The methods for link prediction can be categorized into the following groups. The first group utilizes graph features such as paths between entity pairs [81] and subgraphs [49]. The second group of work consists of methods based on Markov random fields [71, 132]. Knowledge graph embedding based methods [17, 148] are another group of methods for the link prediction task. These methods embed entities into a continuous low dimensional vector space and embed relations as vectors or matrices. The current state-of-the-art knowledge graph embedding is "translation-based" approach [16, 96, 161]. The assumption is that relations are "operators" used to translate entities into other positions in the embedding space. However, the best-performing methods for knowledge graph embedding usually require a large amount of knowledge triples for the training process, which makes them not directly applicable for concept prerequisite learning since there is no large-scale knowledge base for concept prerequisites.

Measuring concept prerequisite relation is also related to the study for measuring and categorizing semantic similarity or relatedness between linguistic items. The methods could be categorized into two groups. The first group is taxonomy-based methods [139], which use information from existing manually constructed taxonomies such as WordNet [4] and Wikitionary [172]. The other group is corpus-based similarity measures, which leverage the contextual information of words in the corpus. Recently corpus-based measures have been widely studied, where the key is to model the semantic representation based on a *latent* space, such as Latent Semantic Analysis (LSA) [38], Probabilistic Latent Semantic Analysis (PLSA) [65], Latent Dirichlet Allocation (LDA) [14] and distributed word embeddings [67, 112], or an *explicit* concept space, such as Explicit Semantic Analysis (ESA) [48], Salient Semantic Analysis (SSA) [59], Temporal Semantic Analysis (TSA) [134] and Sense-Aaware Semantic Analysis (SaSA) [167].

2.6 Summary and Discussion

This chapter studied the problem of measuring prerequisite relations among concepts and proposed RefD, a general, light-weight, and effective metric, to capture the relation. We presented Wikipedia-based implementations of RefD with two different weighting strategies. Experiments on two datasets including seven domains showed that our proposed metric outperformed existing baselines using supervised learning.

Promising future directions would be applying the framework of RefD to other contexts such as measuring the prerequisite relations or reading orders between papers and textbooks. In addition, RefD can be incorporated into existing supervised models for a more accurate measure. Chapter 4 and Chapter 5 will discuss supervised concept prerequisite learning where RefD could be used. Also it would be meaningful to explore ranking different prerequisites of a concept. Besides the rich link structure we could take advantage of more content information from Wikipedia and other resources such as textbooks and scientific papers. Chapter 3 will explore the use of course dependencies for finding concept prerequisite relations.

Chapter 3 Recovering Concept Prerequisite Relations from University Course Dependencies

3.1 Introduction

While it can benefit both learners and instructional designers, discovering prerequisite relations among concepts is usually done manually by domain experts [12]. However, it is inefficient and expensive — and does not scale with large concept sets. A possible solution for scaling is to develop or integrate approaches that automatically infer such prerequisites from the increasing amount of digital educational data. Available sources include knowledge bases, student assessment data, text books, course materials, etc.

Here, we focus on the problem of recovering concept prerequisite relations from course dependencies (denoted as **CPR-Recover** for short). We utilize a similar data setting as that of [169] but instead focus on recovering an accurate and universally shared concept graph from the observed course dependencies rather than extrapolating the course prerequisites to unseen course pairs. We have information and dependencies of courses collected from different universities. As shown in Figure 3.1, courses #1-4 are from the curriculum of University A. Based on their descriptions, course dependencies (e.g. Course #2 depends on Course #1) are used to recover concept prerequisite relations in a shared concept graph. To address the CPR-Recover problem, we propose an unsupervised optimization-based method

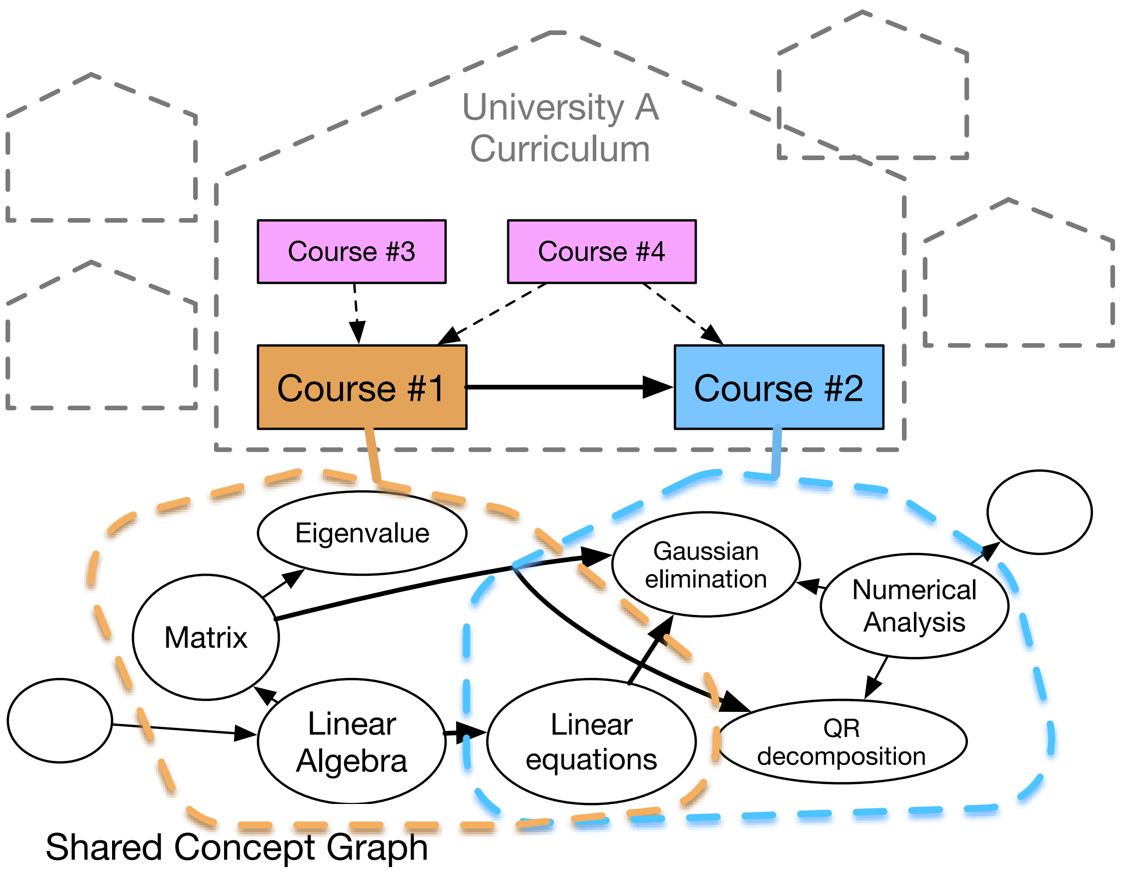


Figure 3.1. Recovering course prerequisite relations from course dependencies.

based on the following two assumptions:

1. **Causality**: The dependency among courses is caused by sufficient evidence provided by prerequisite relations among concepts representing the courses.

2. **Sparsity**: The prerequisites in a concept graph will be sparse, which means the number of prerequisite relations is much smaller than the total number of concept pairs.

Our method is designed to recover the part of concept prerequisite relations that causes course dependencies. Take the example in Figure 3.1, the prerequisite relations (Matrix $\Rightarrow$ Gaussian elimination, Matrix $\Rightarrow$ QR decomposition) are evidences for supporting the dependency between Course #1 and #2, thus can be recoverable from course data. Our method has been tested on both a synthetic dataset and a real computer science course dataset (including course names, descriptions, dependencies, etc.) collected from 11 U.S. universities. In order to make the discovered concept prerequisite relations explicit and interpretable, we represent these courses in terms of Wikipedia concepts. For the evaluation of concept prerequisites, we recruit a group of graduate students to assign the ground truth labels on a filtered subset of concept pairs. Experimental results on the two datasets both demonstrate the superior performance of our approach.

Our main contributions include:

1. A novel method to learn concept-level prerequisite relations from course dependencies that outperforms existing baselines on both a synthetic dataset and a real university course dataset.

2. The first real dataset for studying and evaluating the CPR-Recover problem, which consists of 654 unique computer science courses from 11 universities and 3544 concept pairs with their prerequisite labels.

3.2 Problem Setup

For convenience, we will use the following notations:

- $M = \{1, 2, ...m\}$ is the set of concepts where m is the number of concepts.
- $N = \{1, 2, ..., n\}$ is the set of all course IDs where n is the number of unique courses;
- $\mathbf{x}_i \in \mathbb{R}^m$ is the vector representation of the course i in concept space;
- $\{\mathbf{x}_i\}_{i \in N}$ is the set of courses;
- $i \mapsto j$ represents course i is a prerequisite of course j;
- $\Omega = \{i \mapsto j\}_{i,j \in N}$ is the set of prerequisite relations among courses;
- $\mathbf{A} = (a_{s,t})$ is a m-by-m matrix representing prerequisite relations among concepts, where $a_{s,t}$ is the weight quantifying how concept t depends on concept s.

CPR-Recover Problem Definition. Given a set of courses $\{\mathbf{x}_i\}_{i \in N}$ with a concept representation per course, and a set of observed course dependencies Ω, our goal is to recover the matrix $\mathbf{A}$ which quantifies the strength of prerequisite relations among concepts.

3.3 Our Approach

3.3.1 Course Representation

Since course descriptions are usually in the form of unstructured text, we first calculate a document representation for each course. In order to get an explicit and interpretable concept space, we choose to represent the text using Wikipedia concepts[1]. We represent each course as a bag-of-concepts model. Instead of using all words in the text, we first extract all Wikipedia concepts that are in the course description using Wikipedia Miner [115].

After concept extraction, the course vector $\mathbf{x}_i$ is calculated using term frequency-inverse document frequency (tf-idf).

$$\mathbf{x}_i[k] = tf(c_k, d_i) * \log \frac{W}{df(c_k)} \tag{3.1}$$

where c_k is the k-th concept in the concept space; d_i is the document for the i-th course; $tf(c_k, d_i)$ is the term frequency of c_k in d_i; W is the total number of Wikipedia articles; and $df(c_k)$ is the document frequency for concept c_k in Wikipedia.

3.3.2 CPR-Recover Formulation

Our approach to solve the CPR-Recover problem makes two assumptions: *causality assumption* and *sparsity assumption*. The former assumes that the prerequisite relation between two courses is caused by sufficient evidence provided by prerequisite relations among concepts which the two courses consist of. This serves as a bridge for making inferences between course dependencies and concept prerequisite relations. The latter assumption is that the prerequisite relations in a concept graph are

[1]Each concept corresponds to a unique English Wikipedia article.

sparse: the number of prerequisite relations is much smaller than the total number of concept pairs. Empirically, the frequency of a prerequisite relation existing between a random concept pair is low, even when the concepts come from the same domain. Since concepts are usually linked with their prerequisites, we validate our sparsity assumption by estimating the sparsity of a concept prerequisite graph by sampling from the Wikipedia link graph. A depth-first search under the category "Computer science" returns a domain-specific sub-graph with about 10^5 nodes and 2.8×10^6 edges. The graph density is $\sim 2.8 \times 10^{-4}$, showing the sampled graph is quite sparse.

Based on the two assumptions, we propose to solve the CPR-Recover problem by the following formulation:

$$\begin{aligned}
\min_{\mathbf{A},\xi} \quad & \|\text{vec}(\mathbf{A})\|_1 + \lambda \cdot \sum_{i \mapsto j \in \Omega} \xi_{i \mapsto j}^2 \\
\text{s.t.} \quad & \mathbf{x}_i^T \left[\mathbf{C_{i \mapsto j}} \odot \mathbf{A}\right] \mathbf{x}_j \geq \theta - \xi_{i \mapsto j}, \quad \forall i \mapsto j \in \Omega \\
& a_{s,t} + a_{t,s} = 0, \quad \forall s,t \in M \\
& -1 \leq a_{s,t} \leq 1, \quad \forall s,t \in M \\
& a_{s,t} = 0, \quad \forall (s,t) \notin \mathcal{K}
\end{aligned} \tag{3.2}$$

where θ and λ are constant positive parameters; $\xi_{i \mapsto j}$ is a slack variable for the course pair (i, j); $\odot$ is element-wise product; $\mathbf{C_{i \mapsto j}} = (c_{s,t})$ is a design matrix where $c_{s,t} \in \{0, 1\}$ and $c_{s,t} = 0$ if $\mathbf{x}_i[t] > 0$ or $\mathbf{x}_j[s] > 0$, otherwise $c_{s,t} = 1$; $\mathcal{K}$ is the set of candidate concept prerequisite relations obtained from external prior knowledge.

Constraints result from the causality and sparsity assumptions, and also external knowledge. The first constraint is based on the causality assumption. Every course dependency $i \mapsto j$ is caused by the interaction among $\mathbf{x}_i$, $\mathbf{A}$, and $\mathbf{x}_j$. $\mathbf{C_{i \mapsto j}}$ is incorporated to remove the contribution from the common concepts between $\mathbf{x}_i$ and $\mathbf{x}_j$ to the course dependency $i \mapsto j$. If concepts occur in both course i and j, we assume they are not the cause of the course dependency. The second constraint specifies that $\mathbf{A}$ is a skew-symmetric matrix, which means if concept c_a is a prerequisite of c_b then c_b is not a prerequisite of c_a. The third constraint bounds the strength of prerequisite relation in $[-1, 1]$. The last constraint allows the method to incorporate external knowledge. Specifically, $\mathcal{K}$ here consists of all Wikipedia concept pairs $\{(c_a, c_b)\}$ where there is at least one hyperlink between c_a and c_b. Following Talukdar et al. [151], we assume there is no prerequisite relation between two concepts which are not linked. Additional external knowledge can also be inserted. For example, if some of the concept prerequisite relations is already known from other resources such as manually-built concept graphs, this knowledge can also be incorporated into our method as constraints.

For the objective function, the first term is the regularization term and the second term is the empirical loss. In particular, the L1-norm on parameter **A** exploits the sparsity of computed prerequisite relations, and L2-norm of the slack variables is the loss in this setting. Two distinct variants of our approach come from mutating the choices of these two norms. By replacing the first term by L2-norm instead, one could show that a model proposed in [169] is a special case of this variant. And by replacing the first term by L2-norm and the second term by L1-norm, the resulting variant problem is the soft-margin SVM [33]. For solving problems of moderate scale within minutes, it suffices to use standard quadratic programming solvers. Scalable optimization techniques, such as ADMM or dual coordinate descent, are available to handle large-scale problems. Experimental results below validate that this choice of norms which is coherent with our model assumption and achieves empirically good performance.

3.4 Experiments

For evaluation we conduct experiments on two datasets created by us: a synthetic dataset and a real course dataset. To the best of our knowledge, there is no existing dataset that contains both the course dependency data and prerequisite labels for the underlying concept graph. All experiments are done on a Red Hat Enterprise Linux server with 24 Intel Xeon processors @ 2.67GHz and 32GB of RAM. Mosek[2] is used to solve the optimization problem.

3.4.1 Synthetic Dataset

To simulate the CPR-Recover problem, we generate a concept prerequisite graph, pairs of course with dependencies, and documents representing the courses.

[2] Available online at https://www.mosek.com/

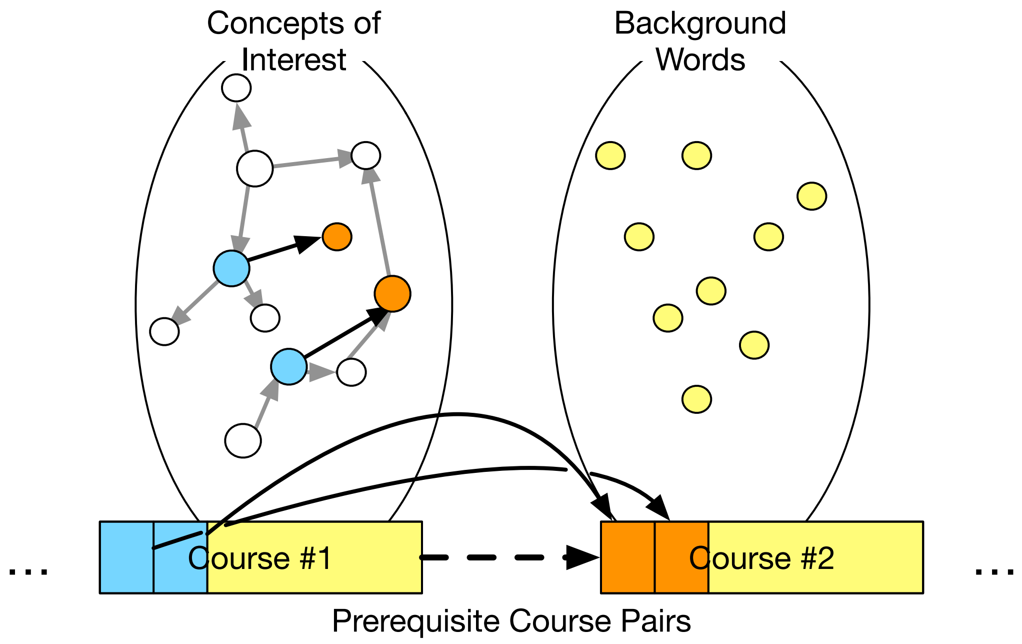


Figure 3.2. Process of synthetic data generation. After a concept prerequisite graph is created, prerequisite course pairs are generated by blending prerequisite concept pairs with background words.

3.4.1.1 Process of Data Generation

The process of data generation illustrated in Figure 3.2 consists of the following two steps:

1. (*Concept prerequisite graph generation*) We follow the Erdős-Rényi model to generate the underlying concept prerequisite graph, i.e., generating a random graph given the number of concepts (nodes) m and the probability p for creating a prerequisite relation (edge) between two concepts. Note every edge is set to be directed from a vertex to another with a larger concept ID. This step results in a directed graph G with m nodes and $|E|$ edges.

2. (*Prerequisite course pairs generation*) Next we generate k pairs of courses with prerequisite relations. Two assumptions are made: (i) The document representing the course consists of both concepts and background words. Prerequisite relations that only exist among concepts and background words are considered as noise. (ii) The prerequisite relation between two documents is consistent with the prerequisite relation between concepts in the two

G	m	$\|E\|$	$\|V\|$	p	l	l_c
G_S	100	100	100	0.01	10	2
G_L	500	2463	500			

Table 3.1. Statistics of two synthetic datasets.

documents. Specifically, given the document length l, the number of concepts in a document l_c, the vocabulary of background words V, we generate a pair of documents (d_i, d_j) representing $i \mapsto j$ with the following three steps:

(a) Randomly sample l_c edges from the concept prerequisite graph G generated from previous step.

(b) For every edge $c_s \mapsto c_t$ from the sampled l_c edges, add c_s to d_i, c_t to d_j.

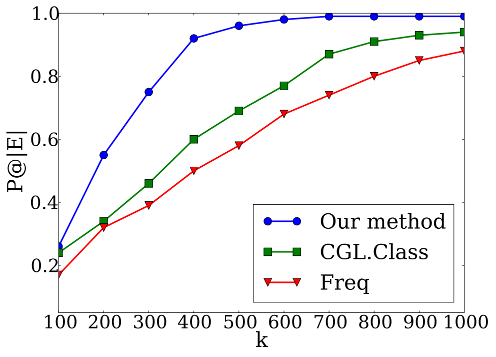
(c) Randomly sample $l - l_c$ background words from V and add them to d_i. Conduct a similar random sampling to add background words to d_j.

For our experiment, two concept prerequisite graphs, G_S and G_L, with different sizes are generated to test performance at different scales. A detailed description of the two experiment settings is shown in Table 3.1. The vocabulary size of background words is set equal to the number of nodes in G. The edge creation probability p is set to be 0.01 to enforce the sparsity on the concept graph. The document length l is set to 10 and each document consists of 2 concepts. We now explore the relation between the method performance and the number of course prerequisite pairs k. Specifically consider the following questions: will the concept prerequisite matrix $\mathbf{A}$ be better recovered if we obtain more pairs of course prerequisites? How will our method perform compared to other baselines?

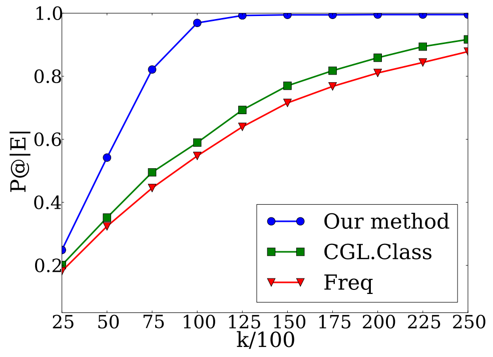
3.4.1.2 Baselines and Evaluation Metrics – Synthetic Data

For comparison we use the two following baselines, one a previous method — **CGL.Class** [169] — and the other a naive method — **Freq** which for the concept pair (c_s, c_t) calculates $a_{s,t}$ as the number of times the pair "co-occurs" in course prerequisite pairs. For $i \mapsto j$, (c_s, c_t) "co-occurs" if $\mathbf{x}_i[s] > 0$ and $\mathbf{x}_j[t] > 0$.

Precision at K ($P@K$) is calculated for the evaluation of $\mathbf{A}$. We first sort the elements of $\mathbf{A}$ in a descending order and get an ordered list $l_{\mathbf{A}} = \{a_{s,t}\}$. $P@K$ is calculated as $\frac{\sum_{i=1}^{K} rel(i)}{K}$, where $rel(\cdot)$ is binary indicator function; $rel(i) = 1$ if



(a) Results for G_S.



(b) Results for G_L.

Figure 3.3. Results on two synthetic datasets.

the concept pair (c_s, c_t) corresponding to the i-th element in $l_{\mathbf{A}}$ belongs to E. In our experiment, we set $K = |E|$ because ideally the top $|E|$ elements of $\mathbf{A}$ should correspond to all $|E|$ edges in the generated concept prerequisite graph.

3.4.1.3 Results – Synthetic Data

Experimental results on the two synthetic datasets are shown in Figure 3.3. For each method $P@|E|$ is calculated with a given number of course prerequisite pairs k. For the experiment setting G_S, k is chosen from $[100, 200, ..., 1000]$. And for G_L, k is chosen from $[2500, 5000, ..., 25000]$. We can see that results for G_S and G_L are consistent and show that: (i) As k increases, the performance of different methods keeps improving; (ii) Our method outperforms both *CGL.Class* and *Freq* for every choice of k. A closer look at the performances of the three methods shows that the difference is decreasing when k becomes small enough ($\approx |E|$) or large enough ($\approx 10|E|$). While for the former case there are not enough observations to make correct inferences, for the latter case all three methods are able to recover most of the concept prerequisites from the sufficient observed course prerequisites. In addition, we can see that when $k \approx 6|E|$ our method can recover almost all edges in G while the other two methods perform significantly worse. We explain this by noting that our method is designed to exploit the sparsity assumption for the concept prerequisite graph G. In contrast, *CGL.Class* and *Freq* do not.

3.4.2 University Course Dataset

In addition to the synthetic dataset, we also create a real course dataset based on data collected from 11 U.S universities, (Carnegie Mellon University, Stanford University, the Massachusetts Institute of Technology, Princeton University, the California Institute of Technology, Purdue University, University of Maryland, Michigan State University, Pennsylvania State University, University of Illinois, and University of Iowa) all with a focus on computer science or computer science like departments (CS). We developed a Web scraper to extract the course information from the online course catalogs of these universities. Besides the basic information such as course ID, name, and description, the course catalogs also provide course prerequisite information. For example, "CS 311 Data Structures and Algorithms" is a prerequisite for "CS 422 Data mining". Our focus on courses in the computer science is two-fold. We have domain expertise in that area and focusing on only one domain will make the ground truth acquisition easier and more realistic. After collecting the course descriptions, we apply Wikipedia Miner [115] to extract Wikipedia concepts.

In total from the 11 universities there were 654 unique CS courses (both undergrad and graduate level), 639 pairs of courses with prerequisite relations, and 569 Wikipedia concepts. And the average number of concepts per course was 4.73. The dataset is available at `https://github.com/harrylclc/eaai17-cpr-recover`.

3.4.2.1 Data Labeling

To accurately label for a given concept pair (c_s, c_t) whether c_s is an actual prerequisite of c_t requires domain knowledge. It can not be done by setting tasks on existing major crowdsourcing platforms such as Amazon Mechanical Turk where workers are not guaranteed to have the knowledge and expertise. As such for labeling we recruited 13 graduate students with CS backgrounds. Instead of labeling all pairs of concepts in the concept space, they only needed to label the set of candidate concept pairs $P = \{(c_s, c_t) | \exists i \mapsto j \in \Omega, \text{ s.t. } \mathbf{x}_i[s] > 0 \text{ and } \mathbf{x}_j[t] > 0\}$, i.e., only the concept pairs that "co-occur" in course prerequisite pairs. Note that for a concept pair $(c_s, c_t) \notin P$, $a_{s,t}$ would be unchanged and stay at zero during the optimization process.

In the course dataset, from 861 course prerequisites we get $|P| = 3544$. For each candidate pair (c_s, c_t), each student annotator decides whether c_s is a prerequisite of c_t and gives a binary label. The labeling task is assigned to the 13 annotators in a way that each concept pair can get labels from three annotators. The majority vote for the three labels is treated as the ground truth. Based on our setting of the labeling process, Fleiss' kappa κ [44] is used to assess the reliability of agreement between annotators. Note that $\kappa = 1$ if annotators are in complete agreement and $\kappa \leq 0$ if the observed agreement among the raters is no more than what would be expected by chance. For the labels we collected, $\kappa = 0.42$, which shows a level of moderate agreement. With no existing labeled concept prerequisite dataset available, such data, though not perfect, is necessary for evaluation. Finally, from the 3544 candidate concept pairs we get 1008 pairs of concepts with prerequisite relations, denoted as P_{true}.

3.4.2.2 Baselines and Evaluation Metrics – University Course Data

Besides *CGL.Class* and *Freq*, our method is also compared with the following two baselines. **RefD** [90]: A link-based metric for measuring prerequisite relations among Wikipedia concepts. For each $(c_s, c_t) \in P$, set $a_{s,t} = RefD(c_t, c_s)$. Note that instead of utilizing course dependencies this method requires the knowledge of the whole Wikipedia graph. We compare with this method in order to explore the performance difference between our method and methods based on external knowledge bases. **Random**: For each $(c_s, c_t) \in P$, randomly choose $a_{s,t}$ from $\{0, 1\}$ with equal probability. Each time the probability of discovering a true prerequisite pair is equal to the ratio of prerequisite pairs P_{true} to all candidate pairs P, which is $|P_{true}|/|P|$.

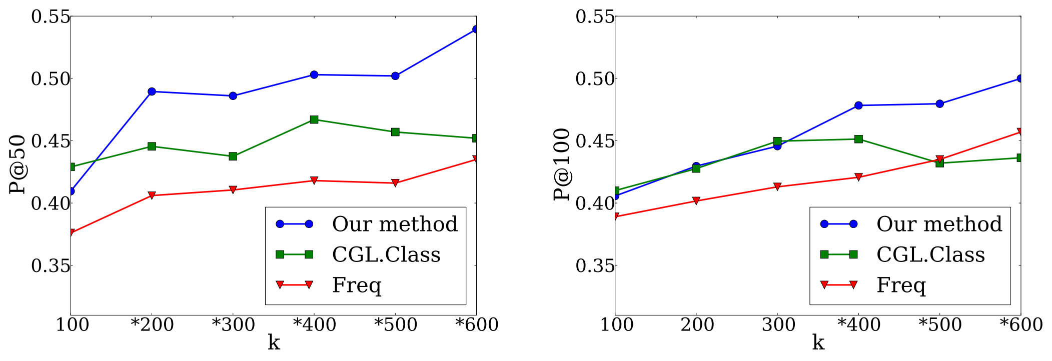
Given the size of concept graph and the limited number of observed course prerequisites in the dataset, the method is expected to recover only a part of all prerequisite concept pairs. Thus we focus on the Top-K precision performance. Specifically, for the evaluation of $\mathbf{A}$, we use precision at K ($P@K$) and average precision at K ($AP@K$). Given the ranked list of scores for each candidate pair, $l_P = \{a_{s,t}\}_{(c_s,c_t)\in P}$, $P@K$ and $AP@K$ are calculated by

$$P@K = \frac{\sum_{i=1}^{K} rel(i)}{K} \tag{3.3}$$

$$AP@K = \frac{\sum_{i=1}^{K} P@i \cdot rel(i)}{\sum_{i=1}^{K} rel(i)} \tag{3.4}$$

Methods	$P@50$	$P@100$	$AP@50$	$AP@100$
Our method	**0.54**	0.50	**0.60**	**0.57**
CGL.Class	0.46	0.42	0.56	0.51
Freq	0.44	0.46	0.37	0.41
RefD	0.52	**0.55**	0.42	0.49
Random	0.28			

Table 3.2. Results on the course dataset.



(a) Results for $P@50$. (b) Results for $P@100$.

Figure 3.4. Results using different numbers of pairs (k) of course dependencies. $*$ indicates that with a given k our method significantly outperforms other two methods ($p < 0.05$).

where $rel(\cdot)$ is binary indicator function; $rel(i) = 1$ if the concept pair (c_s, c_t) corresponding to the i-th element in l_P belongs to P_{true}. In our experiments, we compare different methods with K equal to 50 and 100.

3.4.2.3 Results – University Course Data

Note that there are two parameters θ and λ for our method. We perform a grid search on $\theta \in \{4, 8, 12, 16, 20, 24\}$ and $\lambda \in \{2^{-13}, 2^{-12}, ..., 2^{-6}, 2^{-5}\}$ to find the best parameters. We find that the parameter combination ($\theta = 8, \lambda = 2^{-8}$) yields the best $AP@100$. Table 3.2 lists the experiment results of different methods, from which we have the following findings: Our method outperforms all baselines in terms of $P@50$ and $AP@K$. As for $P@100$, our method performs better than *CGL.Class*, *Freq*, and *Random*, only worse than *RefD*. Such results are encouraging because, as previously mentioned, *RefD* utilizes the link structure of the entire Wikipedia, which contains much more information than the collected course prerequisite dataset. In other words, if we only compare methods which are strictly based on the CPR-Recover problem setting, our method is the best one w.r.t. both $P@K$ and $AP@K$.

Our method is also evaluated in a similar setting to the one used for the synthetic dataset, where our method is compared with *CGL.Class* and *Freq* using different number of course dependencies (k). Since in total only $|\Omega| = 639$ course dependencies are available, k is chosen from $[100, 200, ..., 600]$. For each k, we randomly sample k dependencies from Ω and compare different methods. We repeat such process 40 times and calculate the average performance. Results using different k are shown in Figure 3.4. While different methods perform similarly when k is small, the advantage of our method to other baselines becomes statistically significant ($p < 0.05$) given sufficient k, specifically when $k \geq 200$ for P@50 and $k \geq 400$ for P@100.

To analyze the human performance for identifying concept prerequisites, we evaluate each of the 13 student annotators based on the ground truth P_{true} (i.e., the consensus). The precision and F-score for identifying concept prerequisites by student annotators are 0.75±0.13 ($mean \pm SD$) and 0.75±0.08. Because they have previously acquired background knowledge, it is not surprising that the precision is much higher than the top-K precision of all methods in Table 3.2. From experimental results on synthetic dataset, we have noticed that our method benefits more from the increasing number of course dependencies than baselines do. If a sufficient number of course dependencies is given, the difference between the performances of our method and that of humans is expected to become smaller.

3.4.2.4 Case Study

We further investigate our method by studying examples of prerequisite concept pairs recovered. Table 3.3 lists examples of both true prerequisite pairs (TPP) and false prerequisite pairs (FPP), based on the ground truth labels collected. While we can see from the TPP that we can recover some of the concept prerequisites based on course dependencies, the FPP illustrate errors. Looking closely at the FPP, we hypothesize the errors are due to: (i) The performance of concept extraction is not that good. For example, the Wikipedia Miner extracts *Mathematical optimization* from the course description "...basic program analysis and optimization..." rather than find the correct concept *Program optimization*. *Program optimization* requires

Pairs	(c_s, c_t)
TPP	(Computer programming, System programming)
	(Algorithm, Computer graphics)
	(Probability, Machine learning)
	(Mathematical optimization, Machine learning)
	(Parallel computing, Parallel algorithm)
	(Computer graphics, Computer animation)
	(Computer science, Programming language)
	(Computer programming, Operating system)
	(Computer, Computer network)
	(Data structure, Software engineering)
FPP	(Computer science, Mathematical optimization)
	(Data structure, Analysis of algorithms)
	(Dynamic programming, Algorithm)
	(Computer software, Database)
	(Graph theory, Computer vision)

Table 3.3. Examples of prerequisite concept pairs we recovered, which are categorized as true prerequisite pairs (TPP) and false prerequisite pairs (FPP). Note c_s is expected to be a prerequisite of c_t.

Computer science, but *Mathematical optimization* does not. (ii) A Wikipedia concept can have different levels of interpretation, which will affect the choice for data labeling. For example, both *Database* and *DBMS* correspond to the same Wikipedia concept *Database*. *Computer software* is a prerequisite for *DBMS* but not for the general *Database*. (iii) For our dataset, concept pairs such as (*Dynamic programming*, *Algorithm*) and (*Data structure*, *Analysis of algorithms*) co-occur many times in course prerequisites. Such pairs are more likely to be recovered by data-driven methods including ours and the compared baselines.

3.5 Related Work

To the best of our knowledge, the problem of recovering concept prerequisite relations from course dependencies has not been systematically studied. However, our work is closely related to the design of data-driven methods for automatic prerequisite discovery. Besides the works that have already been discussed in Section 2.5, another closely related work is [169]. For automatic curriculum planing, Yang et al. [169] proposed a supervised framework to infer course prerequisites by constructing a latent concept graph to support prediction. Such a data setting is also utilized in our work. In comparison, their empirical efforts focused more on extrapolating the observed course prerequisites to unseen pairs, while we focus on recovering a universally shared concept graph. In their paper, the latent concept graph based on their approach — not yet formally evaluated as remarked in [169] — is also empirically evaluated with our method in the two experiments.

3.6 Summary

Through this chapter, we explored how to automatically discover concept prerequisite relations, which can further be utilized by many educational applications. Specifically, we proposed an effective data-driven method for recovering concept prerequisite relations from university course dependencies, the CPR-Recover problem. Our method was evaluated on a synthetic dataset and real course datasets derived from computer science or related course listings at 11 US universities and significatly outperformed existing baselines. To our knowledge, this is the first real dataset for the CPR-Recover problem.

Chapter 4 Investigating Active Learning for Concept Prerequisite Learning

4.1 Introduction

Although there has been research on learning prerequisites [90, 99, 125, 144, 151, 156, 159], the lack of large scale prerequisite labels remains a major obstacle for effective machine learning-based solutions. A possible solution for learning a good classifier given limited labeled instances is active learning [9, 30, 145], since it is designed to learn classifiers with significantly fewer labels by actively directing the query to the most "valuable" examples. As such, active learning methods could could also be applied to solving the current challenges of concept prerequisite learning.

To our knowledge, active learning has not been applied to the concept prerequisite learning problem. This chapter introduces the first attempt of applying active learning to the concept prerequisite learning problem. We investigate three families of query selection strategies by comparing their effectiveness on reducing the amount of training data. The first are informativeness-based methods such as uncertainty sampling [86] and query-by-committee [146]. The second are methods which take both informativeness and representativeness into account. The third are diversity-based strategies which aim to cover the feature space as broadly as possible. For classification, we propose a novel set of features for representing concept pairs and choose from four widely used classifiers the most suitable one for conducting active learning experiments. Our experiment results show a clear win for query-by-committee over other compared query strategies and show that active learning can be used to reduce the amount of training data required for concept prerequisite learning.

4.2 Pool-based Active Learning

Pool-based sampling [87] is a typical active learning scenario in which one maintains a labeled set $\mathcal{D}_l$ and an unlabeled set $\mathcal{D}_u$. In particular, we let $\mathcal{D}_u \cup \mathcal{D}_l = \mathcal{D} = \{1, \ldots, n\}$ and $\mathcal{D}_u \cap \mathcal{D}_l = \varnothing$. For $i \in \{1, \ldots, n\}$, we use $\mathbf{x}_i \in \mathbb{R}^d$ to denote a feature vector representing the i-th instance, and $y_i \in \{-1, +1\}$ to denote its ground truth class label. At each round, one or more instances are selected from $\mathcal{D}_u$ whose label(s) are then requested, and the labeled instance(s) are then moved to $\mathcal{D}_l$. Typically instances are queried in a prioritized way such that one can obtain good classifiers trained with a substantially smaller set D_l. We focus on the pool-based sampling setting where queries are selected in serial, i.e., one at a time. Algorithm 1 presents the typical setting of serial pool-based active learning.

Algorithm 1 Pseudocode for pool-based active learning.

```
Input:
  $\mathcal{D} \leftarrow \{1, 2, ..., n\}$    % a data set of $n$ instances
Initialize:
  $\mathcal{D}_l \leftarrow \{s_1, s_2, ..., s_k\}$    % initial labeled set with $k$ seeds
  $\mathcal{D}_u \leftarrow \mathcal{D} \backslash \mathcal{D}_l$    % initial unlabeled set
while $\mathcal{D}_u \neq \emptyset$ do
    Select $s^*$ from $\mathcal{D}_u$    % according to a query strategy
    Query the label $y_{s^*}$ for the selected instance $s^*$
    $\mathcal{D}_l \leftarrow \mathcal{D}_l \cup \{s^*\}$
    $\mathcal{D}_u \leftarrow \mathcal{D}_u \backslash \{s^*\}$
end while
```

4.2.1 Query Strategies

The key component of active learning is the design of an effective criterion for selecting the most "valuable" instance to query, which is often referred to as *query strategy*. We use s^* to refer to the selected instance by the strategy. In general, different strategies follow a greedy framework:

$$s^* = \underset{s \in D_u}{\operatorname{argmax}} \min_{y \in \{-1,1\}} f(s; y, D_l), \tag{4.1}$$

where $f(s; y, \mathcal{D}_l) \in \mathbb{R}$ is a scoring function to measure the risks of choosing y as the label for $\mathbf{x}_s \in D_u$ given an existing labeled set D_l.

We investigate four commonly used query strategies: uncertainty sampling [86], query-by-committee [146], QUIRE [68], and diversity sampling. These strategies are designed based on different assumptions: The first two selection strategies are based on the informativeness of the instance estimated by classifiers; QUIRE is based on the combination of informativeness and representativeness; Diversity sampling is based on the diversity in the feature space. Although being different, we show that under the binary classification setting, they can all be reformulated as Eq. (4.1).

Uncertainty Sampling selects the instance which it is least certain how to label. We choose to study one popular uncertainty-based sampling variant, the *least confident*. Subject to Eq. (4.1), the resulting approach is to let

$$f(s; y, \mathcal{D}_l) = 1 - P_{\Delta(\mathcal{D}_l)}(y_s = y | \mathbf{x}_s), \tag{4.2}$$

where $P_{\Delta(\mathcal{D}_l)}(y_s = y|\mathbf{x}_s)$ is a conditional probability which is estimated from a probabilistic classification model Δ trained on $\{(\mathbf{x}_i, y_i) \mid \forall i \in \mathcal{D}_l\}$.

Query-By-Committee maintains a committee of models trained on labeled data, $\mathcal{C}(\mathcal{D}_l) = \{g^{(1)}, ..., g^{(C)}\}$. It aims to reduce the size of version space. Specifically, it selects the unlabeled instance about which committee members disagree the most based on their predictions. Subject to Eq. (4.1), the resulting approach is to let

$$f(s; y, \mathcal{D}_l) = \textstyle\sum_{k=1}^{C} \mathbf{1}[y \neq g^{(k)}(\mathbf{x}_s)], \tag{4.3}$$

where $g^{(k)}(\mathbf{x}_s) \in \{-1, 1\}$ is the predicted label of $\mathbf{x}_s$ using the classifier $g^{(k)}$.

QUIRE aims to measure and combine the two types of query selection criteria, informativeness and representativeness, using a comprehensive max-margin framework. Subject to Eq. (4.1), the resulting approach is to let

$$f(s; y, \mathcal{D}_l) = (L_{u,l}\mathbf{y}_l + L_{u,s}y)^T L_{u,u}^{-1}(L_{u,l}\mathbf{y}_l + L_{u,s}y) - 2yL_{s,l}\mathbf{y}_l - L_{s,s}, \tag{4.4}$$

where $L = (K + \lambda I)^{-1}$ and K is the kernel matrix of size $n \times n$ and $f(s; y, \mathcal{D}_l)$ is equal to the *negative* margin if $y_s = y$ up to a constant. Due to limited space, please refer to [68] (pp. 1938–1940) for their detailed notations.

Diversity Sampling aims to select instances that cover as much of the feature space as possible. It selects the unlabeled instance with the lowest average cosine similarity between the instance's feature vector and those of the instances in the current training labeled dataset. Subject to Eq. (4.1), the resulting approach is to let

$$f(s; y, \mathcal{D}_l) = \sum_{i \in \mathcal{D}_l} 1 - \cos(\mathbf{x}_s, \mathbf{x}_i) \tag{4.5}$$

where $\cos(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{|\mathbf{x}_i||\mathbf{x}_j|}$ is the cosine similarity function. Note label y is not considered in this method.

4.3 Experimental Design

For experiments, we apply the previously mentioned active learning algorithms to *concept prerequisite learning problem* [90]. Given a pair of concepts (A, B), we predict whether or not A is a prerequisite of B, which is a binary classification problem. Here, cases where B is a prerequisite of A and where no prerequisite relation exists are both considered negative.

4.3.1 Dataset

We use the Wiki concept map dataset [159] which is collected from textbooks on four different educational domains. For each domain, the dataset consists of prerequisite pairs in the concept map. In the preprocessing stage, we validate whether each of the prerequisite relations in the dataset satisfies the required properties of a strict partial order (i.e., transitivity and irreflexivity) and ask domain experts to manually correct their labels if needed. We also expand the dataset by using the irreflexive and transitive properties: (i) add (B, A) as a negative sample if (A, B) is a positive sample; (ii) add (A, C) as a positive sample if both (A, B) and (B, C) are positive samples. Table 4.1 summarizes the statistics of the our final processed dataset.

Domain	**# Concepts**	**# Pairs**	**# Prerequisites**
Data Mining	120	826	292
Geometry	89	1681	524
Physics	153	1962	487
Precalculus	224	2060	699

Table 4.1. Dataset statistics.

4.3.2 Feature Description

For each concept pair (A, B), we calculate two types of features from information retrieval and natural language processing: graph-based and text-based features. Note that for all features, we use a Wikipedia dump of Oct. 2016.

4.3.2.1 Graph-based Features (GF)

The first type of feature is designed to utilize the link structure of Wikipedia. For convenience, we use the following notations: $In(A)$ is the set of concepts that link to A; $Out(A)$ is the set of concepts which A links to; $\mathcal{C} = \{c_1, ..., c_N\}$ is the concept space, i.e. all concepts in Wikipedia. Specifically, different types of graph-based features are:

- In/Out Degree. (GF #1-#4) The in/out degree of A/B.
- Common Neighbors. (GF #5) The number of common neighbors of A and B, i.e. $|Out(A) \cap Out(B)|$.
- # Links. (GF #6-#7) The number of times A/B links to B/A. The link structure within Wikipedia can be used as a proxy for prerequisite relations. The intuition is that a concept is usually linked to its prerequisites.
- Link Proportion. (GF #8-#9) The proportion of pages that link to A/B also link to B/A, i.e. $\frac{|In(A) \cap In(B)|}{|In(A)|}$ and $\frac{|In(A) \cap In(B)|}{|In(B)|}$.
- NGD. (GF #10) The Normalized Google Distance [164] between A and B. Specifically,

$$NGD(A, B) = \frac{\max(\log|In(A)|, \log|In(B)|) - \log|In(A) \cap In(B)|}{\log N - \min(\log|In(A)|, \log|In(B)|)} \quad (4.6)$$

- PMI. (GF #11) The Pointwise Mutual Information relatedness between the incoming links of A and B. [137]

$$PMI(A, B) = \log \frac{N \cdot |In(A) \cap In(B)|}{|In(A)| \cdot |In(B)|} \tag{4.7}$$

- RefD. (GF #12) A metric to measure how differently A and B's related concepts link to each other. Proposed by [90], RefD has been used as a proxy to measure concept prerequisite relations.

$$RefD(A, B) = \frac{\sum_{i=1}^{N} r(c_i, B) \cdot w(c_i, A)}{\sum_{i=1}^{N} w(c_i, A)} - \frac{\sum_{i=1}^{N} r(c_i, A) \cdot w(c_i, B)}{\sum_{i=1}^{N} w(c_i, B)} \tag{4.8}$$

  where $w(c_i, A)$ weights the importance of c_i to A; and $r(c_i, A)$ is an indicator showing whether c_i links to A.

- PageRank. (GF #13) The difference between A and B's PageRank scores [124]. The PageRank score, based on the link analysis, can be used to estimate the importance of concepts.

- HITS. (GF #14-#15) The difference between A and B's hub scores and the difference between their authority scores [78]. Similar to PageRank, authority and hub scores are used as proxies for concept importance.

4.3.2.2 Text-based Features (TF)

The second type of feature is designed to utilize textual content in the Wikipedia page. Note we have trained a topic model [14] (#topics=300) on the Wiki corpus. We have also trained a word2vec [112] (size=300) model on the same corpus with each concept treated as an individual token. Specifically, different types of text-based features are:

- 1st Sent. (TF #1-#2) Whether A/B appears in the first sentence of B/A. The first sentence of a Wikipedia article is usually the definition of the concept and the concepts mentioned in the sentence are more likely to be a prerequisite.

- In Title. (TF #3) Whether A appears in B's title. For example, "machine learning" is contained in "Supervised machine learning".

- Title Jaccard. (TF #4) The Jaccard similarity between A and B's titles.
- Length. (TF #5-#6) The number of words of A/B's content. This might serve as a proxy for complexity level and popularity of the concept.
- Mention. (TF #7-#8) The number of times A/B are mentioned in the content of B/A. The intuition is that the important prerequisites of a concept might be mentioned many times in its content.
- NP. (TF #9-#11) The number of noun phrases in A/B's content; The number of common noun phrases. If the concept is very general, its content tends to have more noun phrases.
- Tf-idf Sim. (TF #12) The cosine similarity between Tf-idf vectors for A and B's first paragraphs.
- Word2vec Sim. (TF #13) The cosine similarity between vectors of A and B trained by word2vec. Both word2vec and tf-idf similarities are measures for semantic relatedness, which is needed because usually two concepts with prerequisite relation are semantically related.
- LDA Entropy. (TF #14-#15) The Shannon entropy of the LDA vector of A/B.

  $$H(A) = -\sum_{i}^{T} p_{Ai} \log p_{Ai} \tag{4.9}$$

  where $\mathbf{p}_A$ is A's LDA vector, i.e., the distribution over T topics. More advanced concepts usually focus on fewer topics, thus leading to a lower LDA entropy.
- LDA Cross Entropy. (TF #16-#17) The cross entropy between the LDA vector of A/B and B/A. Gordon et al. [53] propose to use this feature to capture concept dependencies in a scientific corpus.

  $$H(A; B) = H(A) + D_{KL}(A||B) \tag{4.10}$$

  where $H(A)$ is the entropy of A's LDA vector, and $D_{KL}(A||B)$ is the Kullback-Leibler divergence between A and B's LDA vectors.

Classifier	Metric	Data Mining	Geometry	Physics	Precalulus
NB	P	71.5	84.4	54.3	85.7
	R	28.5	44.3	71.9	66.9
	$F1$	37.8	58.1	61.6	75.0
	AUC	81.4	87.1	85.5	93.2
LR	P	65.8	71.3	58.0	81.7
	R	**77.4**	81.3	**78.8**	**88.4**
	$F1$	71.1	75.8	66.8	84.8
	AUC	85.9	91.6	89.2	95.4
SVM	P	73.7	82.8	77.9	86.7
	R	64.7	69.9	50.3	81.4
	$F1$	68.6	75.5	61.1	83.9
	AUC	85.0	91.3	88.8	95.1
RF	P	**80.7**	**95.0**	**85.2**	**90.2**
	R	73.3	**84.7**	59.3	87.1
	$F1$	**76.7**	**89.5**	**69.9**	**88.6**
	AUC	**92.2**	**97.8**	**93.9**	**97.5**

Table 4.2. Results (%) for concept prerequisite relation classification.

4.4 Experimental Results

4.4.1 Evaluation of Classification

Before investigating the performance of active learning, we first evaluate concept prerequisite learning under the traditional binary classification setting. In our experiments, we employ four widely used binary classifiers: Naïve Bayes (NB), Logistic Regression (LR), Support Vector Machine (SVM) [33], and Random Forest (RF) [18]. Specifically, we set $C = 1.0$ for LR, use a linear kernel for SVM, and use 200 trees for RF. For each dataset, we apply 5-fold cross validation and report the average precision (P), recall (R), F1-score ($F1$) and Area under the ROC curve (AUC).

As shown in Table 4.2, the classification results vary by different methods. Overall, Naïve Bayes performs the worst in terms of both $F1$ and AUC, which is due to the fact that the strong independence assumption does not hold for our designed feature set. For example, the number of noun phrases might be correlated with the number of words; PageRank and HITS scores are not independent either.

Data Mining	**Geometry**	**Physics**	**Precalculus**
Authority diff	PageRank diff	PageRank diff	PageRank diff
LDA entropy of A	In degree of A	RefD	Authority diff
PageRank diff	Out degree of A	# mentions of A in B	RefD
In degree of A	RefD	In degree of B	# mentions of A in B
RefD	# mentions of A in B	Authority diff	Out degree of A
LDA entropy of B	LDA entropy of A	Link proportion of B	A in B's 1st sentence
In degree of B	A in B's 1st sentence	Out degree of A	In degree of A
LDA cross entropy (A;B)	Length of A	In degree of A	Hub diff
Link proportion of A	# NPs in A	LDA entropy of A	# NPs in A
LDA cross entropy (B;A)	# mentions of B in A	# NPs in B	# mentions of A in B

Table 4.3. Top 10 important features for each domain.

As linear classification models, LR and SVM lead to similar $F1$ and AUC while the former has higher recall and the latter has higher precision. Among four methods, Random Forest outperforms other three across all four domains, by 5.6%, 13.7%, 3.1%, and 3.8% respectively w.r.t. $F1$ and 6.3%, 6.1%, 4.7%, and 2.1% w.r.t. AUC. This might be because, compared with a linear combination of features for classification, the procedure of RF (the bagging and random selection of feature set) is more suitable for capturing the relation between the proposed feature set and concept prerequisite relations. We use RF in the following experiments as the classification model.

4.4.2 Feature Analysis

We also conduct a feature analysis in order to gain more insights on the proposed feature set. Table 4.3 lists top 10 features for each domain. Since Random Forest is used, the feature importance is calculated by "mean decrease impurity". It is defined as the total decrease in node impurity, weighted by the probability of reaching that node, averaged over all trees of the ensemble. From Table 4.3, we can observe the following: (i) While the ranking of features is different across four domains, there are many common important features such as PageRank, HITS's authority score, RefD, etc; (ii) Among top features, there are more graph-based features than text-based features. This might be because current text-based features are still very simple and more effective text features are yet to be explored. Several possible choices include lexico-syntactic patterns [60], structural features [125], etc. (iii) Top text-based features are LDA entropy, LDA cross entropy, Mention, and NP. Similarity-based features such as Tf-idf and Word2vec similarities are not as important; (iv) Top graph-based features are PageRank, authority score, RefD, in/out degree, and link proportion. Other graph features such as common neighbors, NGD, and PMI are less important. From observation (iii) and (iv) we find that symmetric pairwise features such as similarity and PMI are not important in current in-domain classification setting. This can be explained by noticing that the motivation of designing these features is to add constraints on the semantic relatedness, which is usually already satisfied in the in-domain setting. We expect such features to be more important in a cross-domain classification setting, where the concept space is much larger and more diverse.

4.4.3 Evaluation of Active Learning

4.4.3.1 Settings

We follow the typical evaluation protocol of pool-based active learning. We first randomly split a dataset into a training set $\mathcal{D}$ and a test set $\mathcal{D}_{test}$ with a ratio of 2:1. Then we randomly select 20 samples from the training set as the initial query set Q and compute its closure $\mathcal{D}_l$. Meanwhile, we set $\mathcal{D}_u = \mathcal{D} \backslash \mathcal{D}_l$. In each iteration, we pick an unlabeled instance from $\mathcal{D}_u$ to query for its label, update the label set $\mathcal{D}_l$, and re-train a classification model on the updated $\mathcal{D}_l \cap \mathcal{D}$. The re-trained classification model is then evaluated on $\mathcal{D}_{test}$. In all experiments, we use a random forests classifier [18] with 200 trees as the classification model. We use Area under the ROC curve (AUC) as the evaluation metric. Taking into account the effects of randomness subject to different initializations, we continue the above experimental process for each method repeatedly with 300 preselected distinct random seeds. Their average scores and confidence intervals ($\alpha = 0.05$) are reported. We compare the following five query strategies, most of which have been introduced in previous sections:

- Random: randomly selecting an instance to query. We choose this as the baseline for comparison.

- LC: least confident sampling, a widely used uncertainty sampling variant. We use a logistic regression model to estimate the posterior probabilities.

- QBC: query-by-committee algorithm. We apply query-by-bagging [108] and use a committee of three decision trees.

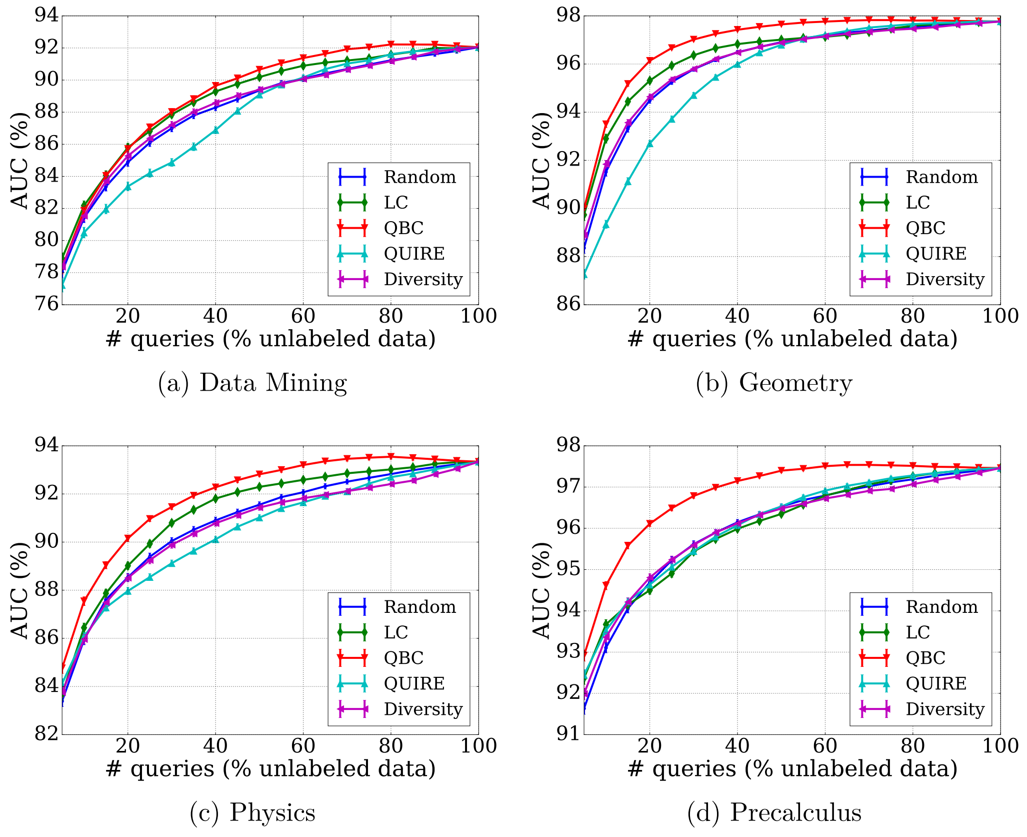


Figure 4.1. Comparison of different query strategies for concept prerequisite classification.

- QUIRE: a strategy for querying informative and representative examples. We follow the authors' experimental approach to use an RBF kernel and set the parameter $\lambda = 1$.

- Diversity: a strategy for selecting the unlabeled instance that is as diverse as possible in the feature space of current labeled set.

4.4.3.2 Results

Figure 4.1 shows the AUC results of different query strategies for concept prerequisite learning. For each case, we present the average values and 95% confidence intervals of repeated 300 trials with different train/test splits. From the figure we have the following observations:

First, comparing results on four domains, we can find different query strategies have relatively consistent learning curves, with the only exception of LC on the Precalculus domain. This is possibly caused by that the number of concept in Precalculus is much larger than other domains and the logistic regression classifier used by LC failed to give an accurate uncertainty estimation. Second, when comparing different strategies with the Random baseline, we find: (i) Informativeness-based methods (least confident sampling and query-by-committee) show substantial improvement over random; QBC is constantly outperforming other query strategies on all domains, which shows the advantage of using ensemble method to estimate uncertainty over the single linear classification model as used by LC. This again suggests that decision tree-based classifiers are more effective given the proposed feature set. (ii) Diversity-sampling is not significantly different from random, which suggests that choosing the instance as diverse as possible in the proposed feature space is not effective. (iii) QUIRE performs worse than random, especially during the early stage of active learning. It is also worth mentioning that QUIRE requires significantly longer time for choosing instances because of calculating the inverse and determinant of large matrices. In addition, for our datasets, by empirically tuning the RBF parameter γ for the best, we still did not find any advantages of QUIRE over LC or QBC. This might be because the used RBF kernel, on which QUIRE's performance is critically dependent, does not really suit our provided features.

To sum up, we find that informativeness-based query strategies, especially query-by-committee, is more effective for concept prerequisite learning given the proposed feature set. Different active learning strategies can be used to reduce the amount of training data required to get an expected AUC score for concept prerequisite learning.

4.5 Summary and Discussion

We made several contributions to concept prerequisite learning. In order to deal with the lack of large scale labels which makes problematic supervised learning for concept prerequisite learning, we investigated the applicability of active learning. Our active learning experiments for comparing different query strategies found that query-by-committee constantly outperforms other methods including uncertainty sampling, QUIRE, and diversity sampling. We proposed a novel set of features for concept pair representation tailored for the concept prerequisite learning problem. The top features identified by the feature importance analysis hopefully will be helpful for other supervised prerequisite learning methods.

Built upon the study described in this chapter, Chapter 5 will explore active learning query strategies better tailored to the concept prerequisite learning problem. In the typical setup of active learning, the dependency among labeled or unlabeled instances is not considered. However, since the prerequisite relation is both transitive and irreflexive, then when an unlabeled instance is labeled, there could be other unlabeled instances whose labels can be deduced by applying logical reasoning with the two properties. Query strategies that can take such properties into account will make active learning more effective.

It would be useful to investigate in more detail the semantic representation of concept pairs for prerequisite learning and to design more complex features such as complexity level features, structural features, etc. and see their effect on the classification performance.

Chapter 5 Active Learning of Strict Partial Orders: A Case Study on Concept Prerequisite Relations

5.1 Introduction

Pool-based active learning is a learning framework where the learning algorithm is allowed to access a set of unlabeled examples and ask for the labels of any of these examples [9, 30, 145]. Its goal is to learn a good classifier with significantly fewer labels by actively directing the queries to the most "valuable" examples. In a typical setup of active learning, the label dependency among labeled or unlabeled examples is not considered. But data and knowledge in the real world are often embodied with prior relational structures. Taking into consideration those structures in building machine learning solutions can be necessary and crucial [50].

The goal of this chapter is to investigate the query strategies in active learning of a strict partial order, namely, when the ground-truth labels of examples constitute an irreflexive and transitive relation. In this chapter, we develop efficient and effective algorithms extending popular query strategies used in active learning to work with such relational data. We study the following problem in the active learning context:

Definition 5 (Active Learning of Strict Orders). *Given a finite set V, a strict order on V is a type of irreflexive and transitive (pairwise) relation. Such a strict order is represented by a subset $G \subseteq V \times V$. Given an unknown strict order G, an* *oracle W that returns $W(u,v) = -1 + 2 \cdot \mathbf{1}[(u,v) \in G] \in \{-1, 1\}$, and a feature extractor $\mathcal{F} : V \times V \mapsto \mathbb{R}^d$, find $h : \mathbb{R}^d \mapsto \{-1, 1\}$ from a hypothesis class $\mathcal{H}$ that predicts whether or not $(u,v) \in G$ for each pair $(u,v) \in V \times V$ and $u \neq v$ (using $h(\mathcal{F}(u,v))$) by querying W a finite number of (u,v) pairs from $V \times V$.*

Our main focus is to develop reasonable query strategies in active learning of a strict order exploiting both the knowledge from (non-consistent) classifiers trained on a limited number of labeled examples and the deductive structures among pairwise relations. Our work also has a particular focus on *partial orders*. If the strict order is total, a large school called "learning to rank" has studied this topic [20,102], some of which are under the active learning setting [39,103]. Learning to rank relies on binary classifiers or probabilistic models that are consistent with the rule of a total order. Such approaches are however limited in a sense to principally modeling a partial order: a classifier consistent with a total order will always have a non-zero lower bound of error rate, if the ground-truth is a partial order but not a total order.

In our active learning problem, incorporating the deductive relations of a strict order in soliciting examples to be labeled is non-trivial and important. The challenges motivating us to pursue this direction can be explained in three folds: First, any example whose label can be deterministically reasoned from a labeled set by using the properties of strict orders does not need further manual labeling or statistical prediction. Second, probabilistic inference of labels based on the independence hypothesis, as is done in the conventional classifier training, is *not* proper any more because the deductive relations make the labels of examples dependent on each other. Third, in order to quantify how valuable an example is for querying, one has to combine uncertainty and logic to build proper representations. Sound and efficient heuristics with empirical success are to be explored.

One related active learning work that deals with a similar setting to ours is [138], whereas equivalence relations are considered instead. Particularly, they made several crude approximations in order to expedite the expected error calculation to a computational tractable level. We approach the design of query strategies from a different perspective while keeping efficiency as one of our central concerns.

To empirically study the proposed active learning algorithm, we apply it to *concept prerequisite learning problem* [90,151], where the goal is to predict whether a concept A is a prerequisite of a concept B given the pair (A, B). Although there have been some research efforts towards learning prerequisites [90, 99, 125, 144, 151, 156, 159], the mathematical nature of the prerequisite relation as strict partial orders has not been investigated. In addition, one obstacle for effective learning-based solutions to this problem is the lack of large scale prerequisite labels. Liang et al. [93] applied standard active learning to this problem without utilizing relation properties of prerequisites. Active learning methods tailored for strict partial orders provide a good opportunity to tackle the current challenges of concept prerequisite learning.

Our main contributions are summarized as follows: First, we propose a new efficient reasoning module for monotonically calculating the deductive closure under the assumption of a strict order. This computational module can be useful for general AI solutions that need fast reasoning in regard to strict orders. Second, we apply our reasoning module to extend two popular active learning approaches to handle relational data and empirically achieve substantial improvements. This is the first attempt to design active learning query strategies tailored for strict partial orders. Third, under the proposed framework, we solve the problem of concept prerequisite learning and our approach appears to be successful on data from four educational domains, whereas previous work have not exploited the relational structure of prerequisites as strict partial orders in a principled way.

5.2 Reasoning of a Strict Order

5.2.1 Preliminary

Definition 6 (Strict Order)**.** *Given a finite set V, a subset G of $V \times V$ is called a strict order if and only if it satisfies the two conditions: (i) if $(a, b) \in G$ and $(b, c) \in G$, then $(a, c) \in G$; (ii) if $(a, b) \in G$, then $(b, a) \notin G$.*

Definition 7 (G-Oracle)**.** *For two subsets $G, H \subseteq V \times V$, a function denoted as $W_H(\cdot, \cdot) : H \mapsto \{-1, 1\}$ is called a G-oracle on H iff for any $(u, v) \in H$, $W_H(u, v) = -1 + 2 \cdot \mathbf{1}[(u, v) \in G]$.*

The G-oracle returns a label denoting whether a pair belongs to G.

Definition 8 (Completeness of an Oracle)**.** *A G-oracle of strict order W_H is called complete if and only if H satisfies: for any $a, b, c \in V$, (i) if $(a, b) \in H \cap G$,*

$(b, c) \in H \cap G$, then $(a, c) \in H \cap G$; (ii) if $(a, b) \in H \cap G$, $(a, c) \in H \cap G^c$, then $(b, c) \in H \cap G^c$; (iii) if $(b, c) \in H \cap G$, $(a, c) \in H \cap G^c$, then $(a, b) \in H \cap G^c$; (iv) if $(a, b) \in H \cap G$, then $(b, a) \in H \cap G^c$, where G^c is the complement of G.

W_H is called complete if it is consistent under transitivity when restricted on pairs from H.

Definition 9 (Closure). *Given a strict order G, for any $H \subseteq V \times V$, its closure is defined to be the smallest set $\overline{H}$ such that $H \subseteq \overline{H}$ and the G-oracle $W_{\overline{H}}$ is complete.*

Proposition 1. *For any $H \subseteq V \times V$, the closure of H subject to a strict order G is unique.* [1]

Proposition 2. *Let G be a strict order of V. For a complete G-oracle W_H, $H \cap G$ is also a strict order of V.*

Definition 10 (Descendant and Ancestor). *Given a strict order G of V and $a \in V$, its ancestor subject to G is $A_a^G := \{b \mid (b, a) \in G\}$ and its descendant is $D_a^G := \{b \mid (a, b) \in G\}$.*

5.2.2 Reasoning Module for Closure Calculation

With the definitions in the previous section, this section proposes a reasoning module that is designed to monotonically calculate the deductive closure for strict orders. Remark that a key difference between the traditional transitive closure and our definition of closure (Definition 8&9) is that the former only focuses on G but the latter requires calculation for both G and G^c. In the context of machine learning, relations in G and G^c correspond to positive examples and negative examples, respectively. Since both of these examples are crucial for training classifiers, existing algorithms for calculating transitive closure such as the Warshall algorithm are not applicable. Thus we propose the following theorem for monotonically computing the closure.

Theorem 1. *Let G be a strict order of V and W_H a complete G-oracle on $H \subseteq V \times V$. For any pair $(a, b) \in V \times V$, define the notation $C_{(a,b)}$ by*

(i) If $(a, b) \in H$, $C_{(a,b)} := H$.

[1]Please see Appendix for the proofs of propositions and theorems introduced hereafter.

(ii) If $(a,b) \in G^c \cap H^c$, $C_{(a,b)} := H \cup N'_{(a,b)}$ where

$$N'_{(a,b)} := \{(d,c) | c \in A_b^{G\cap H} \cup \{b\}, d \in D_a^{G\cap H} \cup \{a\}\},$$

and particularly $N'_{(a,b)} \subseteq G^c$.

(iii) If $(a,b) \in G \cap H^c$, $C_{(a,b)} := H \cup N_{(a,b)} \cup R_{(a,b)} \cup S_{(a,b)} \cup T_{(a,b)} \cup O_{(a,b)}$, where

$$\begin{aligned}
N_{(a,b)} &:= \{(c,d) \mid c \in A_a^{G\cap H} \cup \{a\}, d \in D_b^{G\cap H} \cup \{b\}\},\\
R_{(a,b)} &:= \{(d,c) \mid (c,d) \in N_{(a,b)}\},\\
S_{(a,b)} &:= \{(d,e) \mid c \in A_a^{G\cap H} \cup \{a\}, d \in D_b^{G\cap H} \cup \{b\},\\
&\qquad (c,e) \in G^c \cap H\},\\
T_{(a,b)} &:= \{(e,c) \mid c \in A_a^{G\cap H} \cup \{a\}, d \in D_b^{G\cap H} \cup \{b\},\\
&\qquad (e,d) \in G^c \cap H\},\\
O_{(a,b)} &:= \textstyle\bigcup_{(c,d)\in S_{(a,b)}\cup T_{(a,b)}} N''_{(c,d)},\\
N''_{(c,d)} &:= \{(f,e) \mid e \in A_d^{G\cap(H\cup N_{(a,b)})} \cup \{d\},\\
&\qquad f \in D_c^{G\cap(H\cup N_{(a,b)})} \cup \{c\}\}.
\end{aligned}$$

In particular, $N_{(a,b)} \subseteq G$ and $R_{(a,b)} \cup S_{(a,b)} \cup T_{(a,b)} \cup O_{(a,b)} \subseteq G^c$.

For any pair $(x,y) \in V \times V$, the closure of $H' = H \cup \{(x,y)\}$ is $C_{(x,y)}$.

Figure 5.1 provides an informal explanation of each necessary condition (except for $R_{(a,b)}$) mentioned in the theorem. If (a,b) is a positive example, i.e. $(a,b) \in G$, then (i) $N_{(a,b)}$ is a set of inferred positive examples by transitivity; (ii) $R_{(a,b)}$ is a set of inferred negative examples by irreflexivity; (iii) $S_{(a,b)}$ and $T_{(a,b)}$ are sets of inferred negative examples by transitivity; (iv) $O_{(a,b)}$ is a set of negative examples inferred from $S_{(a,b)}$ and $T_{(a,b)}$. If (a,b) is a negative example, i.e. $(a,b) \in G^c$, then $N'_{(a,b)}$ is a set of negative examples inferred by transitivity.

5.2.3 Computational Efficiency

As we will elaborate later, one computational hurdle of our active learning algorithm is to efficiently calculate the closure set $C_{(x,y)}$ given a complete G-oracle W_H. In particular, among all the formula in Theorem 1, we found the main bottleneck is

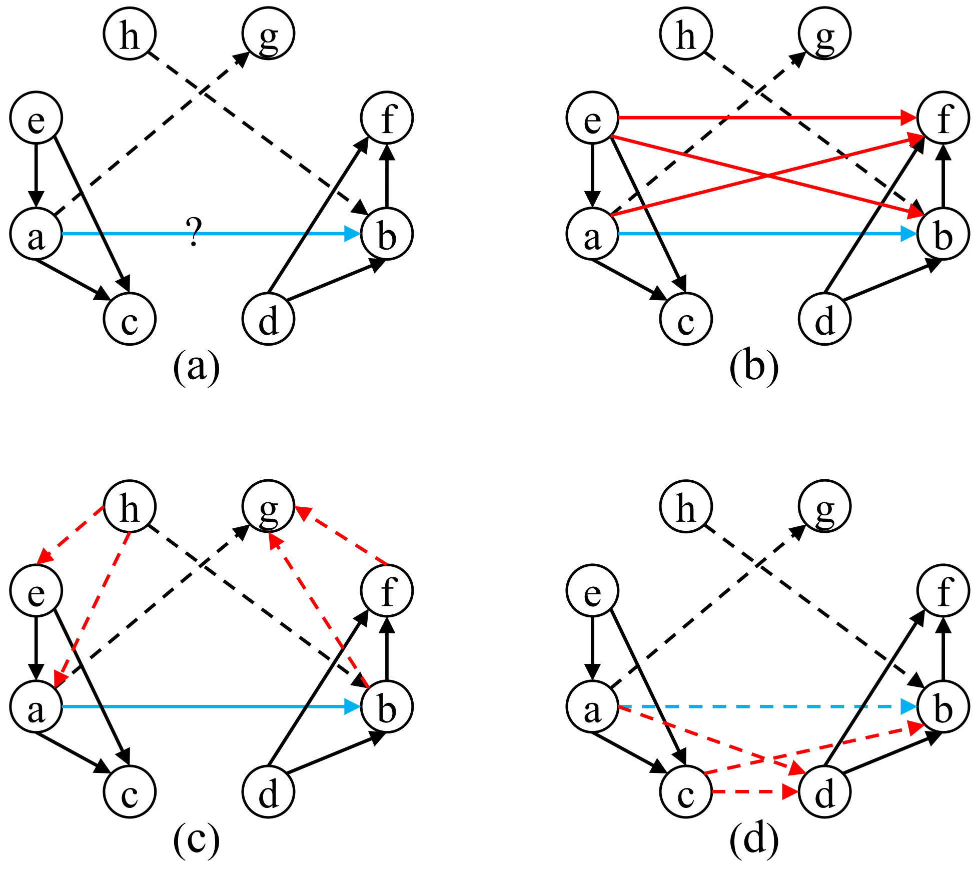


Figure 5.1. Following the notations in Theorem 1: (a) Black lines are pairs in H, solid lines are pairs in G, and dashed lines are pairs in G^c. The pair (a, b) in the cyan color is the pair to be labeled or deduced. (b) If $(a, b) \in G$, $\{(a, b), (e, f), (a, f), (e, b)\} \subseteq N_{(a,b)}$. (c) If $(a, b) \in G$, $\{(h, e), (h, a)\} \subseteq T_{(a,b)}$ and $\{(b, g), (f, g)\} \subseteq S_{(a,b)}$. (d) If $(a, b) \in G^c$, $\{(a, b), (a, d), (c, b), (c, d)\} \subseteq N'_{(a,b)}$. Likewise, if $\exists (x, y) \in G, \text{s.t.} (a, b) \in S_{(x,y)} \cup T_{(x,y)}$, $\{(a, b), (a, d), (c, b)\} \subseteq O_{(x,y)}$.

to efficiently calculate $O_{(x,y)}$ whose worst time complexity is $O(|H|^3)$ (followed by Prop. 3), while others can be done in $O(|H|^2)$.

Proposition 3. *Following the notations in Theorem 1, given $S_{(a,b)} \cup T_{(a,b)}$, the worst time complexity to calculate $O_{(a,b)}$ is $O(|H|^3)$.*

Using Prop. 4, one can show that there exists a pruning rule that cuts a major proportion of redundant set operations in calculating $O_{(a,b)}$.

Proposition 4. *Following the notations in Theorem 1, we have $N''_{(c',d')} \subseteq N''_{(c,d)}$ if $(c', d') \in N''_{(c,d)}$.*

We also conduct empirical studies to examine the growth rate of calculating $C_{(x,y)}$. In practice, we find the empirical growth rate is closer to linear rate, which means the worst time complexity bound presented here is very conservative.

5.3 Active Learning of a Strict Order

Following the same introduction to pool-based active learning as described in Section 4.2, we will start from generalizing Equation 4.1 and show that it is possible to extend Uncertainty Sampling and Query-By-Committee, the top two performing query strategies as shown in Figure 4.1, for considering relational data as a strict order.

Given G a strict order of V, consider a set of data $\mathcal{D} \subseteq V \times V$, where $(a, a) \notin \mathcal{D}, \forall a \in V$. Similar to the pool-based active learning, one needs to maintain a labeled set D_l and an unlabeled set D_u. We require that $\mathcal{D} \subseteq \mathcal{D}_l \cup \mathcal{D}_u$and $\mathcal{D}_l \cap \mathcal{D}_u = \varnothing$. Given a feature extractor $\mathcal{F} : V \times V \mapsto \mathbb{R}^d$, we can build a vector dataset $\{\mathbf{x}_{(a,b)} = \mathcal{F}(a, b) \in \mathbb{R}^d \mid (a, b) \in \mathcal{D}\}$. Let $y_{(a,b)} = -1 + 2 \cdot \mathbf{1}[(a, b) \in G] \in \{-1, 1\}$ be the ground-truth label for each $(a, b) \in V \times V$. Active learning aims to query Q a subset from $\mathcal{D}$ under limited budget and construct a label set $\mathcal{D}_l$ from Q, in order to train a good classifier h on $\mathcal{D}_l \cap \mathcal{D}$ such that it predicts accurately whether or not an unlabeled pair $(a, b) \in G$ by $h(\mathcal{F}(a, b)) \in \{-1, 1\}$.

Active learning of strict orders differs from the traditional active learning in two unique aspects: (i) By querying the label of a single unlabeled instance, one may obtain a set of labeled examples, with the help of strict orders' properties; (ii) The relational information of strict orders could also be utilized by query strategies. We will present our efforts towards incorporating the above two aspects into active learning of a strict order.

5.3.1 Basic Relational Reasoning in Active Learning

A basic extension from standard active learning to one under the strict order setting is to apply relational reasoning when both updating $\mathcal{D}_l$ and predicting labels. Algorithm 2 shows the pseudocode for the pool-based active learning of a

Algorithm 2 Pseudocode for pool-based active learning of a strict order.

```
Input:
  $\mathcal{D} \subseteq V \times V$    % a data set
Initialize:
  $\mathcal{D}_l \leftarrow \{(a_{s_1}, b_{s_1}), (a_{s_2}, b_{s_2}), ..., (a_{s_k}, b_{s_k})\}$    % initial labeled set with $k$ seeds
  $\mathcal{D}_l \leftarrow \overline{\mathcal{D}_l}$    % initial closure
  $\mathcal{D}_u \leftarrow \mathcal{D} \backslash \mathcal{D}_l$    % initial unlabeled set
while $\mathcal{D}_u \neq \emptyset$ do
    Select $(a^*, b^*)$ from $\mathcal{D}_u$    % according to a query strategy
    Query the label $y_{(a^*,b^*)}$ for the selected instance $(a^*, b^*)$
    $\mathcal{D}_l \leftarrow \overline{\mathcal{D}_l \cup \{(a^*, b^*)\}}$
    $\mathcal{D}_u \leftarrow \mathcal{D} \backslash \mathcal{D}_l$
end while
```

strict order. When updating $\mathcal{D}_l$ with a new instance $(a, b) \in \mathcal{D}_u$ whose label $y_{(a,b)}$ is acquired from querying, one first calculates $\overline{\mathcal{D}'_l}$, i.e., the closure of $\mathcal{D}_l \cup \{(a, b)\}$, using Theorem 1, and then sets $\mathcal{D}_l := \overline{\mathcal{D}'_l}$ and $\mathcal{D}_u := D \backslash \overline{\mathcal{D}'_l}$ respectively. Therefore, it is possible to augment the labeled set $\mathcal{D}_l$ with more than one pair at each stage even though only a single instance is queried. Furthermore, the following corollary shows that given a fixed set of samples to be queried, their querying order does not affect the final labeled set D_l constructed.

Corollary 1.1. *Given a list of pairs Q of size m whose elements are from $V \times V$, let $i_1, \ldots, i_m$ and $j_1, \ldots, j_m$ be two different permutations of $1, \ldots, m$. Let $I_0 = \varnothing$ and $J_0 = \varnothing$, and $I_k = \overline{I_{k-1} \cup \{q_{i_k}\}}$, $J_k = \overline{J_{k-1} \cup \{q_{j_k}\}}$ for $k = 1, \ldots, m$, where $\bar{\cdot}$ is defined as the closure set under G. We have $I_m = J_m$, which is the closure of $\{q_i \in V \times V \mid i = 1, \ldots, m\}$.*

Corollary 1.1 is a straightforward result from the uniqueness of closure, which is also verified by our experiments. The labeled set $\mathcal{D}_l$ contains two kinds of pairs based on where their labels come from: The first kind of labels comes directly from queries, and the second kind comes from the relational reasoning as explained by Theorem 1. Such an approach has a clear advantage over standard active learning at the same budget of queries, because labels of part of the test pairs can be inferred deterministically and as a result there will be more labeled data for supervised training. In our setup of active learning, we train classifiers on $\mathcal{D} \cap \mathcal{D}_l$ and use them for predicting the labels of remaining pairs that are not in $\mathcal{D}_l$.

5.3.2 Query Strategies with Relational Reasoning

The relational active learning framework as explained in the previous section however does not consider incorporating relational reasoning in its query strategy. We further develop a systematic approach on how to achieve this.

We start from the following formulation: at each stage, one chooses a pair (a^*, b^*) to query based on

$$(a^*, b^*) = \operatorname*{argmax}_{(a,b)\in\mathcal{D}_u} \min_{y\in\{-1,1\}} F(\mathcal{S}(y_{(a,b)} = y), \mathcal{D}_l), \tag{5.1}$$

$$\mathcal{S}(y_{(a,b)} = y) = (\overline{\mathcal{D}_l \cup \{(a,b)\}} \backslash \mathcal{D}_l) \cap \mathcal{D}. \tag{5.2}$$

Again, F is the scoring function. $\mathcal{S}(y_{(a,b)} = y)$ is the set of pairs in $\mathcal{D}$ whose labels, originally unknown ($\notin \mathcal{D}_l$), can now be inferred by assuming $y_{(a,b)} = y$ using Theorem 1. For each $(u,v) \in \mathcal{S}(y_{(a,b)} = y)$, its inferred label is denoted as $\hat{y}_{(u,v)}$ in the sequel. One can see that this formulation is a generalization of Eq. (4.1). We now proceed to develop extensions for the two query strategies to model the dependencies between pairs imposed by the rule of a strict order. Following the same notations as described in Section 4.2 with the only difference that the numbering index is replaced by the pairwise index, we propose two query strategies tailored to strict orders.

Uncertainty Sampling with Reasoning. With relational reasoning, one not only can reduce the uncertainty of the queried pair (a, b) but also may reduce that of other pairs deduced by assuming $y_{(a,b)}{=}y$. The modified scoring function reads:

$$F(\mathcal{S}(y_{(a,b)} = y), \mathcal{D}_l) = \sum_{(u,v)\in\mathcal{S}(y_{(a,b)}=y)} 1 - P_{\Delta(\mathcal{D}_l\cap\mathcal{D})}(y_{(u,v)} = \hat{y}_{(u,v)} | \mathbf{x}_{(u,v)}). \tag{5.3}$$

Query-by-Committee with Reasoning. Likewise, one also has the extension for QBC, where $\{g^{(k)}\}_{k=1}^{C}$ is a committee of classifiers trained on bagging samples of $\mathcal{D}_l \cap \mathcal{D}$,

$$F(\mathcal{S}(y_{(a,b)} = y), \mathcal{D}_l) = \sum_{(u,v)\in\mathcal{S}(y_{(a,b)}=y)} \textstyle\sum_{k=1}^{C} \mathbf{1}(\hat{y}_{(u,v)} \neq g^{(k)}(\mathbf{x}_{(u,v)})). \tag{5.4}$$

5.3.3 Bounds on the Number of Queries

Note that any strict order G can be described as a directed acyclic graph (DAG). We show the lower and upper bounds on the number of queries that are needed to learn a consistent classifier for G.

Theorem 2. *Given a strict order G of V, let A be a consistent learner that makes m queries to the oracle before termination, then*

$$|\underline{G}| \leq m \leq |\overline{G}| \tag{5.5}$$

where $\underline{G}$ is the transitive reduction [6] of G and $\overline{G}$ is the closure (Def. 9) of G.

The above lower and upper bounds are tight in the sense that there exist DAGs G_1 and G_2, such that $|\underline{G_1}| = |G_1|$ and $|\overline{G_2}| = |G_2|$. With the power of the active learning paradigm, we want to empirically show that the number of queries needed is much smaller.

5.4 Experiments

For evaluation, same as that in Chapter 4, we also apply the proposed active learning algorithms to concept prerequisite learning problem. Experiments are based on the same Wiki concept map dataset [159] and the same feature set described in the previous chapter. We follow the evaluation protocol of active learning as described in Section 4.4.3.1 with the only difference that here we compare the following four query strategies:

- Random: randomly select an instance to query.
- LC: least confident sampling, a widely used uncertainty sampling variant. We use logistic regression to estimate posterior probabilities.
- QBC: query-by-committee algorithm. We apply query-by-bagging [108] and use a committee of three decision trees.
- CNT: a simple baseline query strategy designed to greedily select an instance whose label can potentially infer the most number of unlabeled instances.

Method	**Use reasoning when updating $\mathcal{D}_l$**	**Use reasoning to select the instance to query**	**Use learning to select the instance to query**
Random	✗	✗	✗
LC, QBC	✗	✗	✓
Random-R	✓	✗	✗
LC-R, QBC-R	✓	✗	✓
CNT	✓	✓	✗
LC-R+, QBC-R+	✓	✓	✓

Table 5.1. Summary of compared query strategies.

Following the previous notations, the scoring function for CNT is

$$F(\mathcal{S}(y_{(a,b)} = y), \mathcal{D}_l) = |S(y_{(a,b)} = y)|$$

which is solely based on logical reasoning.

For experiments, we test each query strategy under three settings: (i) Traditional active learning where no relational information is considered. Query strategies under this setting are denoted as Random, LC, and QBC. (ii) Relational active learning where relation reasoning is applied to updating $\mathcal{D}_l$ and predicting labels of $\mathcal{D}_{test}$. Query strategies under this setting are denoted as Random-R, LC-R, and QBC-R. (iii) Besides being applied to updating $\mathcal{D}_l$, relational reasoning is also incorporated in the query strategies. Query strategies under this setting are the baseline method CNT and our proposed extensions of LC and QBC for strict partial orders, denoted as LC-R+ and QBC-R+, respectively. Table 5.1 summarizes the query strategies studied in the experiments.

5.4.1 Experiment Results

5.4.1.1 Effectiveness Study

Figure 5.2 shows the AUC results of different query strategies. For each case, we present the average values and 95% C.I. of repeated 300 trials with different train/test splits. In addition, Figure 5.3 compares the relations between the number of queries and the number of labeled instances across different query strategies.

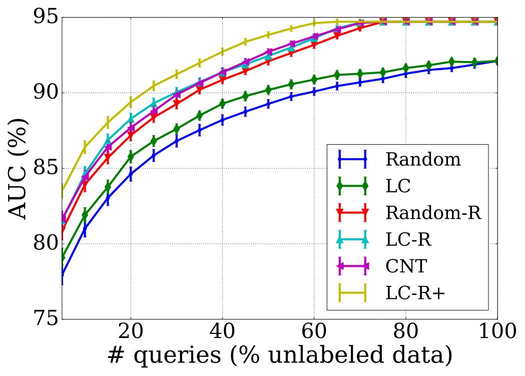
Note that in the relational active learning setting querying a single unlabeled instance will result in one or more labeled instances. According to Figure 5.2 and Figure 5.3, we have the following observations:

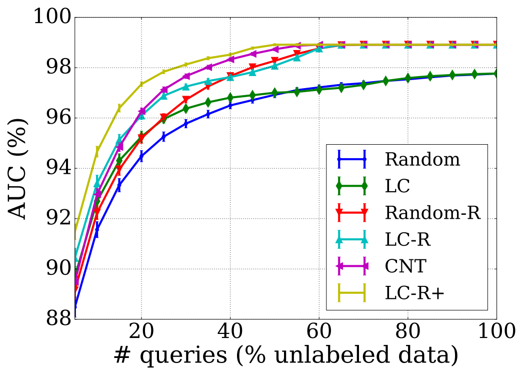
First, by comparing query strategies under the settings (ii) and (iii) with setting (i), we observe that incorporating relational reasoning into active learning substantially improves the AUC performance of each query strategy. In addition, we find the query order, which is supposed to be different for each strategy, does not affect $\mathcal{D}_l$ at the end when $\mathcal{D} \subseteq \mathcal{D}_l$. Thus, it partly verifies Corollary 1.1. Second, our proposed LC-R+ and QBC-R+ significantly outperform other compared query strategies. Specifically, when comparing them with LC-R and QBC-R, we see that incorporating relational reasoning into directing the queries helps to train a better classifier. Figure 5.3 shows that LC-R+ and QBC-R+ lead to more labeled instances when using the same amount of queries than that of LC-R and QBC-R. This partly contributes to the performance gain. Third, LC-R+ and QBC-R+ are more effective at both collecting a larger labeled set and training better classifiers than the CNT baseline. In addition, by comparing CNT with LC-R, QBC-R, and Random-R, we observe that a larger size of the labeled set does not always lead to a better performance. Such observations demonstrate the necessity of combining deterministic relational reasoning and probabilistic machine learning in designing query strategies.

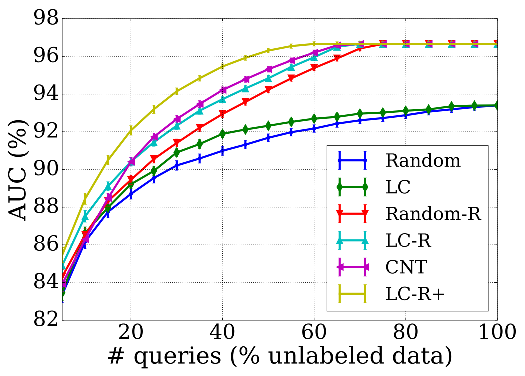
5.4.1.2 Efficiency Study

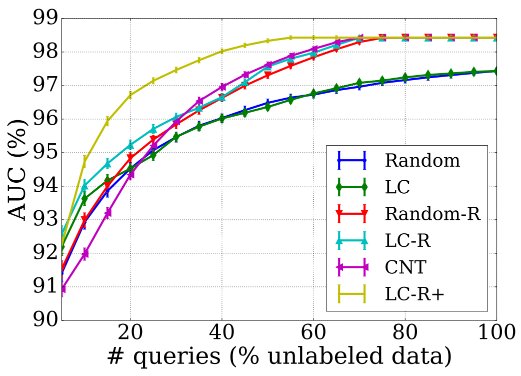
The proposed reasoning module is designed to be plugged into any algorithm that needs reasoning of strict orders. Thus besides verifying effectiveness, it is also important to investigate its efficiency. We conduct empirical studies on the runtime of the reasoning module.

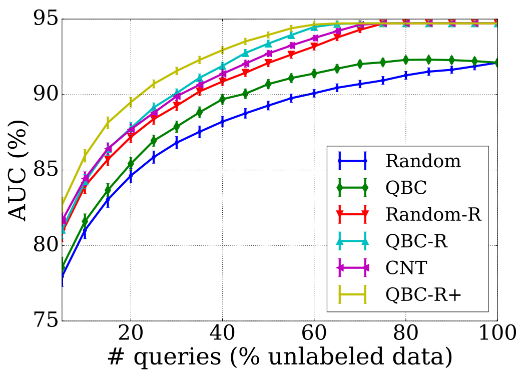
Figure 5.4 shows the relation between the average runtime for calculating the new closure using Theorem 1 and the size of the current labeled closure $|H|$. Results for both LBC-R+ and QBC-R+ are presented. We can see that as $|H|$ keeps increasing during the pool-based active learning process, the average runtime of calculating $C_{(x,y)}$ increases almost linearly and even decreases a little at the end. Although the worst case time complexity for calculating Theorem 1 is $O(|H|^3)$ (for $O_{(a,b)}$) and $O(|H|^2)$ (for others), the runtime required is directly related to the number of ascendants and descendants of elements in V, which is usually different for the four


(a) Data Mining - LC


(b) Geometry - LC


(c) Physics - LC


(d) Precalculus - LC


(e) Data Mining - QBC

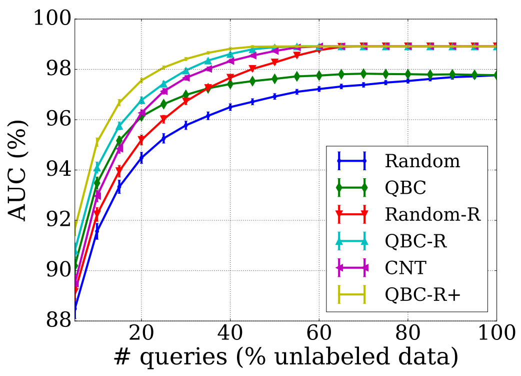
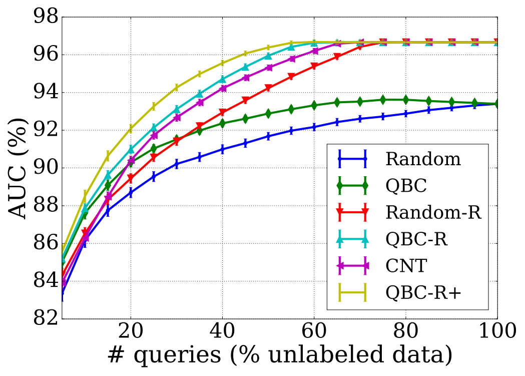
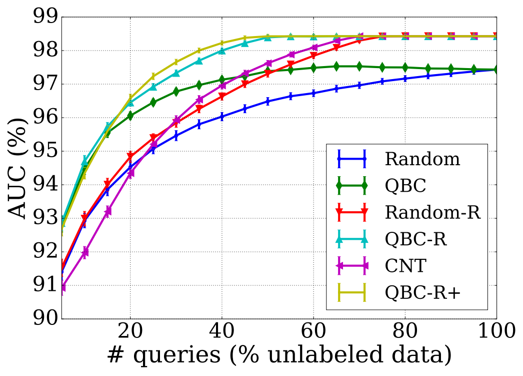

(f) Geometry -QBC

Figure 5.2. Comparison of different query strategies' AUC scores for concept prerequisite learning.

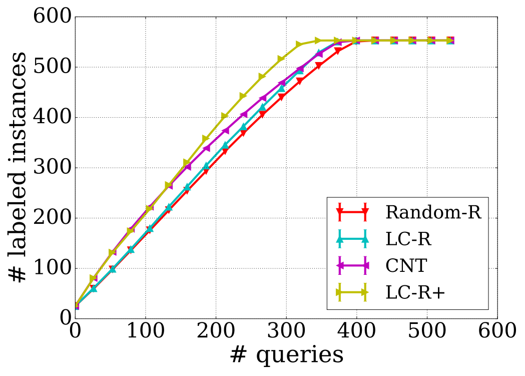


(g) Physics - QBC

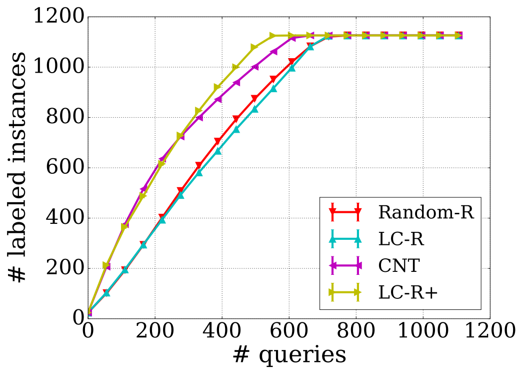


(h) Precalculus - QBC

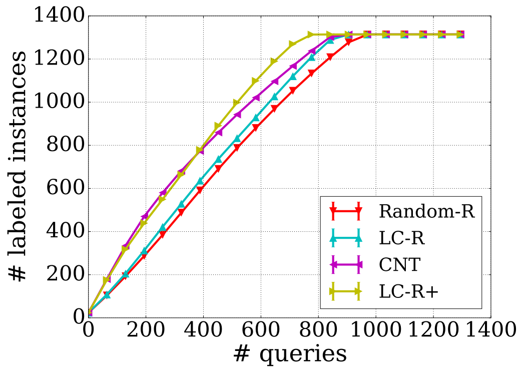
Figure 5.2. Comparison of different query strategies' AUC scores for concept prerequisite learning. (cont.)



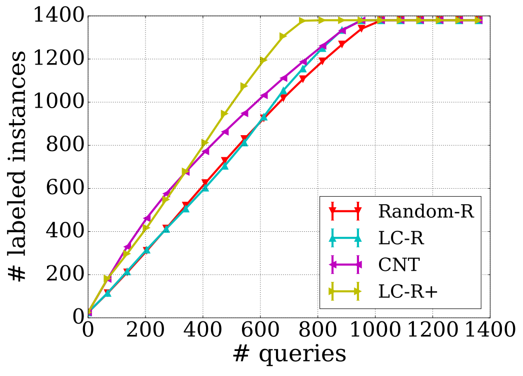
(a) Data Mining - LC



(b) Geometry - LC

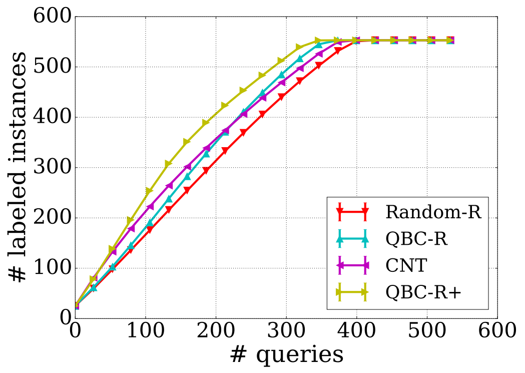


(c) Physics - LC

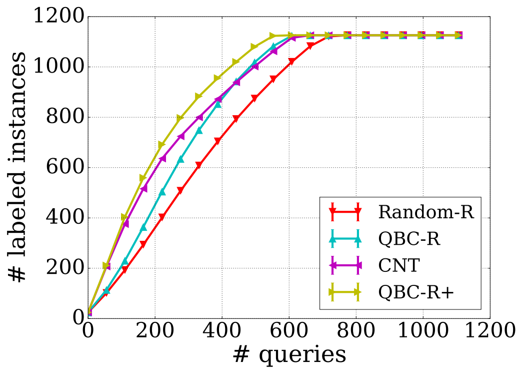


(d) Precalculus - LC

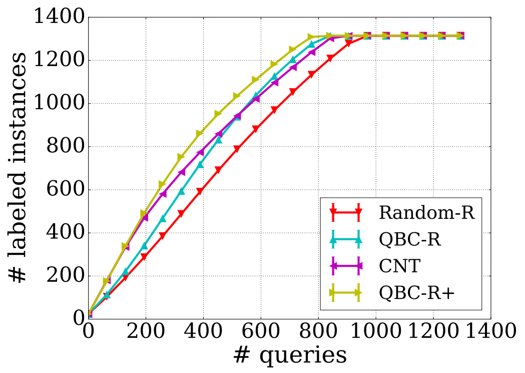
Figure 5.3. Comparison of relations between the number of queries and the number of labeled instances when using different query strategies.



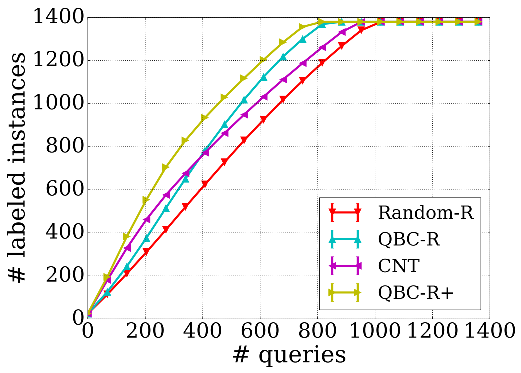
(e) Data Mining - QBC



(f) Geometry - QBC



(g) Physics - QBC

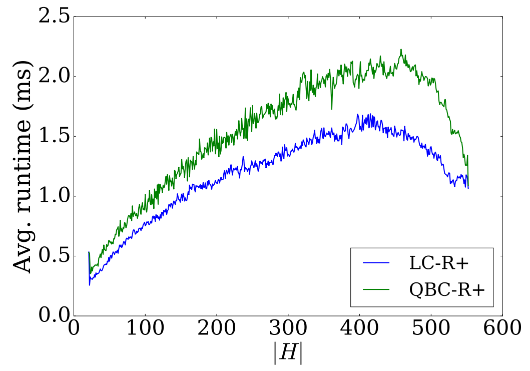


(h) Precalculus - QBC

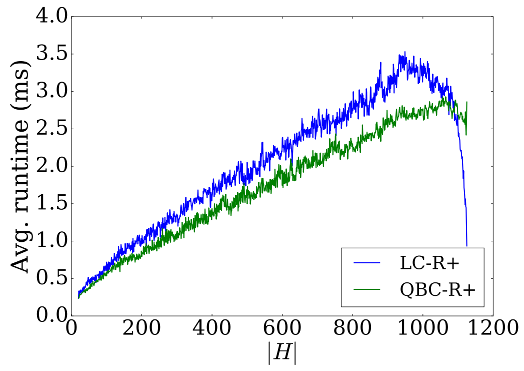
Figure 5.3. Comparison of relations between the number of queries and the number of labeled instances when using different query strategies. (cont.)

strict order datasets used. If the few ascendants and descendants effectively control the size of calculations as we have observed, the runtime will be short regardless of a large $|H|$. This might explain why the growth of calculating $C_{(x,y)}$ is near linear.

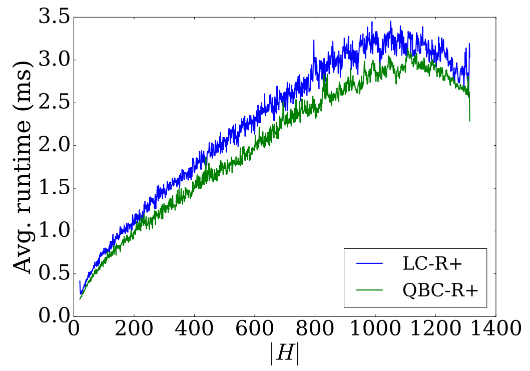
We also empirically evaluate the effects of using Proposition 4 on the efficiency. Specifically, we measure the total runtime of $O_{(a,b)}$ calculation in Theorem 1 for a full round of active learning (until $\mathcal{D}_u = \emptyset$) with and without applying the pruning rule induced by Proposition 4. The results are shown in Table 5.2 where the numbers are the average runtime over all 300 different rounds of active learning for each dataset. We can see that the pruning can lead to a speedup of 40-55% in the LC-R+ experiments and a speedup of 49-55% in the QBC-R+ experiments, which shows that Proposition 4 is helpful for higher efficiency.



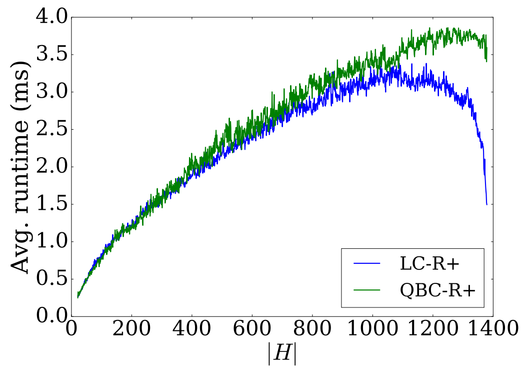
(a) Data Mining



(b) Geometry



(c) Physics



(d) Precalculus

Figure 5.4. The average runtime for calculating the new closure using Theorem 1 v.s. the size of current labeled closure $|H|$.

5.5 Related Work

In general, there have been various types of strategies for active learning. The most popular active learning approach is to query the most informative instances. Typical algorithms in this category include uncertainty sampling [10, 86, 87, 152], query-by-committee [36, 45, 146], mutual information based sampling [55, 56], and expected error reduction sampling [141]. Another line of active learning approaches select instances based on how representative the unlabeled instances are, where clustering method is usually applied to measure representativeness [24, 37, 121]. In addition, there are also query strategies that select instances based on the combination of informativeness and representativeness [41, 68, 160, 168].

As for active learning of relational data, there have been research efforts towards

Domain	**LC-R+**		**QBC-R+**	
	w/o pruning	**w/ pruning**	**w/o pruning**	**w/ pruning**
Data mining	5.5	3.3 (-40%)	5.5	2.6 (-53%)
Geometry	85.5	39.2 (-54%)	69.6	31.4 (-55%)
Physics	111.0	58.8 (-47%)	117.1	60.2 (-49%)
Precalculus	134.3	59.9 (-55%)	167.4	74.8 (-55%)

Table 5.2. The effect of Proposition 4 on the total runtime (s) of $O_{(a,b)}$ calculation in Theorem 1 for a full round of active learning (until $\mathcal{D}_u = \emptyset$).

applying active learning to "learning to rank" [23, 29, 39, 40, 103]. However, to our knowledge, there is no existing query strategy that is designed for strict partial orders. Often, learning to rank relies on binary classifiers or probabilistic models that are consistent to the rule of a total order. Active learning methods for ranking are not suitable for partial orders, since a classifier consistent with a total order will always lead to a non-zero lower bound of error rate given the groundtruth as a partial order but not a total order. One related active learning work that deals with a similar setting to ours is [138], whereas equivalence relations are considered instead. Particularly, they made several crude approximations in order to expedite the example selection process to a computational tractable level. We approach the design of query strategies from a quite different perspective while keeping efficiency as one of our central concerns.

5.6 Summary and Discussion

This chapter proposes an active learning framework tailored to relational data in the form of strict partial orders. An efficient reasoning module is proposed to extend two commonly used query strategies – uncertainty sampling and query by committee. Experiments on concept prerequisite learning show that incorporating relational reasoning in both selecting valuable examples to label and expanding the training set significantly improves standard active learning approaches. Future work could be to explore the following: (i) apply the reasoning module to extend other query strategies; (ii) test the proposed algorithm on other types of strict partial orders in which active learning is important; (iii) active learning of strict partial orders from a noisy oracle.

Chapters 2-5 have introduced research efforts on machine learning methods for concept prerequisite learning. Compared with other well-studied relations such as “is-a”, the research on concept prerequisite relations is at an early stage. The proposed active learning research will help deal with the current situation where the number of concept prerequisite labels is very limited. As more researchers work on this topic and contribute to the datasets, there will be more opportunities for developing effective data-driven learning methods, even for neural-net based methods. The ultimate goal is to have a scalable learning method which can accurately measure the prerequisite relation among concepts. Next chapter will switch to another educational application and propose supervised learning methods for automatic distractor generation for multiple choice questions.

Chapter 6 Automatic Distractor Generation for Multiple Choice Questions

6.1 Introduction

Multiple choice questions (MCQs) are widely used as an assessment of students' knowledge and skills. A MCQ consists of three elements: (i) *stem*, the question sentence; (ii) *key*, the correct answer; (iii) *distractors*, alternative answers used to distract students from the correct answer. Among all methods for creating good MCQs, finding reasonable distractors is crucial and usually the most time-consuming. We here investigate automatic *distractor generation* (DG), i.e., generating distractors given the stem and the key to the question. We focus on the case where distractors are not limited to single words and can be phrases and sentences.

Rather than generate trivial wrong answers, the goal of DG is to generate plausible false answers - good distractors. Specifically, a "good" distractor should be at least semantically related to the key [52], grammatically correct given the stem, and consistent with the semantic context of the stem. Taking these criterion into consideration, most existing methods for DG are based on various similarity measures. These include WordNet-based metrics [118], embedding-based similarities [54, 70, 80], n-gram co-occurrence likelihood [63], phonetic and morphological similarities [130], structural similarities in an ontology [149], a thesaurus [150], context similarity [131], context-sensitive inference [171], and syntactic similarity [25]. Then distractors are selected from a candidate distractor set based on a weighted combination of similarities, where the weights are determined by heuristics.

In contrast to above-mentioned similarity-based methods, this chapter explores machine learning based methods to solve DG. The chapter will start with a brief literature review (Section 6.2) on automatic question generation and distractor generation. Section 6.3 proposes a generative model leanred from training generative adversarial nets [51] to create useful distractors for automatically creating fill-in-the-blank (FITB) multiple choice questions. Section 6.4 investigates how machine learning models, specifically ranking models, can be used to select useful distractors for multiple choice questions.

6.2 Related Work

6.2.1 Automatic Question Generation

Dating back to the 1970s when Autoquest system [166] was proposed, there has been a considerable amount of prior work on automatic question generation for educational purposes. Most existing work on automatic question generation can be categorized into two families: (i) **Wh-question generation** [27, 35, 62, 118] and (ii) **fill-in-the-blank question generation** (FITB-QG) [2, 11, 54, 63, 80, 84, 97, 131, 150, 165].

The research on Wh-question generation focuses on syntax-based transformation and aims to change syntactic structures of sentences to convert them into interrogative sentences as questions. The work in this area mainly emphasizes the grammaticality of the generated questions. Typical approaches for generating Wh-questions include syntactic transformation rules [118], template/pattern-based generation [27, 35], overgenerating transformations and ranking [61, 62], etc. These methods relies heavily on the rule-based syntactic transformation of a declarative sentence. Only recently there have been few studies [42, 173] that are fully data-driven and do not rely on manually generated rules.

Unlike Wh-question generation that focuses on syntax-based transformation, FITB-QG avoids the grammaticality issue by blanking out existing one or more words from a given good sentence. Early work in FITB-QG mainly focused on English language learning, with applications including evaluating students' vocabulary [130] and testing knowledge of using verbs [150], adjectives [97], and prepositions [84]. Recently, FITB-QG generated exercise questions as multiple-choice quizzes for one or multiple subjects [2, 11, 54, 80]. Our method aligns more closely to FITB-QG for general knowledge assessment.

6.2.2 Distractor Generation

For the DG problem, many systems utilize *WordNet* [113] to find synonyms or other related words as distractors [119]. Although WordNet has 117,000 synsets, its coverage is limited when compared to general knowledge bases such as Wikipedia. Other work has explored using the link structure of ontologies [149], which usually requires a pre-existing domain-specific ontology. Other methods choose distractors from sentences in a constrained set of source texts [2, 74]. Our methods do not have such constraints and can select distractors from the entire Wikipedia or any pre-defined candidate distractor set. Others also investigate various similarity metrics for DG, including embedding-based similarities [54, 70, 80], n-gram co-occurrence likelihood [63], phonetic and morphological similarities [130], structural similarities in an ontology [149], a thesaurus [150], context similarity [131], context-sensitive inference [171], and syntactic similarity [25]. Our proposed methods are fundamentally different from these unsupervised similarity-based methods in that the training is supervised.

6.3 Distractor Generation with Generative Adversarial Nets for Automatically Creating Fill-in-the-blank Questions

6.3.1 Introduction

Different from existing approaches which heavily depend on the key, we propose to learn distractor distribution conditioned on the stem, since the semantic information conveyed by the stem is also critical to generate "good" distractors. Specifically, we adapt generative adversarial nets (GANs) [51] to tackle DG. We simultaneously train two models: a generative model G that captures real data (corresponding to the key to the FITB question) distribution given a context (corresponding to the stem), and a discriminative model D that estimates the probability that a sample comes from the real training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. Distractors can be generated according to the distribution estimated by G.

The proposed GAN model is trained on the Wikipedia corpus and is able to predict Wiki entities as distractors. We evaluate the proposed method on two biology question datasets: (i) Wiki-FITB where 30 sentences from Wikipedia are selected and transformed into FITB questions; (ii) Course-FITB where 92 FITB questions are selected from actual college-level exams. Our method is compared with a widely used word2vec-based method. For each question, a list of distractors is generated and evaluated by domain experts. Given predictions of the two methods, we propose to apply a second-stage learner to utilize information in both the stem and the key. This outperforms both the proposed GAN-based method and the word2vec-based method on two datasets, with 51.7% and 48.4% of distractors generated being acceptable.

Our major contribution is summarized as follows.

- The proposed machine learning-based approach is fundamentally different from previous unsupervised similarity-based approaches and it is the first application of GANs to DG.

- The proposed method only uses stem information and it can be used in combination with existing key-based methods for generating distractors that better fit question context.

- The college-level exam FITB question set can be used for evaluating distractor generation or general FITB-QG.

6.3.2 Methods

6.3.2.1 Generative Adversarial Nets

Proposed by Goodfellow et al. [51], generative adversarial nets are a novel approach to train a generative model. The key to GANs are two "adversarial" models: the Generator G and the Discriminator D. G is a generative model that aims to capture real data distribution $p_{data}(\mathbf{x})$. D is a discriminative model that estimates the probability that a sample came from the real training data rather than G.

Both G and D could be a non-linear mapping function. To learn the generator's distribution p_g over data $\mathbf{x}$, G parameterized by θ_g maps a prior noise distribution $p_z(\mathbf{z})$ to the data space as $G(\mathbf{z}; \theta_g)$. On the other hand, the discriminator $D(\mathbf{x}; \theta_d)$ parameterized by θ_d outputs a single scalar representing the probability that a sample $\mathbf{x}$ came from training data rather than p_g.

D is trained to maximize the probability of assigning the correct label to both training examples and samples from G. Simultaneously G is trained to maximize the probability of D making a mistake, i.e., minimizing $\log(1 - D(G(\mathbf{z})))$. The whole training procedure for G and D follows a two-player minimax game:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{6.1}$$

6.3.2.2 Conditional GANs for DG

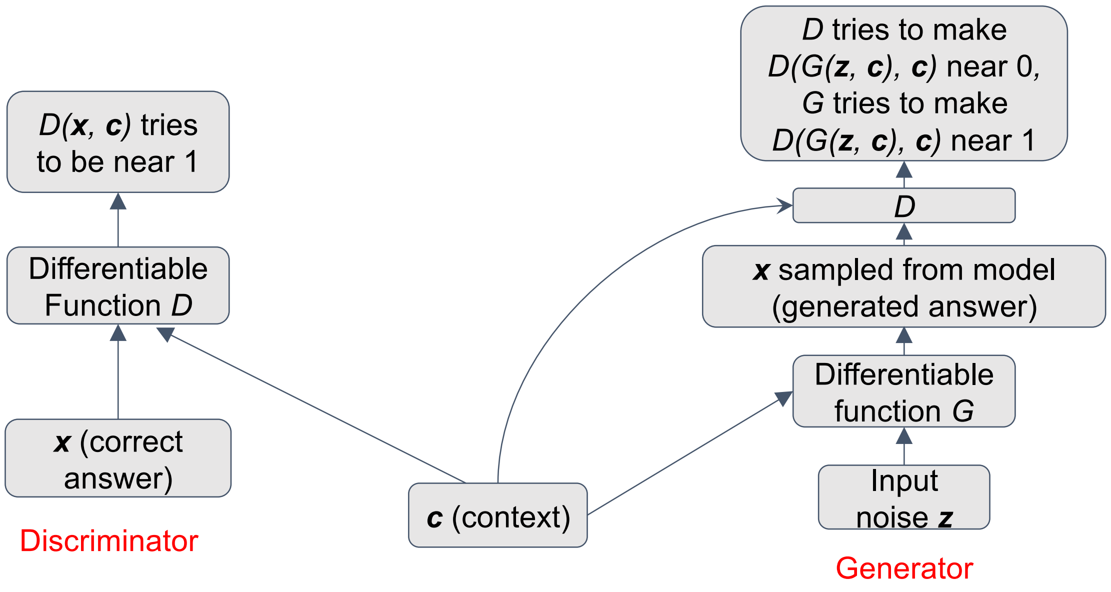
Since our goal is to generate distractors given a question sentence, we therefore adapt a GAN to distractor generation such that both the generator and the discriminator are conditioned on the extra context $\mathbf{c}$ learned from the question sentence (see Section 6.3.2.3). For this we put $\mathbf{c}$ into both the discriminator and generator as additional input. More precisely in the generator, the combination of a noise vector $\mathbf{z}$ and $\mathbf{c}$ is taken as a joint input, while in the discriminator, both the generated sample $\mathbf{x}$ and $\mathbf{c}$ are utilized to determine whether $\mathbf{x}$ came from training data. As a consequence, the minimax game in Equation 6.1 can be rewritten as:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x}|\mathbf{c})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z}|\mathbf{c})))] \tag{6.2}$$

which is the conditional GAN proposed by [117].

The neural-network based training framework provides additional flexibility on how the input vectors are combined. In the generator, we simply concatenate both $\mathbf{z}$ and $\mathbf{c}$ to build another vector representation: $\tilde{\mathbf{z}} = \mathbf{z} \odot \mathbf{c}$ where $\odot$ represents the concatenation. In the discriminator, we concatenate the linear transformation of the generated sample $\mathbf{x}$ and $\mathbf{c}$: $\tilde{\mathbf{x}} = (W_x\mathbf{x}) \odot \mathbf{c}$, where W_x is a weight matrix to be learned. $\tilde{\mathbf{x}}$ is then fed to a multi-layer perceptron.

GANs have a serious limitation requiring that the composition of the generator and the discriminator are fully differentiable. This is not true for discrete variables such as tokens in the text. Since the generator has an output softmax layer which



D tries to make D(G(z, c), c) near 0, G tries to make D(G(z, c), c) near 1
D(x, c) tries to be near 1
D
Differentiable Function D
x sampled from model (generated answer)
Differentiable function G
x (correct answer)
c (context)
Input noise z
Discriminator
Generator

Figure 6.1. Conditional GANs for Distractor Generation.

can be interpreted as the probability of yielding each token, we sample a discrete token value from the distribution. As such, the back-propagation algorithm alone cannot provide a valid training signal for the generator since the sampling operation is not differentiable.

For this a number of approaches have been proposed, including policy gradient [170] and Gumbel softmax trick [69, 106]. We adopt the Gumbel softmax method since the policy gradient method involves the design of a reward function, which can lead to training instability. While prior work on Gumbel softmax trick were on datasets with a small number of classes (e.g. 10 classes for MNIST dataset), here we investigate its effectiveness on datasets with orders of magnitude more classes.

Similar to the re-parameterization trick in variational auto-encoder (VAE) [77], we efficiently draw samples $\mathbf{x}$ from a categorical distribution with class probabilities $\boldsymbol{\pi}$ using the Gumbel-Max trick [107]:

$$\mathbf{x} = \text{one\_hot}(\underset{i}{\text{argmax}}[g_i + \log \pi_i]) \approx \text{softmax}(\mathbf{g} + \log \boldsymbol{\pi}) \tag{6.3}$$

where g_i $(i = 1, 2, \cdots, K)$ is an i.i.d sample drawn from Gumbel$(0, 1)$ distribution and K is the total number of classes. The softmax function is used as a continuous, differentiable approximation to argmax, generating a K-dimensional sample vector

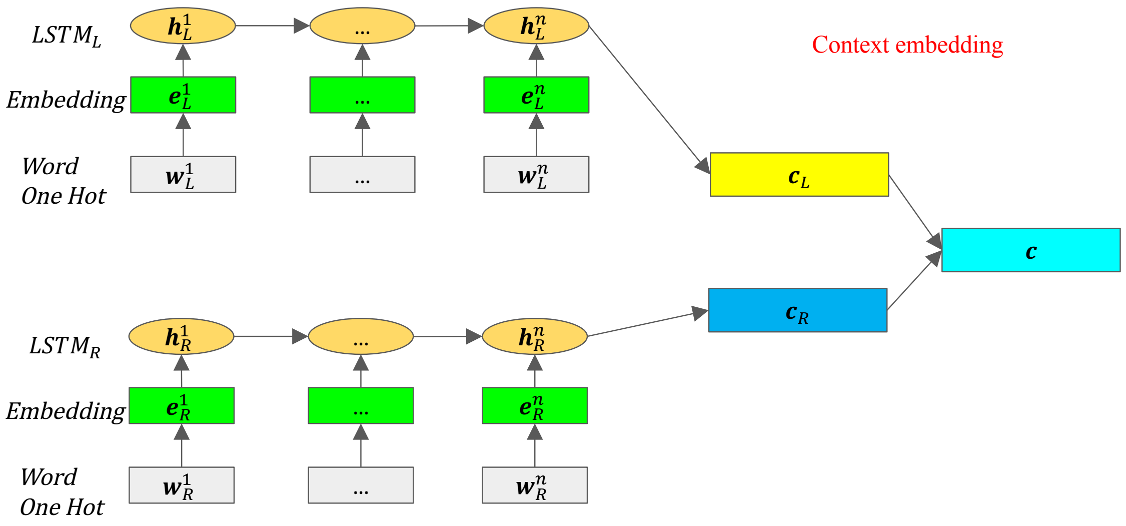


Figure 6.2. Context Modeling of the Question Sentence

$\mathbf{x} \in \mathbb{R}^{K-1}$ where

$$x_i = \frac{\exp((g_i + \log \pi_i)/\tau)}{\sum_{j=1}^{K} \exp((g_j + \log \pi_j)/\tau)} \tag{6.4}$$

with τ being a temperature hyper-parameter.

Since the Gumbel-Max trick enables back-propagating training signals from the discriminator to the generator, we can use the minimax optimization for discrete variables.

Figure 6.1 presents the general architecture of the proposed conditional GAN for DG, where **c** is the learned context vector representation of the question stem, **x** the vector representation of the correct answer or the generated sample, and **z** a noise vector sampled from a prior noise distribution. G represents the generator, and D represents the discriminator.

6.3.2.3 Context Modeling of the Question Sentence

The architecture proposed in Figure 6.1 requires a context vector **c** of the question stem. To model the context, we use long short-term memory (LSTM) [64]. Specifically, since each question sentence could be divided into a left (s_L) and a right (s_R) part by the position of the blank, two LSTMs (LSTM$_L$ and LSTM$_R$) are used to model the left context ($\mathbf{c}_L$) and the right context ($\mathbf{c}_R$) separately. The context vector $\mathbf{c}$ is therefore a concatenation of $\mathbf{c}_L$ and $\mathbf{c}_R$. Figure 6.2 presents the model architecture. Consider the question sentence "*A* _____ *is a networking device that forwards data packets between computer networks.*". The question is first split into two parts:"*A*" and "*is a networking device that forwards data packets between computer networks.*" The left and right part is then fed into LSTM$_L$ and LSTM$_R$ respectively, resulting in $\mathbf{c}_L$ and $\mathbf{c}_R$. Next we concatenate $\mathbf{c}_L$ and $\mathbf{c}_R$ to obtain the context vector $\mathbf{c}$. In practice, we reverse the order of the words in the right part s_R when feeding it to LSTM$_R$, in order to emphasize neighboring words around the blank.

6.3.2.4 Implementation Details

The generator and the discriminator is implemented by a 2-layer perceptron with a hidden size of 350. The network utilizes Leaky ReLU [105] activation. We set the noise vector $\mathbf{z}$ as a 50-dimensional vector drawn from $\mathcal{N}(0,1)$ and each LSTM to have a hidden size of 150. During training, Adam algorithm [75] with a learning rate of 0.001 is used. The temperature τ is fixed as 1.

6.3.3 Experiments

6.3.3.1 Data Preparation

Training the proposed conditional GAN model requires a large number of (stem, key) pairs. We propose to utilize the Wikipedia corpus for creating the training set. To remove part of the sentence as a blank, we exploit the link structure among different Wiki pages. Specifically, we substitute the link in a sentence with a blank to get the stem and use the linked Wiki concept[1] as the key. Thus sentences with links could be transformed into (stem, key) pairs. For example, consider a sentence from Wikipedia *Machine learning is the subfield of computer science that, according to Arthur Samuel in 1959, gives "computers the ability to learn without being explicitly programmed."* It is transformed into a tuple (*Machine learning is the subfield of* _____ *that, according to Arthur Samuel in 1959, gives "computers the ability to*

[1]Each concept corresponds to an English Wiki article.

learn without being explicitly programmed.”, Computer_science), representing the question sentence and correct answer. Note that DG model usually is tailored to a certain domain, such as physics, biology, mathematics, etc. For each domain, we select a subset of all created pairs as training data based on whether the key appears in the domain-specific concept vocabulary. Such vocabulary is built by breadth-first searches starting from several main concepts of the domain and filtering with several semantic similarities such as LDA [14], word2vec, etc.

Our experiments here focus on biology. With the procedure described above, we build a vocabulary with 8879 biology-related concepts and create a training set with 1.62 million (stem, key) pairs. In addition, we create two test sets for evaluation: (i) **Wiki-FITB**: 30 FITB questions based on sentences in Wikipedia, selected by a domain expert; (ii) **Course-FITB**: 92 FITB questions from actual exams for two college-level biology courses and GRE (biology subject) 2016.

6.3.3.2 Experiment Settings

For each (stem, key) pair, we apply the proposed FITB-QG method to generate a list of distractors. Three domain experts with teaching experience, a Ph.D. in biology, a Ph.D. candidate in biology and a Ph.D. candidate in entomology were then asked collaboratively to label each of the top-4 predictions as a *Good*, *Fair*, or *Bad* distractor.

We compare the proposed GAN-based FITB-QG model (**GAN**) with a frequently used similarity-based method (**W2V**), which generates distractors based on the word2vec similarity between the candidate and the key. We trained a word2vec model on the Wiki corpus with each concept treated as an individual token.

Since a GAN only utilizes information in the *stem* part while W2V only utilizes information in the *key* part, we additionally apply *a second-stage learner* (**GAN+W2V**) to combine the strengths of GAN and W2V. For each (stem, key, distractor) tuple, we use the prediction score and the ranking of GAN and W2V as four features, and train a logistic regression classifier to predict the probability of a distractor being good, fair, or bad. The final distractor predictions are ranked by the probability of being bad estimated by the second-stage learner.

Dataset	Methods	Good	Fair	Bad
Wiki-FITB	GAN	28.4 (10.8)	10.8 (5.5)	60.8 (10.5)
	W2V	35.8 (6.8)	10.0 (5.0)	54.2 (8.0)
	GAN + W2V	40.0 (7.8)	11.7 (5.0)	48.3 (8.6)
Course-FITB	GAN	17.7 (5.0)	9.2 (3.5)	73.1 (5.9)
	W2V	32.9 (5.3)	11.9 (3.5)	55.2 (5.7)
	GAN + W2V	34.3 (5.7)	14.1 (3.9)	51.6 (6.0)

Table 6.1. Distractor generation results. Numbers are 95% confidence intervals of percentages of generated distractors being good, fair, or bad, calculated in a leave-one-out manner.

GAN	W2V	GAN + W2V
Speciation	Natural selection	Speciation
Transcription	Macroevolution	Natural selection
Inbreeding	Evolutionary biology	Microevolution
Genetic drift	Microevolution	Genetic drift

Table 6.2. Distractor generation examples for question "Changes in gene frequency over time describes the process of ______.", whose key is *Evolution*. (Legend: Good, Fair, Bad)

.

6.3.3.3 Experimental Results

The distractor generation results on two datasets are shown in Table 6.1. We evaluate each method in a leave-one-out manner and report the 95% confidence intervals of percentages of generated distractors being good, fair, or bad.

When comparing GAN with W2V, we can see that they achieve comparable performance on Wiki-FITB and that W2V is significantly better than GAN on Course-FITB. Since the GAN is based on the question stem, its distractor generation process is solely dependent on the learned association between the context information and distractors. Course-FITB, collected from actual college exams, is a more challenging dataset because its writing style is different from the part of Wikipedia on which the GAN is trained. Such difference makes it difficult for the GAN to apply the association learned from Wikipedia to questions in Course-FITB. By design distractors are often semantically related to the key (e.g. DNA and RNA). As such similarity-based methods like W2V can provide a strong baseline since they explicitly utilize information about the key. W2V is limited in that it can only output the same distractors for a key regardless of question stems being different. Since question stems sharing the same keys may still emphasize on different aspects, it is desirable to generate diverse distractors for each specific question stem. As such context-based methods such as GAN are still valuable.

We can see that GAN + W2V reduces the mean percentages of badly generated distractors, compared to both GAN and W2V. The percentages of "acceptable" (good + fair) distractors are 51.7% for Wiki-FITB and 48.4% for Course-FITB. Although the difference is not significant given the small sizes of test set, the proposed second-stage learner provides an initial attempt to combine the strengths of GAN and W2V. Such learning-based method is a more systematic way than the ad-hoc weighted combination of different predictions.

Table 6.2 shows an example of generated distractors for the question "Changes in gene frequency over time describes the process of ___." We can see that GAN and W2V generate very different distractors. Since GAN is based on context, its predictions are related to "gene" in the stem. As for W2V, the key "Evolution" is utilized to retrieve similar concepts. In addition, we observe that GAN + W2V's predictions are a mix of GAN's and W2V's results, which reduces the percentages of bad distractors and makes the distractors more diverse.

6.3.4 Summary

This section applied conditional GANs for distraction generation for FITB problems which to our knowledge is the first use of GANs for this problem. Experiments on two collected biology question sets showed that the proposed context-based method is a valuable complement to previous similarity-based methods and that a second-stage learner can be applied to combining the strengths of two types of DG methods in order to achieve a better performance. Such methods should significantly help instructors create better FITB questions. Future work could be to explore: (i) a unified GAN model which can include key information; (ii) better context modeling to improve model generalization.

6.4 Distractor Generation for Multiple Choice Questions Using Learning to Rank

6.4.1 Introduction

In contrast to the previous similarity-based methods, we apply learning-based ranking models to select distractors that resemble those in actual exam MCQs. Specifically, we propose two types of models for DG: feature-based and NN-based models. Our models are able to take existing heuristics as features and learn from these questions a function beyond a simple linear combination. Learning to generate distractors has been previously explored in a few studies. Given a blanked question, Sakaguchi et al. [142] use a discriminative model to predict distractors and Liang et al. [92] apply generative adversarial nets. They view DG as a multi-class classification problem and use answers as output labels while we use them as input. Other related work [162] uses a random forest. However, with the reported binary classification metrics, the quality of the top generated distractors is not quantitatively evaluated. Here we conduct a more comprehensive study on various learning models and devise ranking evaluation metrics for DG.

Machine learning of a robust model usually requires large-scale training data. However, to the best of our knowledge, there is no benchmark dataset for DG, which makes it difficult to directly compare methods. Prior methods were evaluated on different question sets collected from textbooks [2], Wikipedia [92], ESL corpuses [142], etc. We propose to evaluate DG methods with two datasets: the recently released SciQ dataset [162] (13.7K MCQs) and the MCQL dataset (7.1K MCQs) that we made. These two datasets can be used as benchmarks for training and testing DG models. Our experimental results show that feature-based ensemble learning methods (random forest and LambdaMART) outperform both the NN-based method and unsupervised baselines for DG.

6.4.2 Learning to Rank for Distractor Generation

We solve DG as the following ranking problem:

Definition 11 (Distractor Ranking Problem). *Given a candidate distractor set $\mathcal{D}$ and a MCQ dataset $\mathcal{M} = \{(q_i, a_i, \{d_{i1}, ..., d_{ik}\})\}_{i=1}^{N}$, where q_i is the question stem,*

a_i is the key, $D_i = \{d_{i1}...d_{ik}\} \subseteq \mathcal{D}$ are the distractors associated with q_i and a_i, find a point-wise ranking function r: $(q_i, a_i, d) \to [0, 1]$ for $d \in \mathcal{D}$, such that distractors in D_i are ranked higher than those in $\mathcal{D} - D_i$.

This problem formulation is similar to "learning to rank" [102] in information retrieval. To learn the ranking function, we investigate two types of models: feature-based models and NN-based models.

6.4.2.1 Feature-based Models

6.4.2.1.1 Feature Description Given a tuple (q, a, d), a feature-based model first transforms it to a feature vector $\phi(q, a, d) \in \mathbb{R}^d$ with the function ϕ. We design the following features for DG, resulting in a 26-dimension feature vector:

- *Emb Sim.* Embedding similarity between q and d and the similarity between a and d. We use the average GloVe embedding [129] as the sentence embedding. Embeddings have been shown to be effective for finding semantically similar distractors [54, 80].

- *POS Sim.* Jaccard similarity between a and d's POS tags. The intuition is that ditractors might also be noun phrases if the key is a noun phrase.

- *ED.* Edit distance between a and d. This measures the spelling similarity and is useful for cases such as selecting "RNA" as a distractor for "DNA".

- *Token Sim.* Jaccard similarities between q and d's tokens, a and d's tokens, and q and a's tokens. This feature is motivated by the observation that distractors might share tokens with the key.

- *Length.* a and d's character and token lengths and the difference of lengths. This feature is designed to explore whether distractors and the key are similar in terms of lengths.

- *Suffix.* The absolute and relative length of a and d's longest common suffix. The key and distractors often have common suffixes. For example, "maltose", "lactose", and "suctose" could be good distractors for "fructose".

- *Freq.* Average word frequency in a and d. Word frequency has been used as a proxy for words' difficulty levels [31]. This feature is designed to select distractors with a similar difficulty level as the key.
- *Single.* Singular/plural consistency of a and d. This checks the consistency of singular vs. plural usage, which will select grammatically correct distractors given the stem.
- *Num.* Whether numbers appear in a and d. This feature will cover cases where distractors and keys contain numbers, such as "90 degree", "one year", "2018", etc.
- *Wiki Sim.* If a and d are Wikipedia entities, we calculate their Wiki embedding similarity. The embedding is trained using word2vec [112] on Wikipedia data with each Wiki entity treated as an individual token. This feature is a complement to Emb Sim where sentence embedding is a simple average of word embeddings.

6.4.2.1.2 Classifiers We study the following three feature-based classifiers: (i) Logistic Regression: an efficient generalized linear classification model; (ii) Random Forest [18]: an effective ensemble classification model; (iii) LambdaMART [21]: a gradient boosted tree based learning-to-rank model. To train these models, following previous notations, we use D_i as positive examples and sample from $\mathcal{D} - D_i$ to get negative examples.

6.4.2.2 NN-based Models

Based on the recently proposed method IRGAN [157], we propose an adversarial training framework for DG. Our framework consists of two components: a generator G and a discriminator D. G is a generative model that aims to capture the conditional probability of generating distractors given stems and answers $P(d|q,a)$. D is a discriminative model that estimates the probability that a distractor sample comes from the real training data rather than G. Figure 6.3 presents the proposed adversarial training architecture for DG.

Assume that the discriminator is based on an arbitrary scoring function $f_\phi(d,q,a) \in \mathbb{R}$ parameterized by ϕ, then the objective for D is to maximize the following log-

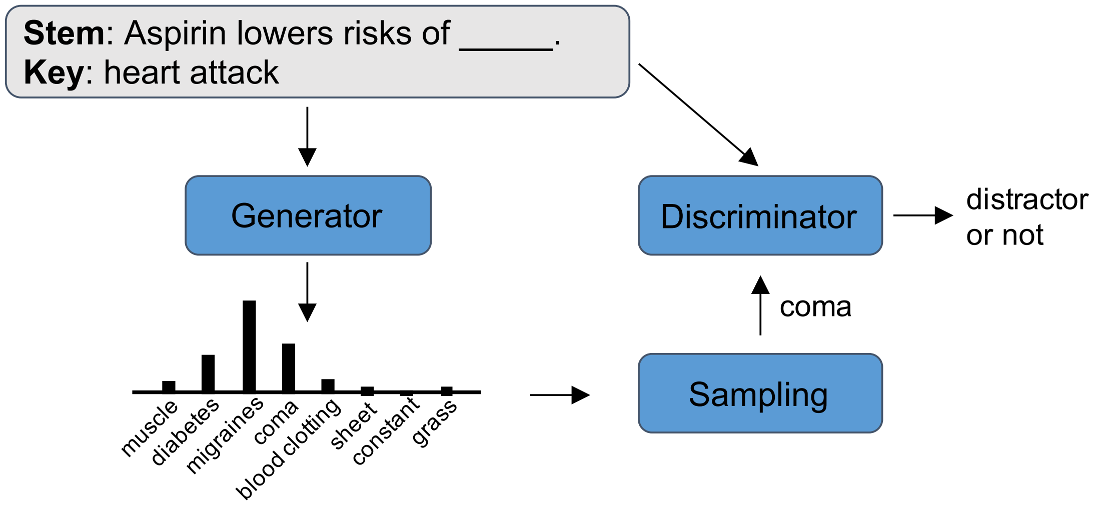


Figure 6.3. Proposed adversarial training framework for DG.

likelihood:

$$\max_{\phi} \; \mathbb{E}_{d \sim P_{\text{true}}(d|q,a)}[\log(\sigma(f_\phi(d,q,a)))] + \mathbb{E}_{d \sim P_\theta(d|q,a)}[\log(1-\sigma(f_\phi(d,q,a)))] \quad (6.5)$$

where σ is the sigmoid function. For the generator G, we choose another scoring function $f_\theta(d,q,a) \in \mathbb{R}$ parameterized by θ, evaluate it on every possible distractor d_i given a (q,a) pair, and sample generated distractors based on the discrete probability after applying softmax:

$$p_\theta(d_i|q,a) = \frac{\exp(\tau \cdot f_\theta(d_i,q,a))}{\sum_j \exp(\tau \cdot f_\theta(d_j,q,a))} \quad (6.6)$$

where τ is a temperature hyper-parameter.

In practice, since the total size of distractors is large, it is very time-consuming to evaluate on every possible d_i. Following the common practice as in [22, 157], we uniformly sample K candidate distractors for each (q,a) pair and evaluate f_θ on each $d_i, \forall i \in [1,K]$. The objective for G is to "fool" D so that D mis-classifies distractors generated by G as positive:

$$\min_{\theta} \; \mathbb{E}_{d \sim P_\theta(d|q,a)}[\log(1-\sigma(f_\phi(d,q,a)))] \quad (6.7)$$

The training procedure follows a two-player minimax game, where D and G are

Dataset	$\lvert\mathcal{D}\rvert$	# MCQs	# Train	# Valid	# Test	Avg. # Dis
SciQ	22379	13679	11679	1000	1000	3
MCQL	16446	7116	5999	554	563	2.91

Table 6.3. Dataset Statistics.

alternatively optimized towards their own objective.

The scoring function f_ϕ and f_θ can take arbitrary forms. IRGAN utilizes a convolutional neural network based model to obtain sentence embeddings and then calculates the cosine similarities. However, such a method ignores the word-level interactions, which is important for the DG task. For example, if the stem asks "which physical unit", good distractors should be units. Therefore, we adopt the Decomposable Attention model (DecompAtt) [127] proposed for Natural Language Inference to measure the similarities between q and d. We also consider the similarities between a and d. Since they are usually short sequences, we simply use the cosine similarity between summed word embeddings. As such, the scoring function is defined as a linear combination of DecompAtt(d, q) and Cosine(d, a).

6.4.2.3 Cascaded Learning Framework

To make the ranking process more efficient and effective, we propose a cascaded learning framework, a multistage ensemble learning framework that has been widely used for computer vision [155]. We experiment with 2-stage cascading, where the first stage ranker is a simple model trained with part of the features in Sec. 6.4.2.1.1 and the second stage ranker can be any aforementioned ranking model. Such cascading has two advantages: (i) The candidate size is significantly reduced by the first stage ranker, which allows the use of more expensive features and complex models in the second stage; (ii) The second stage ranker can learn from more challenging negative examples since they are top predictions from previous stage, which can make the learning more effective.

6.4.3 Experiments

6.4.3.1 Datasets

We evaluate the proposed DG models on the following two datasets: (i) **SciQ** [162]: crowdsourced 13.7K science MCQs covering biology, chemistry, earth science, and physics. The questions span elementary level to college introductory level in the US. (ii) **MCQL**: 7.1K MCQs crawled from the Web. Questions are about biology, physics, and chemistry and at the Cambridge O level and college level.

For SciQ, we follow the original train/valid/test splits. For MCQL, we randomly divide the dataset into train/valid/test with an approximate ratio of 10:1:1. We convert the dataset to lowercase, filter out the distractors such as "all of them", "none of them", "both A and B", and keep questions with at least one distractor. We use all the keys and distractors in the dataset as candidate distractor set $\mathcal{D}$. Table 6.3 summarizes the statistics of the two datasets after preprocessing. $|\mathcal{D}|$ is the number of candidate distractors. # MCQs is the total number of MCQs. # Train/Valid/Test is the number of questions in each split of the dataset. Avg. # Dis is the average number of distractors per question.

6.4.3.2 Experiment Settings

We use Logistic Regression (**LR**) as the first stage ranker. As for the second stage, we compare LR, Random Forest (**RF**), LambdaMART (**LM**), and the proposed NN-based model (**NN**). Specifically, we set C to 1 for LR, use 500 trees for RF, and 500 rounds of boosting for LM. For first stage training, the number of negative samples is set to be equal to the number of distractors, which is 3 for most questions. And we sample 100 negative samples for second stage training. More details can be found in the supplementary material. In addition, we also study the following unsupervised baselines that measure similarities between the key and distractors: (i) pointwise mutual information (**PMI**) based on co-occurrences; (ii) edit distance (**ED**), which measures the spelling similarity; and (iii) GloVe embedding similarity (**Emb Sim**).

For evaluation, we report top recall (R@10), precision (P@1, P@3), mean average precision (MAP@10), normalized discounted cumulative gain (NDCG@10), and mean reciprocal rank (MRR). Specifically,

$$R@K = \frac{\sum_{i=1}^{K} rel(i)}{\#distractors} \tag{6.8}$$

$$P@K = \frac{\sum_{i=1}^{K} rel(i)}{K} \tag{6.9}$$

$$AP@K = \frac{\sum_{i=1}^{K} P@i \cdot rel(i)}{\sum_{i=1}^{K} rel(i)} \tag{6.10}$$

$$MAP@K = \frac{\sum_{q=1}^{Q} AP@K(q)}{Q} \tag{6.11}$$

$$DCG@K = \sum_{i=1}^{K} \frac{2^{rel(i)} - 1}{\log_2(i+1)} \tag{6.12}$$

$$NDCG@K = \frac{DCG@K}{IDCG@K} \tag{6.13}$$

$$MRR = \frac{1}{Q} \sum_{i=1}^{Q} \frac{1}{rank(i)} \tag{6.14}$$

where $rel(\cdot)$ is a binary indicator function; $rel(k) = 1$ if the item at rank k is a distractor associated with a MCQ otherwise $rel(k) = 0$; $\#distractors$ is the number of distractors associated with a MCQ; Q is the number of MCQs in the dataset; $IDCG@K$ is the ideal discounted cumulative gain, which is calculated by ranking all distractors at the top of the result list; $rank(i)$ refers to the rank position of the first distractor for the i-th MCQ.

6.4.3.3 Training and Implementation Details

6.4.3.3.1 Feature-based Models. We use the implementations of scikit-learn [128] for logistic regression and random forest experiments. For LambdaMART experiments, we use the XGBoost library [26]. For both SCIQ and MCQL datasets we train with 500 rounds of boosting, step size shrinkage of 0.1, maximum depth of 30, minimum child weight of 0.1 and minimum loss reduction of 1.0 for partition. For calculating Wiki Sim features, we use a Wikipedia dump of Oct. 2016. Part of speech tags are calculated with NLTK [13].

The logistic regression used for the first stage ranker is based on features including: Emb Sim, POS Sim, ED, Token Sim, Length, Suffix, and Freq. Models for the second stage ranker is based on all features described in Sec. 6.4.2.1.1.

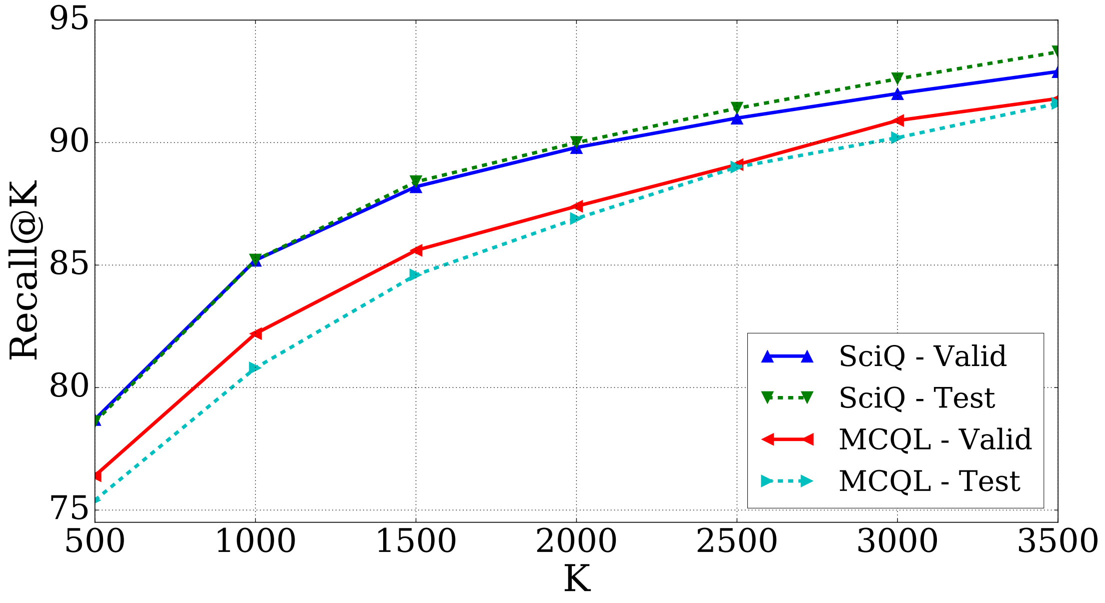


Figure 6.4. Recall@K for the first stage ranker.

6.4.3.3.2 NN-based Models. Our NN-based models are implemented with TensorFlow [1]. When training the generator, we first uniformly select $K = 512$ candidates and then sample 16 distractors according to Equation 6.6. The temperature τ is set to 5. Our scoring functions are based on Decomposable Attention Model [127]. The word embeddings are initialized using the pre-trained GloVe [129] (840B tokens), and the embedding size is 300. Our model is optimized using Adam algorithm [76] with a learning rate of 1e-4 and a weight decay of 1e-6.

Since the sampling process in G is not differentiable, the gradient-decent-based optimization in the original GAN paper [51] is not directly applicable. To tackle this problem, we use policy gradient based reinforcement learning as in IRGAN.

6.4.3.4 Experimental Results

6.4.3.4.1 First Stage Ranker The main goal of the first stage ranker is to reduce the candidate size for the later stage while achieving a relatively high recall. Figure 6.4 shows the Recall@K for the first stage ranker on the two datasets. Validation set is used for choosing top K predictions for later stage training. We empirically set K to 2000 for SciQ and 2500 for MCQL to get a recall of about 90%.

1st Stage Ranker	2nd Stage Ranker	R@10	P@1	P@3	MAP @10	NDCG @10	MRR
LR	PMI	11.0	2.1	3.1	3.6	6.8	8.8
	ED	14.3	12.6	9.2	8.7	12.5	18.9
	Emb Sim	19.3	9.3	9.0	9.6	14.2	17.5
	LR	29.7	14.8	14.1	14.7	22.1	27.6
	RF	**44.1**	36.8	**27.0**	**28.4**	**38.0**	49.2
	LM	43.3	**37.2**	26.4	28.0	37.5	**49.3**
	NN	24.6	11.7	11.7	11.6	23.1	25.7
RF	—	41.4	31.2	23.7	25.0	34.4	44.0
LM	—	39.1	26.5	22.6	22.9	31.8	40.4

Table 6.4. Ranking results (%) for DG on SciQ dataset.

1st Stage Ranker	2nd Stage Ranker	R@10	P@1	P@3	MAP @10	NDCG @10	MRR
LR	PMI	20.7	5.9	6.8	7.8	13.5	16.2
	ED	32.1	34.6	23.6	23.7	30.5	42.8
	Emb Sim	32.1	25.6	18.4	20.4	26.9	33.9
	LR	42.9	29.3	24.5	26.6	35.1	42.2
	RF	48.4	**45.5**	**32.7**	**35.4**	**43.8**	**54.8**
	LM	**49.4**	42.8	31.5	34.5	43.4	53.6
	NN	36.5	22.9	22.5	22.7	34.6	36.7
RF	—	48.0	40.9	30.4	33.6	42.0	51.1
LM	—	46.7	42.5	30.6	33.0	41.6	52.7

Table 6.5. Ranking results (%) for DG on MCQL dataset.

6.4.3.4.2 Distractor Ranking Results Table 6.4 and Table 6.5 lists the ranking results for DG. From the table we observe the following: (i) The proposed ranking models perform better than unsupervised similarity-based methods (PMI, ED, and Emb Sim) most of the time, which is expected since similarity-based heuristics are used as features. (ii) Ensemble models - RF and LM - have comparable performance and are significantly better than other methods. These ensemble methods are more suitable for capturing the nonlinear relation between the proposed feature set and distractors. (iii) NN performs worse than feature-based models. The main reason is that NN is solely based on word embeddings. Although embedding

#	**SciQ**	**MCQL**
1	Emb Sim (a, d)	Emb Sim (a, d)
2	Freq d	Token Sim (a, d)
3	Freq a	ED
4	Wiki Sim	Suffix
5	Emb Sim (q, d)	Suffix / len(d)
6	Suffix	Freq a
7	Suffix / len(d)	Wiki Sim
8	Suffix / len(a)	Freq d
9	Token Sim (a, d)	Emb Sim (q, d)
10	ED	Suffix / len(a)

Table 6.6. Top 10 important features for learning to rank distractors.

similarity is the most important feature, information provided by other top features such as ED, Suffix, Freq is missing in NN. Given the limited training examples (11.6K for SciQ and 6K for MCQL), it is difficult to learn a robust end-to-end NN-based model.

6.4.3.4.3 Feature Analysis We conduct a feature analysis to have more insights on the proposed feature set. Feature importance is calculated by "mean decrease impurity" using RF. It is defined as the total decrease in node impurity, weighted by the probability of reaching that node, averaged over all trees of the ensemble. Table 6.6 lists the top 10 important features for SciQ and MCQL datasets. We find that: (i) the embedding similarity between a and d is the most important feature, which shows embeddings are effective at capturing semantic relations between a and d. (ii) String similarities such as Token Sim, ED, and Suffix are more important in MCQL than those in SciQ. This is consistent with the observation that ED has relatively good performance as seen in Table 6.5. (iii) The set of top 10 features is the same for SciQ and MCQL, regardless of order.

6.4.3.4.4 Effects of Cascaded Learning Since we choose the top 2000 for SciQ and 2500 for MCQL from first stage, the ranking candidate size is reduced by 91% for SciQ and 85% for MCQL, which makes the second stage learning more efficient. To study whether cascaded learning is effective, we experiment with RF and LM without 2-stage learning, as shown as the bottom two rows in Table 6.1. Here we sample 100 negative samples for training models in order to make a fair comparison with other methods using 2-stage learning. We can see that the performance is better when cascaded learning is applied.

6.4.4 Summary

This section investigated DG as a ranking problem and applied feature-based and NN-based supervised ranking models to the task. Experiments with the SciQ and the MCQL datasets empirically show that ensemble learning models (random forest and LambdaMART) outperform both the NN-based method and unsupervised baselines. The MCQL data is publicly available upon request. The two datasets can be used as benchmarks for further DG research. Future work will be to design a user interface to implement the proposed models to help teachers with DG and collect more user data for model training.

6.5 Discussion

This chapter has discussed supervised learning methods for automatic distractor generation for creating multiple choice questions. Despite the methodology contribution, we also notice two limitations: (i) The proposed DG methods currently only create questions that test the ability of recalling facts and basic concepts. According to Bloom's Taxonomy [8], these questions are ideal for lower-level material. Generating assessments at higher levels of the taxonomy is beyond the scope of the dissertation. (ii) The proposed methods are only evaluated with small-scale crowdsourcing without being tested under a real educational setting. Thus a promising future direction should be to deploy the proposed DG methods to an actual MCQ creation system and to let people take the assessments generated. The deployment of such a MCQ creation system is non-trivial and is yet to be explored. Next chapter will introduce BBookX, an educational application where the proposed methods for distractor generation and concept prerequisite learning would potentially be useful.

Chapter 7 | BBookX: a Computer-facilitated Book Creation Tool

7.1 Introduction

Textbooks play a crucial role in both the teaching and learning process. However, in practice, textbooks have several issues. Authoring a new textbook usually requires a great amount of time and effort, even for experts. A tool that facilitates the book creation process could be very helpful. It is not easy to keep an existing textbook up-to-date, especially for fast changing domains such as computer science. For example, most existing machine learning books as of this date lack recent developments in deep learning. However, most of such information can be found in OERs, such as publicly available scientific papers, lecture notes, Wikipedia, etc. For certain classes, teachers may want to modify the book content, change the order of sections, or combine content from different books. This can give flexibility for designing personalized textbooks.

To deal with these issues this chapter introduces BBookX [88, 89], a novel collaborative computer-facilitated textbook creation system. The goal for BBookX is to automatically build with author guidance online textbooks from open educational resources (OERs). Designed to utilize information retrieval techniques to intelligently harvest existing OERs, BBookX is currently built on top of Wikipedia. However, it can also incorporate other available OERs. Such a tool has the potential to reduce the cost of creating and maintaining learning resources, contribute to open educational resources, and provide more up to date information for fast changing fields.

BBookX has an interactive Web interface that supports real-time book creation. Given a set of user-generated criteria, it returns a list of relevant material that can be put into the book. Users can accept, reject, or reorder the returned results. Such feedback allows the system to reformulate the query in order to return better results. In addition, BBookX can create an open version of existing textbooks by automatically linking their existing book chapters to Wiki articles.

7.2 System Overview

The system overview of BBookX shown in Figure 7.1 consists of two major components: the open book repository construction and the interactive book creation tool. The open book repository is built from two resources. First, existing online open access textbooks are collected. Second, for other textbooks, it is possible to create a Wiki-based open version by linking chapters to Wikipedia articles. Both open access books and Wiki-based open books are stored in the open book repository and indexed using Solr/Lucene[1]. The interactive book creation component allows users to specify the information of the book which they want to build using queries. The system will then retrieve a list of indexed educational resources ranked by the relevance to the query. An interactive user interface provides easy click selection and drag/drop functions allowing users to evaluate the returned resources. User feedback is utilized by an explicit relevance feedback mechanism to reformulate the query to generate a new list of results. The generated book will be refined through such an interactive search process. Details of these two components are discussed later.

7.3 Open Book Repository

7.3.1 Collecting Open Access Textbooks

In order to construct our open book repository, we first create a collection of existing online open access textbooks by crawling different websites. Our available data

[1]http://lucene.apache.org/solr/

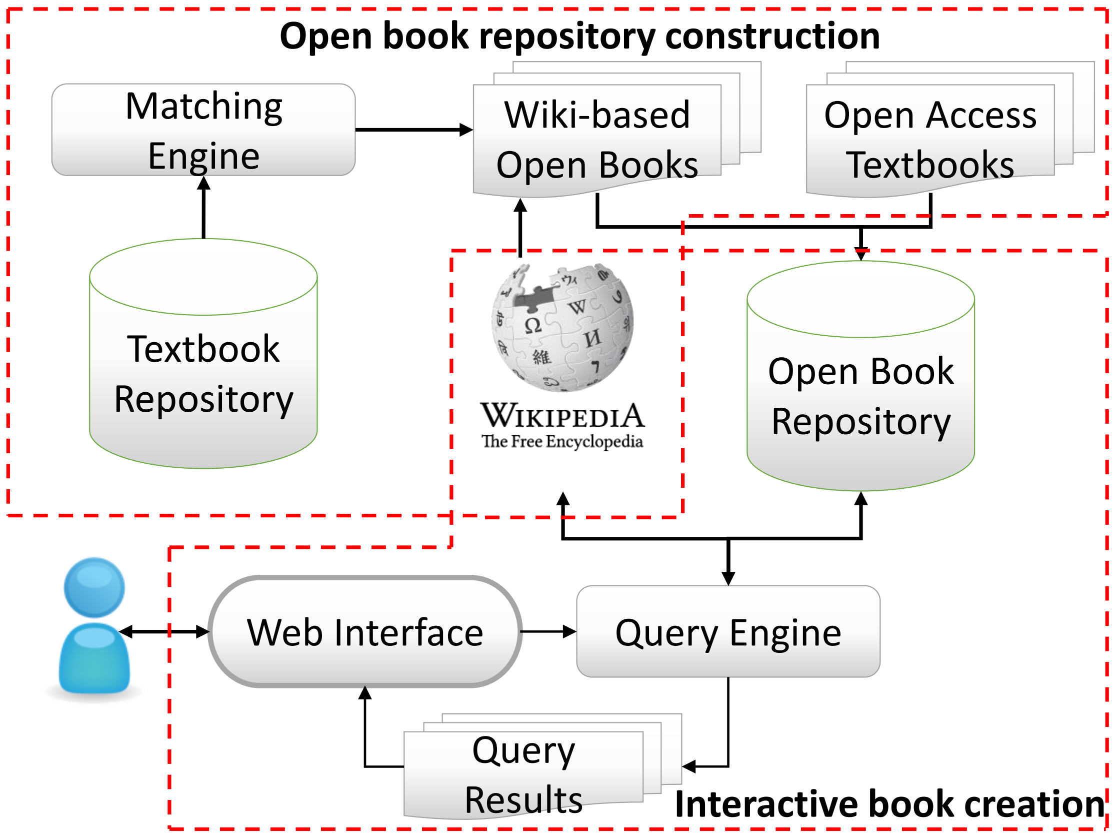


Figure 7.1. BBookX system overview.

sources include Wikibooks, Saylor Academy[2], MIT OCW[3], OpenStax College[4], etc. So far we have collected more than 3000 open access textbooks.

7.3.2 Generating Open Books Using Wikipedia

In order to provide high-quality and well-structured open books, we also utilize classic textbooks created by experts. Here we create an open version of existing books using Wikipedia. Our method is to link book chapters to Wiki articles. Specifically, for each chapter we provide users a list of Wiki articles which are most relevant to the topics covered in the book. Our method consists of three modules: concept identification, candidate selection and candidate ranking.

Concept Identification identifies the important concepts discussed in each

[2]http://www.saylor.org/books/
[3]http://ocw.mit.edu/courses/online-textbooks/
[4]http://openstaxcollege.org/

Textbook	P@1	P@3	P@5	MAP@10
Computer networks	0.84	0.52	0.42	0.37
Macroeconomics	0.83	0.54	0.42	0.34
Precalculus	0.83	0.46	0.39	0.34

Table 7.1. Performance of the candidate ranking on three different textbooks.

book chapter. First, we build a domain-specific dictionary which contains the concepts related to the book topic. A depth-first search method is used to crawl Wikipedia with the seed set to be the main Wiki page related to the topic. Titles of Wiki articles visited by our crawler will be added to the concept dictionary [137]. Second, we match concepts from the dictionary in the book chapter and calculate the importance score for each concept using term frequency-inverse document frequency (tf-idf).

Candidate Selection selects the candidate Wiki articles related to the concepts in the book chapter. Two approaches are applied to collecting the candidate set. The similarity between the title of the book chapter and the title of Wiki articles is determined whereby if the Wiki title also appears in the chapter title, then the Wiki article is added to the candidate set. In addition, the similarity between the content of Wikipedia articles and that of the book chapter when each book chapter and Wikipedia article is represented as a tf-idf vector using all vector space concepts in the dictionary. The content similarity is calculated by the cosine similarity between tf-idf vectors and top N Wiki articles with high similarity are included in the candidate set.

Candidate Ranking ranks the candidates for the most relevant Wiki articles. Using a learning to rank model, SVM^{rank} [72], different features are extracted for training including local features, such as content similarity and the Jaccard distance between titles, and global features such as redundancy features and consistency features[5]. While local features are able to capture the relatedness between the book chapter and Wiki candidates, global features are used to ensure global coherence between Wiki candidates. SVM^{rank} is tested on three textbooks from different domains. For each book chapter, three graduate students label each Wiki candidate as "relevant" or "irrelevant" with evaluation of 5-fold cross-validation on all chapters of the three books. As listed in Table 7.1, the performance of the proposed ranking

[5]More details of the proposed candidate ranking method can be found in our recent work [158].

3. The Application Layer	**3. The Application Layer**
3.1 Principles	3.1 Principles
3.1.1 The Peer-to-peer Model	3.1.1 Peer-to-peer
3.1.2 The Transport Services	3.1.2 Connectionless-mode Network Service
3.2 Application-level Protocols	3.2 Application-level Protocols
3.2.1 The Domain Name System	3.2.1 Domain Name System
3.2.2 Electronic Mail	3.2.2 Email
3.2.3 The HyperText Transfer Protocol	3.2.3 Hypertext Transfer Protocol
5. The Network Layer	**5. The Network Layer**
5.2 Internet Protocol	5.2 Internet Protocol
5.2.1 IP Version 4	5.2.1 IPv4
5.2.2 ICMP Version 4	5.2.2 Internet Control Message Protocol

Figure 7.2. Example of a Computer Networking book created using Wikipedia. The left part is the table of contents of the original book. The right part is the generated table of contents.

method is consistent on three different textbooks.

Figure 7.2 shows a part of the book generated for an existing textbook "Computer Networking: Principles, Protocols, and Practice". The left side of the figure is the table of contents of the original book. Our method of the subsections (e.g. 3.1.1, 3.1.2) show the top candidate Wiki article for each section on the right side and link the book chapter sections to the relevant Wiki article.

7.3.3 Indexing Subsystem

The indexing subsystem indexes all open educational resources collected by the system, including Wikipedia articles, open access textbooks, and Wiki-based open books created by BBookX. A Wikipedia dump of February 2016 is used. Each Wikipedia article and book chapter indexed is first preprocessed, which includes tokenization, stop word and punctuation removal, conversion to lower case, and stemming. Then Solr/Lucene is used to build a full text index for the content of each document. To calculate the similarity score, key phrases of each document are also extracted and indexed [147]. Specifically, anchor texts are extracted from each Wiki article as key phrases. For book chapters, the Maui tool [110] is used to extract keyphrases. Besides full text and key phrases, the metadata of each document is also indexed, such as Wiki title, book chapter title, document source, etc.

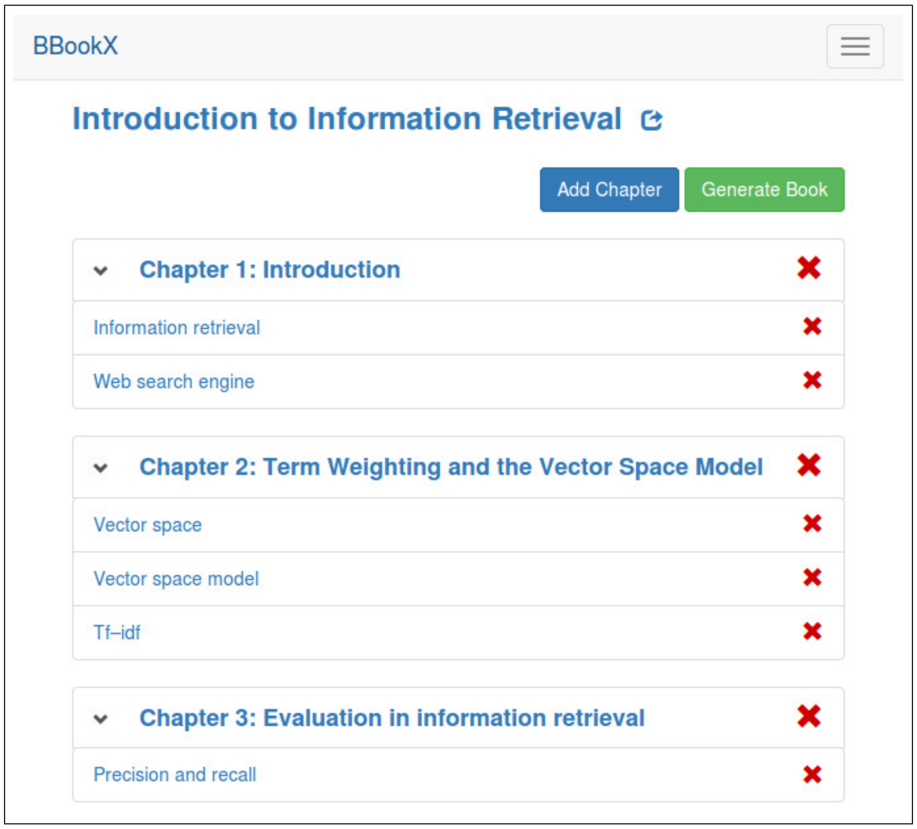
7.4 Interactive Book Creation

Parts of the user interface for the interactive book creation tool are shown in Figure 7.3.

7.4.1 User Interface

The software for BBookX is a node application built with the SANE stack (Sailsjs and Emberjs). Sailsjs is used for the API interface to store in MongoDB the application data, including information about users and their built books. Emberjs is used to build the JavaScript front end.

The current web interface components are shown in Figure 7.3. To use BBookX, users start with registration and login. They can then choose to keep working on previously built books in the "Library" or start building a new book. The book building process first requires adding a book title. As shown in figure 7.3a, users can then click the "Add Chapter" button to work on a new chapter. Figure 7.3b shows the interface to add a chapter. After creating a chapter title and adding a short description which can be keyphrases or sentences, users click the "Run" button to let BBookX retrieve a list of relevant OER. When the system finishes searching, users can review each result by clicking the links that allow a preview in a new browser tab. Users then click the results they wish to keep and add them to the "Saved Results" list. If users are still not satisfied with the results or want more relevant OER, they can click "Regenerate Results" to re-run the search. User feedback is be utilized by the system to reformulate the query for other results. In addition, BBookX supports reordering OER or chapters by dragging and dropping. The built books can be exported as text files for further editing or as HTML for embedding in other Web pages.



Introduction to Information Retrieval

Chapter 1: Introduction
- Information retrieval
- Web search engine

Chapter 2: Term Weighting and the Vector Space Model
- Vector space
- Vector space model
- Tf–idf

Chapter 3: Evaluation in information retrieval
- Precision and recall

(a) Building a book.

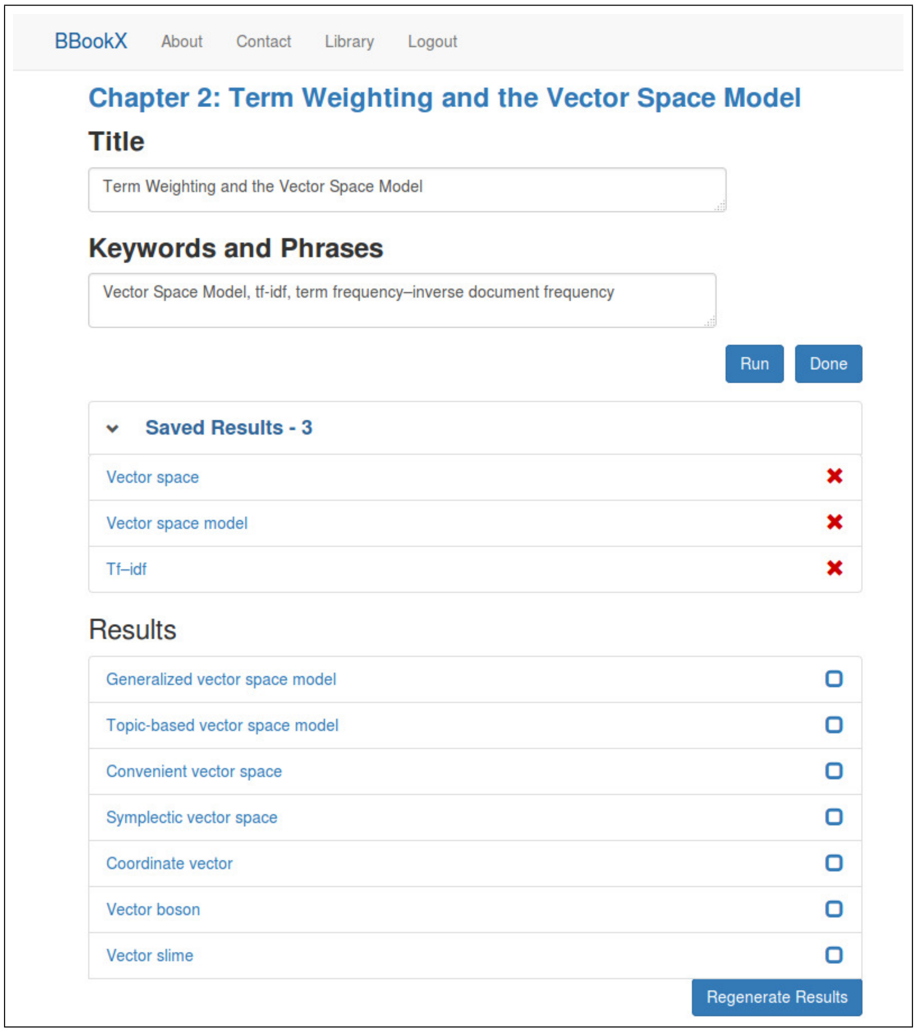
Figure 7.3. Interactive User interface.

7.4.2 Query Subsystem

The query subsystem receives the information of a chapter/subchapter as a query and returns a ranked list of relevant educational resources, including Wiki articles and sections of open books. It mainly consists of the following three processes:

Querying: For a query $q = (t_q, c_q)$, where t_q is the chapter title and c_q is the associated descriptive text, BBookX retrieves a set of candidate relevant resources D by querying t_q and c_q in the pre-built full text index. The key phrases of q, denoted as k_q, are also extracted for the following ranking process.

Ranking: For each candidate resource d in D, BBookX calculates the similarity score between q and d, denoted as $sim(q, d)$, by considering features similar to local features described in Section 7.3.2, such as title similarity and text similarity. D is sorted by similarity score in descending order as a ranked list.


BBookX About Contact Library Logout

Chapter 2: Term Weighting and the Vector Space Model

Title

Term Weighting and the Vector Space Model

Keywords and Phrases

Vector Space Model, tf-idf, term frequency–inverse document frequency

Run Done

Saved Results - 3

- Vector space ✖
- Vector space model ✖
- Tf–idf ✖

Results

- Generalized vector space model ☐
- Topic-based vector space model ☐
- Convenient vector space ☐
- Symplectic vector space ☐
- Coordinate vector ☐
- Vector boson ☐
- Vector slime ☐

Regenerate Results

(b) Adding a chapter.

Figure 7.3. Interactive User interface. (cont.)

In our system, $sim(q, d)$ is calculated as

$$sim(q, d) = \alpha_1 \cdot cosSim(c_q, c_d) + \alpha_2 \cdot cosSim(k_q, k_d) + \alpha_3 \cdot Jaccard(t_q, t_d)$$

where $cosSim(c_q, c_d)$ is the cosine similarity between the word vectors of the content of q and d, $cosSim(k_q, k_d)$ is the cosine similarity between key phrase vectors of q and d, and $Jaccard(t_q, t_d)$ is the Jaccard similarity between the title of q and d. Specifically, $\alpha_1 = 0.2$, $\alpha_2 = 0.2$, and $\alpha_3 = 0.6$.

Relevance feedback: Users can decide whether to keep a returned item or retrieve a new ranked list of items. BBookX incorporates a relevance feedback mechanism, which includes three steps: 1) Store the results kept by users as relevant results; 2) Select the top 20 key phrases from these results using term frequency weights; 3) Perform query expansion by adding these selected key phrases to the description c_q. A new query q' is then used to retrieve relevant results. Since users usually will not write a detailed chapter description, the relevance feedback will help users to find relevant results. Of course, other methods can be used and are currently being explored.

7.5 Field Test

BBookX was deployed with students in an introductory information sciences and technology course. A lab assignment was created that required the use of BBookX, asking the students to use the software to generate a 3-chapter book, covering topics at the intersection of each students' major and information sciences. An exploratory survey was then administered (n=140), used to better understand the usability of BBookX, On a four point scale (very positive, positive, negative, very negative), 72% rated their overall experience with BBookX either positive (59%) or very positive (13%), 89% rated the learnability of BBookX either positive (45%) or very positive (44%), 65% rated their satisfaction with BBookX either positive (55%) or very positive (10%), and 62% rated the efficiency of using BBookX either positive (46%) or very positive (16%). During a post-lab discussion in class, it was discovered that some of the students did not realize that the searches within each chapter are designed to be iterative; the more you accept or reject results, the more accurate the successive search. This requires forther exploration, as one hypothesis on why some students found the efficiency and satisfaction of using BBookX to be negative is that they only ran a single search within each chapter, thus not experiencing the results getting more personalized based on user actions.

The most interesting question asked students if BBookX surfaced interesting pages of content, including things the student did not know before completing this homework. On a 4-point scale (strongly agree, agree, disagree, strongly disagree), 73% answered either agree (61%) or strongly agree (12%). This speaks positively about the algorithms powering the software, as they are finding and surfacing pages of content that provide new learning opportunities for students focused on their specific majors. Of note is that this was a general education course, with a wide range of class standings, from freshman to 5-th year seniors.

7.6 Book Publishing

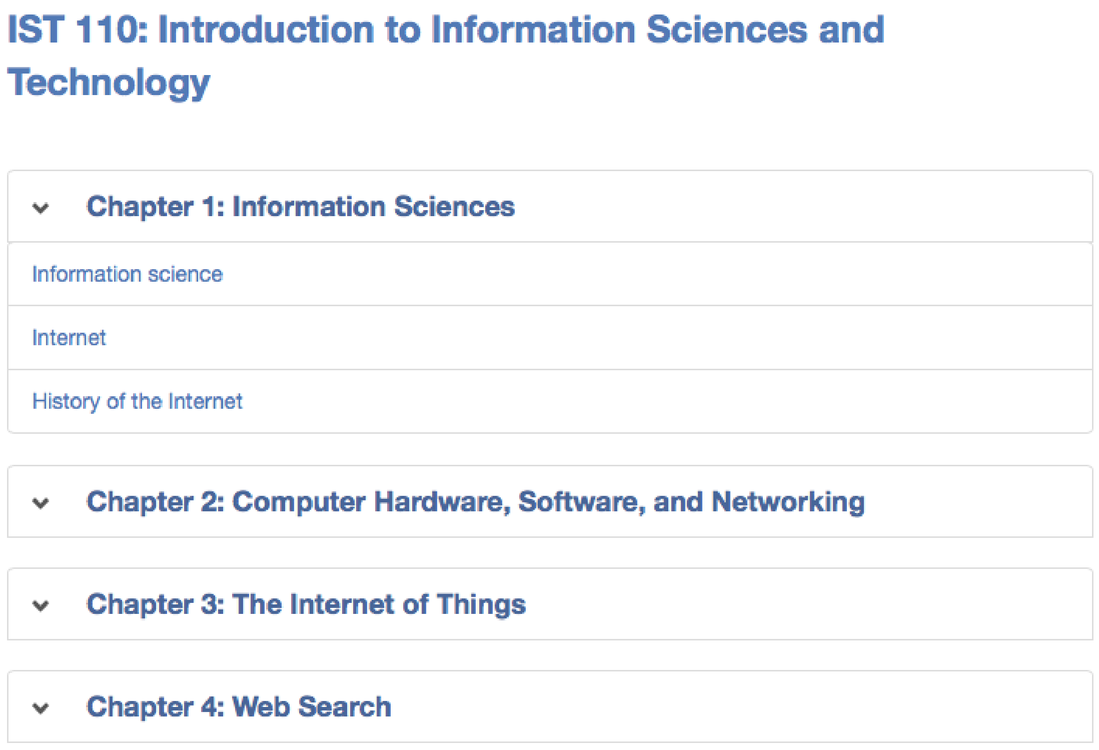
Once a user finalizes his or her book, BBookX features a book generation tool. Currently, book generation involves extracting text-based content from the selected Wikipedia entries, and combining them in a text file with some formatting. This feature was conceptually well-received by user testers, though it quickly was apparent that distributing a lengthy text file to students as their primary textbook was undesirable. The second phase of development planned for BBookX involves a new publishing UI that users will interact with after finalizing a book. The publishing UI is intended to be a flexible UI that allows users to add, edit, and delete content from the generated book. Once finished in the publishing phase, the book can be shared with students in a variety of methods, such as a designed web interface, an e-text, PDF, and possibly other formats.

The current implementation to quickly make the finalized books more accessible is to use a 'Share' feature, which creates an iframe containing a link to the final book, and users can embed this iframe on websites and in Learning Management Systems (LMS)(see Figure 7.4). This creates a quick way to share created books, though users do not have any ability to edit books. One user is leveraging a book created in this way as the primary textbook to support a course of 150 students.

7.7 Related Work

To our knowledge, there is little work similar to ours. While there are systems for helping create books, such as FlexBook[6], Wikibooks, and METIS [101], BBookX is novel because of its automatic textbook creation aspects. Other existing textbook building systems do not support the automatic retrieval and organization of OER, but require manually attached related resources. FlexBook is a textbook authoring platform where users can produce and customize the book content by re-purposing educational content. Wikibooks provides a wiki-based platform which allows different users to collaboratively create books [15].

[6]CK-12 Foundation: http://www.ck12.org/


IST 110: Introduction to Information Sciences and Technology

Chapter 1: Information Sciences

Information science

Internet

History of the Internet

Chapter 2: Computer Hardware, Software, and Networking

Chapter 3: The Internet of Things

Chapter 4: Web Search

Figure 7.4. Example of a completed book, when embedded on a website. The first chapter is expanded, to view the links to each section.

The method used in the interactive query subsytem of BBookx is related to relevance feedback [143]. Three types of feedback can be utilized: explicit feedback, which is collected by users explicitly marking relevant and irrelevant documents [163]; implicit feedback, which is inferred by the system based on users' observable behaviors such as clicks and votes [3, 73]; and pseudo feedback, which is gathered by assuming the top-k ranked documents are relevant [19]. Currently, BBookX uses only explicit feedback.

Our methods for generating open books using Wikipedia is also similar to wikification [111, 114], which automatically identifies concept mentions in text and links them to referents in Wikipedia. The difference is our focus is on extracting the most important concepts for each book chapter, not named entities mentioned in the general text.

7.8 Summary and Discussion

This chapter introduced BBookX, a novel search-based system designed to facilitate the online book creation with the help of OERs. BBookX was launched within the Pennsylvania State University on May 26th, 2015, then opened for general use beginning October of 2016 at `https://bbookexp.psu.edu/`. When examining system log data in December of 2016, BBookX has 416 users, that created 419 books. Over 47,000 searches were exectuted within BBookX, resulting in 1,086 book chapters. Note that BBookX system is still a work in progress and being actively updated. There are ongoing faculty user testing and interviews to further evaluate different aspects of the system [133].

BBookX is an actual educational application where the methods proposed in previous chapters could be applied. First, a key difference between BBookX and traditional search engines is that the retrieved OERs by BBookX ideally should not only be relevant but also should follow a correct learning order. The proposed methods on concept prerequisite learning could be beneficial for BBookX to organize different retrieved chapters in an appropriate learning order. Second, since the goal of the proposed BBookX system is to facilitate the textbook authoring process, the ability to automatically create questions will be of great interest to its potential users. This provides a great opportunity to deploy automatic MCQ creation methods. The proposed QG methods could also be applied.

Chapter 8 Conclusion and Future Work

8.1 Conclusion

This dissertation focused on machine learning methods for two educational applications: concept prerequisite learning and automatic distractor generation. The first part of the dissertation focused on data-driven methods for concept prerequisite learning, which was discussed in chapters 2 through 5:

The dissertation started by introducing a simple but effective link-based feature – RefD – for measuring concept prerequisite relations. RefD is designed to measure how differently two concepts refer to each other and uses the difference as a proxy for concept prerequisite relations. RefD has been shown effective and outperforms an existing supervised learning baseline for the task. After that the dissertation proposed to learn concept prerequisites from university course dependencies. An optimization based framework was proposed to solve the CPR-Recover problem. Experiment results on a synthetic dataset and an actual course dataset both showed that the proposed method outperformed existing baselines.

Later the dissertation investigated supervised learning for concept prerequisite learning and focused on active learning to handle the lack of concept prerequisite labels. As the first study to apply active learning to concept prerequisite learning, this study proposes a novel set of features tailored for prerequisite classification and compares the effectiveness of four widely used query strategies. Experimental results for domains including data mining, geometry, physics, and precalculus show that active learning could reduce the amount of training data required for concept prerequisite learning.

To explore the mathematical nature of a prerequisite relation being a strict partial order, the dissertation proposed an active learning framework tailored for strict partial orders. The proposed approach incorporates relational reasoning not only in finding new unlabeled pairs whose labels can be deduced from an existing label set, but also in devising new query strategies that consider the relational structure of labels. Experiments on concept prerequisite relations showed the proposed framework can substantially improve the classification performance with the same query budget compared to other baseline approaches.

The second part of this dissertation studied automatic distractor generation for creating multiple choice questions, as discussed in Chapter 6. In contrast with previous unsupervised similarity-based DG methods, we proposed two data-driven, learning-based approaches for DG:

First, we proposed a generative model learned from training GANs to create useful distractors for automatically creating fill-in-the-blank questions. A generator G is trained to capture the conditional answer distribution given the question sentence. And the discriminator D is trained to estimate the conditional probability that an answer came from the training data rather than G given the question sentence. Experimental results showed that this context-based method achieved comparable performance to a frequently used word2vec-based method for the Wiki dataset. In addition, we proposed a second-stage learner to combine the strengths of the two methods, which further improved the performance on both datasets. Second, we studied how ranking models can select useful distractors for creating MCQs. We proposed models which can learn to select distractors that resemble those distractors in actual exam questions. We empirically studied feature-based and NN-based ranking models with experiments on the recently released SciQ dataset and our MCQL dataset. Experimental results demonstrate that feature-based ensemble learning methods (random forest and LambdaMART) outperform both the NN-based method as well as unsupervised baselines.

In the final chapter of this dissertation we introduced BBookX, a computer-facilitated book creation tool. Using information retrieval techniques, BBookX is designed to facilitate the online book creation process by searching OERs. It was initially launched within the Pennsylvania State University and then opened for general use in October of 2016. BBookX is an actual educational application where the methods for concept prerequisite learning and automatic distractor generation proposed in previous chapters could be applied.

8.2 Future Work

Built on the work described in this dissertation, future work can take multiple directions.

First, our work on active learning of strict partial orders assumes that the oracle never makes mistakes. It would be interesting to study active learning of strict partial orders from a noisy oracle. The reasoning module for closure calculation proposed in Section 5.2 would be problematic when the labels are inconsistent with the properties of strict partial orders. In addition, it would be very helpful to develop a knowledge base for educational purposes which focuses on the knowledge concepts. Existing knowledge bases such as YAGO[1] and Freebase[2] are largely based on named entities including people, organizations, groups, etc. However, knowledge concepts such as "Machine learning", "Deep learning", and "Support vector machine" should interest educational applications. Having such a knowledge base would potentially enable deep learning-based relation extraction methods for concept prerequisite relations and other relations of interest.

Besides distractor generation, another important task for automatic MCQ creation is *question stem generation*: creating question sentences from an input paragraph. Traditional methods for question stem generation relies heavily on the rule-based syntactic transformation of a declarative sentence. Only recently have a few studies [42, 173] worked outside such manually generated rules. Developing data-driven question stem generation methods, thus, would be a promising direction to take.

It is also worth investigating the deployment of the methods proposed in this dissertation to actual educational applications. BBookX is at an early stage and is still a work in progress. It would be interesting and nontrivial to incorporate automatic prerequisite discovery and question generation methods into the BBookX system. The system can also serve as an evaluation platform for these methods. In addition, the feedback collected from users' clicks could be used as signals for

[1]https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago

[2]https://developers.google.com/freebase/

training supervised models.

Appendix Supplemental Material for Chapter 5

1 Missing Proofs

1.1 Proof of Proposition 1

Proof. For any two supersets $H_1 \neq H_2$ of H whose oracles W_{H_1}, W_{H_2} are complete, $W_{H_1 \cap H_2}$, on a smaller set $H_1 \cap H_2$, is also complete (by definition). $\square$

1.2 Proof of Proposition 2

Proof. For any $(a, b), (b, c) \in H \cap G$, because W_H is complete, $(a, c) \in H \cap G$ (by Definition 9 (i)). For any $(a, b) \in H \cap G$, $(b, c) \in H \cap G^c \subseteq (H \cap G)^c$ (by Definition 9 (iv)). Therefore, $H \cap G$ is also a strict order of V (by Definition 6). $\square$

1.3 Proof of Theorem 1

1.3.1 Well-definiteness

It is trivial that if W_H is complete, $G \cap (H \cup N_{(x,y)})$ is also a strict order. Therefore, $N''_{(c,d)}$ is well defined, so is $O_{(x,y)}$.

1.3.2 Necessity

One can easily verify that if $(x, y) \in G \cap H^c$, both $N_{(x,y)} \subseteq \overline{H'}$ (Definition 8 (i)), $R_{(x,y)} \subseteq \overline{H'}$ (Definition 8 (iv)), and $S_{(x,y)} \cup T_{(x,y)} \cup O_{(x,y)} \subseteq \overline{H'}$ (Definition 8 (ii),(iii)) from the definition of closure, and likewise if $(x, y) \notin G$, $N'_{(x,y)} \subseteq \overline{H'}$ (Definition 8 (ii), (iii)). In another word, $C_{(x,y)}(H) \subseteq \overline{H'}$. Also, see Fig. 5.1 for the explanation of each necessary condition mentioned.

1.3.3 Sufficiency

One can see $A^G_{a'} \subseteq A^G_a$ if $a' \in A^G_a$ and $D^G_{a'} \subseteq D^G_a$ if $a' \in D^G_a$. That is, an ancestor of ancestor is also an ancestor (briefly, AAA), and a descendant of descendant is also a descendant (briefly, DDD).

Now we proceed to prove $C_{(x,y)}$ is complete using contradiction which finalizes the proof of our result $C_{(x,y)} = \overline{H'}$. If $C_{(x,y)}$ is not complete, by definition, one of the four conditions in Definition 8 must fail.

If Definition 8 (i) fails, there must exist a, b, c such that $(a, b) \in C_{(x,y)} \cap G$, $(b, c) \in C_{(x,y)} \cap G$, while $(a, c) \notin C_{(x,y)}$. In this case, if both (a, b) and (b, c) are from H, because H is complete, $(a, c) \in H \cap G$ contradicts the assumption. Hence at least one of (a, b) and (b, c) is not included in H. By the definition of $C_{(x,y)}$, if one pair belongs to $G \cap H^c$, it must come from $N_{(x,y)}$. Therefore, it implies $(x, y) \in G \cap H^c$.

$\underline{\text{Cases 1: If } (a, b) \in N_{(x,y)} \text{ and } (b, c) \in N_{(x,y)}}$, $(y, b) \in G$ and $(b, x) \in G$. That however implies $(y, x) \in G$, contradicting G's definition as a strict order (See Definition 6 (ii)).

$\underline{\text{Cases 2: If } (a, b) \in N_{(x,y)} \text{ and } (b, c) \in H}$, $a \in A^{G\cap H}_x$ and $c \in D^{G\cap H}_y$ (by DDD). It implies $(a, c) \in N_{(x,y)} \subseteq C_{(x,y)}$.

$\underline{\text{Cases 3: If } (a, b) \in H \text{ and } (b, c) \in N_{(x,y)}}$, $a \in A^{G\cap H}_x$ (by AAA) and $c \in D^{G\cap H}_y$. It implies $(a, c) \in N_{(x,y)} \subseteq C_{(x,y)}$.

In summary, Definition 8 (i) holds for $C_{(x,y)}$.

If Definition 8 (ii) fails, there must exist a, b, c such that $(a, b) \in C_{(x,y)} \cap G = G \cap (H \cup N_{(x,y)})$, $(a, c) \in C_{(x,y)} \cap G^c$, while $(b, c) \notin C_{(x,y)}$. In this case, if both (a, b) and (a, c) are from H, because H is complete, $(b, c) \in H \cap G^c$ contradicts the assumption. Hence *at least one of (a, b) and (a, c) is not included in H*. We divide the statement into the following cases to discuss:

$\underline{\text{Cases 1: If } (x, y) \in G,\ (a, b) \in N_{(x,y)}, \text{ and }}$ $\underline{(a, c) \in H \cap G^c}$, $(a, x) \in G \cap H$ and $(y, b) \in G \cap H$. Because $(a, x) \in H \cap G$, and $(a, c) \in H \cap G^c$, $(x, c) \in H \cap G^c$. Because $\{(x, y), (y, b)\} \subseteq G \cap (H \cup N_{(x,y)})$, $(x, b) \in G \cap (H \cup N_{(x,y)})$. Therefore, $(b, c) \in N''_{(x,c)} \subseteq C_{(x,y)}$.

Cases 2: If $(x,y) \in G$ and $(a,c) \in R_{(x,y)}$, $(c,a) \in N_{(x,y)}$, thus $(c,b) \in N_{(x,y)}$. It implies $(b,c) \in R_{(x,y)} \subseteq C_{(x,y)}$.

Cases 3: If $(x,y) \in G$ and $(a,c) \in S_{(x,y)}$, $(y,a) \in G \cap H$ and $\exists (d,c) \in G^c \cap H$ such that $(d,x) \in G \cap H$. Thus, $(x,c) \in G^c \cap H \Rightarrow (y,c) \in S_{(x,y)} \Rightarrow (b,c) \in N''_{(y,c)} \subseteq C_{(x,y)}$.

Cases 4: If $(x,y) \in G$ and $(a,c) \in T_{(x,y)}$, $(c,x) \in G \cap H$ and $\exists (a,d) \in G^c \cap H$ such that $(y,d) \in G \cap H$. Thus, $(a,y) \in G^c \cap H \Rightarrow (a,x) \in T_{(x,y)} \Rightarrow (a,b) \notin N_{(x,y)}$. Because $(a,b) \in G \cap (H \cup N_{(x,y)})$, one has $(a,b) \in G \cap H \Rightarrow (b,x) \in T_{(x,y)} \Rightarrow (b,c) \in N''_{(b,x)} \subseteq C_{(x,y)}$.

Cases 5: If $(x,y) \in G$ and $(a,c) \in O_{(x,y)}$, there exists $(d,e) \in S_{(a,b)} \cup T_{(a,b)}$ such that $(a,c) \in N''_{(d,e)}$. Therefore, $\{(a,b),(d,a),(c,e)\} \subseteq G \cap (H \cup N_{(x,y)})$. Hence, $(b,c) \in N''_{(d,e)} \subseteq C_{(x,y)}$.

Cases 6: If $(x,y) \in G^c$, we have $(a,b) \in G \cap H$ and $(a,c) \in N'_{(x,y)}$. Thus $(x,b) \in G \cap H \Rightarrow (b,c) \in N'_{(x,y)} \subseteq C_{(x,y)}$.

In summary, all six cases above contradict the assumption $(b,c) \notin C_{(x,y)}$. Thus Definition 8 (ii) holds for $C_{(x,y)}$. Given we have verified the condition of Definition 3 (ii), one can also prove that Definition 3 (iii) holds in a similar way, because their statements as well as definitions of $S_{(x,y)}$ and $T_{(x,y)}$ are symmetric.

One can easily see that Definition 3 (iv) holds for $C_{(x,y)}$, because $G \cap (H \cup N_{(x,y)})$ is also a strict order and $R_{(x,y)} \subseteq C_{(x,y)}$. □

1.4 Proof of Proposition 3

Proof. If $(c,d) \in S_{(a,b)}$, $c \in D_b^{G\cap H} \cup \{b\}$. then $c \notin A_a^{G\cap H} \cup \{a\}$, thus $D_c^{G\cap(H\cup N_{(a,b)})} = D_c^{G\cap H}$ (by Definition of $N_{(a,b)}$). It holds that $D_c^{G\cap(H\cup N_{(a,b)})} \cup \{c\} \subseteq D_b^{G\cap H} \cup \{b\}$. Therefore, one has $N''_{(c,d)} \subseteq N''_{(b,d)} \subseteq O_{(a,b)}$ if $(c,d) \in S_{(a,b)}$. Likewise, $N''_{(c,d)} \subseteq N''_{(c,a)} \subseteq O_{(a,b)}$ if $(c,d) \in T_{(a,b)}$. One has

$$O_{(a,b)} = \left[\bigcup_{(c,d)\in S_{(a,b)}} N''_{(b,d)} \right] \cup \left[\bigcup_{(c,d)\in T_{(a,b)}} N''_{(c,a)} \right],$$

whose time complexity is $O(|H|^3)$. □

1.5 Proof of Proposition 4

Proof. Let $(c', d') \in N''_{(c,d)}$. If $d' \neq d$, $(d', d) \in G \cap (H \cup N_{(a,b)})$. Then, $A_{d'}^{G \cap (H \cup N_{(a,b)})} \subseteq A_{d}^{G \cap (H \cup N_{(a,b)})}$ (AAA). Thus,

$$A_{d'}^{G \cap (H \cup N_{(a,b)})} \cup \{d'\} \subseteq A_{d}^{G \cap (H \cup N_{(a,b)})} \cup \{d\}.$$

Likewise,

$$D_{c'}^{G \cap (H \cup N_{(a,b)})} \cup \{c'\} \in D_{c}^{G \cap (H \cup N_{(a,b)})} \cup \{c\}.$$

By the definition of $N''_{(c',d')}$ and $N''_{(c,d)}$, one has $N''_{(c',d')} \subseteq N''_{(c,d)}$. □

1.6 Proof of Theorem 2

We first introduce the notion of *transitive reduction* before we proceed:

Definition 12 (Transitive Reduction [6]). *Let G be a directed acyclic graph. We say $\underline{G}$ is a transitive reduction of G if:*

(i) There is a directed path from vertex u to vertex v in $\underline{G}$ iff there is a directed path from u to v in G, and

(ii) There is no graph with fewer arcs than $\underline{G}$ satisfying (i).

For directed acyclic graph G, Aho et al. [6] have shown that the transitive reduction is unique and is a subgraph of G. Let G be a simple directed acyclic graph (DAG). In compliance with Def. 9, we use $\overline{G}$ to denote the transitive closure of G. Define $S(G)$ as the set of graphs such that every graph in $S(G)$ has the same transitive closure as G, i.e.,

$$S(G) \coloneqq \{G' \mid \overline{G'} = \overline{G}\}$$

Aho et al. [6] have shown that $S(G)$ is closed under intersection and union. Further more, for DAG G, the following relationship holds:

$$\underline{G} = \bigcap_{G' \in S(G)} G' \subseteq G \subseteq \bigcup_{G' \in S(G)} G' = \overline{G}$$

Next, we give the proof of Theorem 2 below.

Proof. With the fact that negative labels from the query oracle cannot help to induce positive labels in the graph, we can bound the number of queries, m, needed to learn a classifier:

$$m \geq |\underline{G}|$$

The proof is by simple contradiction based on the definition of the transitive reduction of G and the fact that A is a consistent learner. On the other hand, there is a learning algorithm A that simply remembers all the queries with positive labels and predict all the other inputs as negative. For this algorithm A, it suffices for A to make $|\overline{G}|$ queries. □

2 Experiment Environment

All experiments are conducted on an Ubuntu 14.04 server with 256GB RAM and 32 Intel Xeon E5-2630 v3 @ 2.40GHz processors. Active learning query strategies are implemented in Python2.7. Code and data will be publicly available.

Bibliography

[1] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *OSDI* (2016), vol. 16, pp. 265–283.

[2] Agarwal, M., and Mannem, P. Automatic gap-fill question generation from text books. In *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications, BEA@ACL 2011* (2011), ACL, pp. 56–64.

[3] Agichtein, E., Brill, E., and Dumais, S. Improving web search ranking by incorporating user behavior information. In *Proceedings of SIGIR* (2006), pp. 19–26.

[4] Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paşca, M., and Soroa, A. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2009), Association for Computational Linguistics, pp. 19–27.

[5] Agrawal, R., Golshan, B., and Papalexakis, E. Data-driven synthesis of study plans. Tech. Rep. TR-2015-003, Data Insights Laboratories, 2015.

[6] Aho, A. V., Garey, M. R., and Ullman, J. D. The transitive reduction of a directed graph. *SIAM Journal on Computing 1*, 2 (1972), 131–137.

[7] Aleven, V. A., and Koedinger, K. R. An effective metacognitive strategy: Learning by doing and explaining with a computer-based cognitive tutor. *Cognitive science 26*, 2 (2002), 147–179.

[8] Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Raths, J., and Wittrock, M. C. A taxonomy for learning, teaching, and assessing: A revision of bloomâĂŹs taxonomy of educational objectives, abridged edition. *White Plains, NY: Longman* (2001).

[9] Angluin, D. Queries and concept learning. *Machine learning 2*, 4 (1988), 319–342.

[10] Balcan, M.-F., Broder, A., and Zhang, T. Margin based active learning. In *International Conference on Computational Learning Theory* (2007), Springer, pp. 35–50.

[11] Becker, L., Basu, S., and Vanderwende, L. Mind the gap: learning to choose gaps for question generation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2012), ACL, pp. 742–751.

[12] Bergan, J. R., and Jeska, P. An examination of prerequisite relations, positive transfer among learning tasks, and variations in instruction for a seriation hierarchy. *Contemporary Educational Psychology 5*, 3 (1980), 203–215.

[13] Bird, S., and Loper, E. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions* (2004), ACL, p. 31.

[14] Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. *Journal of Machine Learning Research 3* (2003), 993–1022.

[15] Bonk, C. J., Lee, M. M., Kim, N., and Lin, M.-F. G. The tensions of transformation in three cross-institutional wikibook projects. *The Internet and Higher Education* (2009), 126–135.

[16] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems* (2013), pp. 2787–2795.

[17] Bordes, A., Weston, J., Collobert, R., and Bengio, Y. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011,* (2011).

[18] Breiman, L. Random forests. *Machine learning 45*, 1 (2001), 5–32.

[19] Buckley, C., Salton, G., Allan, J., and Singhal, A. Automatic query expansion using SMART: TREC 3. In *Proceedings of TREC* (1994), pp. 69–80.

[20] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning* (2005), ACM, pp. 89–96.

[21] Burges, C. J. From ranknet to lambdarank to lambdamart: An overview. Tech. rep., June 2010.

[22] Cai, L., and Wang, W. Y. Kbgan: Adversarial learning for knowledge graph embeddings. In *NAACL* (2018), ACL.

[23] Carterette, B., Allan, J., and Sitaraman, R. Minimal test collections for retrieval evaluation. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (2006), ACM, pp. 268–275.

[24] Chattopadhyay, R., Wang, Z., Fan, W., Davidson, I., Panchanathan, S., and Ye, J. Batch mode active sampling based on marginal probability distribution matching. *ACM Transactions on Knowledge Discovery from Data (TKDD) 7*, 3 (2013), 13.

[25] Chen, C.-Y., Liou, H.-C., and Chang, J. S. Fast: an automatic generation system for grammar tests. In *Proceedings of the COLING/ACL on Interactive presentation sessions* (2006), ACL, pp. 1–4.

[26] Chen, T., and Guestrin, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (2016), ACM, pp. 785–794.

[27] Chen, W., Aist, G., and Mostow, J. Generating questions automatically from informational text. In *Proc. of the 2nd Workshop on Question Generation* (2009), pp. 17–24.

[28] Chen, Y., Wuillemin, P.-H., and Labat, J.-M. Discovering prerequisite structure of skills through probabilistic association rules mining. *International Educational Data Mining Society* (2015).

[29] Chu, W., and Ghahramani, Z. Extensions of gaussian processes for ranking: semisupervised and active learning. *Learning to Rank* (2005), 29.

[30] Cohn, D. A., Ghahramani, Z., and Jordan, M. I. Active learning with statistical models. *Journal of Artificial Intelligence Research 4*, 1 (1996), 129–145.

[31] Coniam, D. A preliminary inquiry into using corpus word frequency data in the automatic generation of english language cloze tests. *Calico Journal 14*, 2-4 (1997), 15–33.

[32] Corbett, A. T., Koedinger, K. R., and Anderson, J. R. Intelligent tutoring systems. In *Handbook of Human-Computer Interaction (Second Edition)*. Elsevier, 1997, pp. 849–874.

[33] Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[34] Croft, W., and Cruse, D. A. *Cognitive linguistics*. Cambridge University Press, 2004.

[35] Curto, S., Mendes, A. C., and Coheur, L. Exploring linguistically-rich patterns for question generation. In *Proc. of the UCNLG+ eval: Language Generation and Evaluation Workshop* (2011), ACL, pp. 33–38.

[36] Dagan, I., and Engelson, S. P. Committee-based sampling for training probabilistic classifiers. In *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995* (1995), pp. 150–157.

[37] Dasgupta, S., and Hsu, D. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning* (2008), ACM, pp. 208–215.

[38] Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*, 6 (1990), 391–407.

[39] Donmez, P., and Carbonell, J. G. Optimizing estimated loss reduction for active sampling in rank learning. In *Proceedings of the 25th international conference on Machine learning* (2008), ACM, pp. 248–255.

[40] Donmez, P., and Carbonell, J. G. Active sampling for rank learning via optimizing the area under the roc curve. In *European Conference on Information Retrieval* (2009), Springer, pp. 78–89.

[41] Donmez, P., Carbonell, J. G., and Bennett, P. N. Dual strategy active learning. In *European Conference on Machine Learning* (2007), Springer, pp. 116–127.

[42] Du, X., Shao, J., and Cardie, C. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers* (2017), pp. 1342–1352.

[43] Evanini, K., and Wang, X. Automatic detection of plagiarized spoken responses. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications* (2014), Association for Computational Linguistics, pp. 22–27.

[44] Fleiss, J. L. Measuring nominal scale agreement among many raters. *Psychological bulletin 76*, 5 (1971), 378.

[45] Freund, Y., Seung, H. S., Shamir, E., and Tishby, N. Selective sampling using the query by committee algorithm. *Machine Learning 28*, 2-3 (1997), 133–168.

[46] Fu, R., Guo, J., Qin, B., Che, W., Wang, H., and Liu, T. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (2014), vol. 1.

[47] Fu, R., Qin, B., and Liu, T. Exploiting multiple sources for open-domain hypernym discovery. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (2013), pp. 1224–1234.

[48] Gabrilovich, E., and Markovitch, S. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (2007), vol. 7, pp. 1606–1611.

[49] Gardner, M., and Mitchell, T. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (2015), pp. 1488–1498.

[50] Getoor, L., and Taskar, B. Introduction to statistical relational learning.

[51] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *NIPS* (2014), pp. 2672–2680.

[52] Goodrich, H. C. Distractor efficiency in foreign language testing. *TESOL Quarterly* (1977), 69–78.

[53] Gordon, J., Zhu, L., Galstyan, A., Natarajan, P., and Burns, G. Modeling concept dependencies in a scientific corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers* (2016).

[54] Guo, Q., Kulkarni, C., Kittur, A., Bigham, J. P., and Brunskill, E. Questimator: Generating knowledge assessments for arbitrary topics. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (2016), pp. 3726–3732.

[55] Guo, Y. Active instance sampling via matrix partition. In *Advances in Neural Information Processing Systems* (2010), pp. 802–810.

[56] Guo, Y., and Schuurmans, D. Discriminative batch mode active learning. In *Advances in neural information processing systems* (2008), pp. 593–600.

[57] Halawa, S., Greene, D., and Mitchell, J. Dropout prediction in moocs using learner activity features. *Experiences and best practices in and around MOOCs 7* (2014), 3–12.

[58] Harley, J. M., Trevors, G. J., Azevedo, R., et al. Clustering and profiling students according to their interactions with an intelligent tutoring system fostering self-regulated learning. *JEDM| Journal of Educational Data Mining 5*, 1 (2013), 104–146.

[59] Hassan, S., and Mihalcea, R. Semantic relatedness using salient semantic analysis. In *Proceedings of AAAI* (2011).

[60] Hearst, M. A. Automatic acquisition of hyponyms from large text corpora. In *14th International Conference on Computational Linguistics, COLING 1992* (1992), ACL, pp. 539–545.

[61] Heilman, M., and Smith, N. A. Question generation via overgenerating transformations and ranking. Tech. rep., DTIC Document, 2009.

[62] Heilman, M., and Smith, N. A. Good question! statistical ranking for question generation. In *The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2010), ACL, pp. 609–617.

[63] Hill, J., and Simha, R. Automatic generation of context-based fill-in-the-blank exercises using co-occurrence likelihoods and google n-grams. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications* (2016), ACL, pp. 23–30.

[64] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[65] Hofmann, T. Probabilistic latent semantic indexing. In *Proceedings of SIGIR* (1999), ACM, pp. 50–57.

[66] Huang, C.-T., Lin, W.-T., Wang, S.-T., and Wang, W.-S. Planning of educational training courses by data mining: Using china motor corporation as an example. *Expert systems with applications 36*, 3 (2009), 7199–7209.

[67] Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL* (2012), ACL, pp. 873–882.

[68] Huang, S., Jin, R., and Zhou, Z. Active learning by querying informative and representative examples. *IEEE Trans. Pattern Anal. Mach. Intell. 36*, 10 (2014), 1936–1949.

[69] Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *ICLR* (2017).

[70] Jiang, S., and Lee, J. Distractor generation for chinese fill-in-the-blank items. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications* (2017), ACL, pp. 143–148.

[71] Jiang, S., Lowd, D., and Dou, D. Learning to refine an automatically extracted knowledge base using markov logic. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on* (2012), IEEE, pp. 912–917.

[72] Joachims, T. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (2006), pp. 217–226.

[73] Joachims, T., Granka, L., Pan, B., Hembrooke, H., and Gay, G. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of SIGIR* (2005), pp. 154–161.

[74] Karamanis, N., Ha, L. A., and Mitkov, R. Generating multiple-choice test items from medical text: A pilot study. In *Proceedings of the Fourth International Natural Language Generation Conference* (2006), ACL, pp. 111–113.

[75] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. In *ICLR* (2015).

[76] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. In *ICLR* (2015).

[77] Kingma, D. P., and Welling, M. Auto-encoding variational bayes. In *ICLR* (2014).

[78] Kleinberg, J. M. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM) 46*, 5 (1999), 604–632.

[79] Kotlerman, L., Dagan, I., Szpektor, I., and Zhitomirsky-Geffet, M. Directional distributional similarity for lexical inference. *Natural Language Engineering 16*, 04 (2010), 359–389.

[80] Kumar, G., Banchs, R. E., and D'Haro, L. F. Revup: Automatic gap-fill question generation from educational texts. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications, BEA@NAACL-HLT 2015, June 4, 2015, Denver, Colorado, USA* (2015), pp. 154–161.

[81] Lao, N., and Cohen, W. W. Relational retrieval using a combination of path-constrained random walks. *Machine learning 81*, 1 (2010), 53–67.

[82] Laurence, S., and Margolis, E. Concepts and cognitive science. *Concepts: core readings* (1999), 3–81.

[83] Lee, C.-H., Lee, G.-G., and Leu, Y. Application of automatically constructed concept map of learning to conceptual diagnosis of e-learning. *Expert Systems with Applications 36*, 2 (2009), 1675–1684.

[84] Lee, J., and Seneff, S. Automatic generation of cloze items for prepositions. In *INTERSPEECH* (2007), pp. 2173–2176.

[85] Lenci, A., and Benotto, G. Identifying hypernyms in distributional semantic spaces. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation* (2012), Association for Computational Linguistics, pp. 75–79.

[86] Lewis, D. D., and Catlett, J. Heterogeneous uncertainty sampling for supervised learning. In *ICML* (1994), pp. 148–156.

[87] Lewis, D. D., and Gale, W. A. A sequential algorithm for training text classifiers. In *SIGIR Forum* (1994), pp. 3–12.

[88] Liang, C., Wang, S., Wu, Z., Williams, K., Pursel, B., Brautigam, B., Saul, S., Williams, H., Bowen, K., and Giles, C. Bbookx: An automatic book creation framework. In *The ACM Symposium on Document Engineering* (2015).

[89] Liang, C., Wang, S., Wu, Z., Williams, K., Pursel, B., Brautigam, B., Saul, S., Williams, H., Bowen, K., and Giles, C. L. Bbookx: Building online open books for personalized learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (2016), pp. 4369–4370.

[90] Liang, C., Wu, Z., Huang, W., and Giles, C. L. Measuring prerequisite relations among concepts. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (2015), Association for Computational Linguistics, pp. 1668–1674.

[91] Liang, C., Yang, X., Dave, N., Wham, D., Pursel, B., and Giles, C. L. Distractor generation for multiple choice questions using learning to rank. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications, BEA@NAACL* (2018), ACL, pp. 284–290.

[92] Liang, C., Yang, X., Wham, D., Pursel, B., Passonneau, R., and Giles, C. L. Distractor generation with generative adversarial nets for automatically creating fill-in-the-blank questions. In *Proceedings of the Knowledge Capture Conference* (2017), ACM, p. 33.

[93] Liang, C., Ye, J., Wang, S., Pursel, B., and Giles, C. L. Investigating active learning for concept prerequisite learning. *Proc. EAAI* (2018).

[94] Liang, C., Ye, J., Wu, Z., Pursel, B., and Giles, C. L. Recovering concept prerequisite relations from university course dependencies. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.* (2017), pp. 4786–4791.

[95] Liang, C., Ye, J., Zhao, H., Pursel, B., and Giles, C. L. Active learning of strict partial orders: A case study on concept prerequisite relations. *arXiv preprint arXiv:1801.06481* (2018).

[96] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015).

[97] Lin, Y.-C., Sung, L.-C., and Chen, M. C. An automatic multiple-choice question generation scheme for english adjective understanding. In *ICCE* (2007), pp. 137–142.

[98] Litman, D. J., and Silliman, S. Itspoke: An intelligent tutoring spoken dialogue system. In *Demonstration papers at HLT-NAACL 2004* (2004), Association for Computational Linguistics, pp. 5–8.

[99] Liu, H., Ma, W., Yang, Y., and Carbonell, J. Learning concept graphs from online educational data. *Journal of Artificial Intelligence Research 55* (2016), 1059–1090.

[100] Liu, J., Jiang, L., Wu, Z., Zheng, Q., and Qian, Y. Mining learning-dependency between knowledge units from text. *The VLDB Journal 20*, 3 (2011), 335–345.

[101] Liu, L., Vernica, R., Hassan, T., Venkata, N. D., Lei, Y., Fan, J., Liu, J., Simske, S. J., and Wu, S. Metis: A multi-faceted hybrid book learning platform. In *Proceedings of the 2016 ACM Symposium on Document Engineering* (2016).

[102] Liu, T.-Y. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval 3*, 3 (2009), 225–331.

[103] Long, B., Chapelle, O., Zhang, Y., Chang, Y., Zheng, Z., and Tseng, B. Active learning for ranking through expected loss optimization. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval* (2010), ACM, pp. 267–274.

[104] Lykourentzou, I., Giannoukos, I., Nikolopoulos, V., Mpardis, G., and Loumos, V. Dropout prediction in e-learning courses through the combination of machine learning techniques. *Computers & Education 53*, 3 (2009), 950–965.

[105] Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing* (2013).

[106] Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR* (2017).

[107] Maddison, C. J., Tarlow, D., and Minka, T. A* sampling. In *NIPS* (2014), pp. 3086–3094.

[108] Mamitsuka, N. A. H. Query learning strategies using boosting and bagging. In *Machine learning: proceedings of the fifteenth international conference (ICML'98)* (1998), vol. 1, Morgan Kaufmann Pub.

[109] McNamee, P., Snow, R., Schone, P., and Mayfield, J. Learning named entity hyponyms for question answering. In *Third International Joint Conference on Natural Language Processing, IJCNLP 2008* (2008), pp. 799–804.

[110] Medelyan, O., Frank, E., and Witten, I. H. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of EMNLP* (2009), pp. 1318–1327.

[111] Mihalcea, R., and Csomai, A. Wikify!: linking documents to encyclopedic knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management* (2007), pp. 233–242.

[112] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS* (2013), pp. 3111–3119.

[113] Miller, G. A. Wordnet: a lexical database for english. *Communications of the ACM 38*, 11 (1995), 39–41.

[114] Milne, D., and Witten, I. H. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management* (2008), pp. 509–518.

[115] Milne, D., and Witten, I. H. An open-source toolkit for mining wikipedia. *Artificial Intelligence 194* (2013), 222–239.

[116] Mintz, M., Bills, S., Snow, R., and Jurafsky, D. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP* (2009), ACL, pp. 1003–1011.

[117] Mirza, M., and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[118] Mitkov, R., and Ha, L. A. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing-Volume 2* (2003), Association for Computational Linguistics, pp. 17–22.

[119] Mitkov, R., Ha, L. A., and Karamanis, N. A computer-aided environment for generating multiple-choice test items. *Natural language engineering 12*, 2 (2006), 177–194.

[120] Napoles, C., and Callison-Burch, C. Systematically adapting machine translation for grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications* (Copenhagen, Denmark, September 2017), Association for Computational Linguistics, Association for Computational Linguistics, pp. 345–356.

[121] Nguyen, H. T., and Smeulders, A. Active learning using pre-clustering. In *Proceedings of the twenty-first international conference on Machine learning* (2004), ACM, p. 79.

[122] Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE 104*, 1 (2016), 11–33.

[123] Ohland, M. W., Yuhasz, A. G., and Sill, B. L. Identifying and removing a calculus prerequisite as a bottleneck in clemson's general engineering curriculum. *Journal of Engineering Education 93*, 3 (2004), 253–257.

[124] Page, L., Brin, S., Motwani, R., and Winograd, T. The pagerank citation ranking: Bringing order to the web. Tech. rep., Stanford InfoLab, 1999.

[125] Pan, L., Li, C., Li, J., and Tang, J. Prerequisite relation learning for concepts in moocs. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2017), vol. 1, pp. 1447–1456.

[126] Pardos, Z. A., Heffernan, N. T., Anderson, B., and Heffernan, C. L. The effect of model granularity on student performance prediction using bayesian networks. In *International Conference on User Modeling* (2007), Springer, pp. 435–439.

[127] Parikh, A., Täckström, O., Das, D., and Uszkoreit, J. A decomposable attention model for natural language inference. In *EMNLP* (2016), ACL, pp. 2249–2255.

[128] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *Journal of machine learning research 12*, Oct (2011), 2825–2830.

[129] Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), ACL, pp. 1532–1543.

[130] Pino, J., and Eskenazi, M. Semi-automatic generation of cloze question distractors effect of students' l1. In *International Workshop on Speech and Language Technology in Education* (2009), pp. 65–68.

[131] Pino, J., Heilman, M., and Eskenazi, M. A selection strategy to improve cloze question quality. In *Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains. 9th International Conference on Intelligent Tutoring Systems.* (2008), pp. 30–42.

[132] Pujara, J., Miao, H., Getoor, L., and Cohen, W. Knowledge graph identification. In *International Semantic Web Conference* (2013), Springer, pp. 542–557.

[133] Pursel, B., Liang, C., Wang, S., Wu, Z., Williams, K., Brautigam, B., Saul, S., Williams, H., Bowen, K., and Giles, C. L. Bbookx: Design of an automated web-based recommender system for the creation of open learning content. In *Proceedings of the 25th International Conference Companion on World Wide Web* (2016), International World Wide Web Conferences Steering Committee, pp. 929–933.

[134] Radinsky, K., Agichtein, E., Gabrilovich, E., and Markovitch, S. A word at a time: computing word relatedness using temporal semantic

analysis. In *Proceedings of the 20th international conference on World wide web* (2011), ACM, pp. 337–346.

[135] Ramachandran, L., Cheng, J., and Foltz, P. Identifying patterns for short answer scoring using graph-based lexico-semantic text matching. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications* (2015), pp. 97–106.

[136] Ramesh, A., Goldwasser, D., Huang, B., Daumé III, H., and Getoor, L. Modeling learner engagement in moocs using probabilistic soft logic. In *NIPS Workshop on Data Driven Education* (2013), vol. 21, p. 62.

[137] Ratinov, L., Roth, D., Downey, D., and Anderson, M. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1* (2011), ACL, pp. 1375–1384.

[138] Rendle, S., and Schmidt-Thieme, L. Active learning of equivalence relations by minimizing the expected loss using constraint inference. In *IEEE International Conference on Data Mining* (2008), IEEE, pp. 1001–1006.

[139] Resnik, P. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes* (1995), pp. 448–453.

[140] Ritter, A., Soderland, S., and Etzioni, O. What is this, anyway: Automatic hypernym discovery. In *AAAI Spring Symposium: Learning by Reading and Learning to Read* (2009), pp. 88–93.

[141] Roy, N., and McCallum, A. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown* (2001), 441–448.

[142] Sakaguchi, K., Arase, Y., and Komachi, M. Discriminative approach to fill-in-the-blank quiz generation for language learners. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (2013), vol. 2, ACL, pp. 238–242.

[143] Salton, G., and Buckley, C. Improving retrieval performance by relevance feedback. *Readings in information retrieval 24*, 5 (1997), 355–363.

[144] Scheines, R., Silver, E., and Goldin, I. Discovering prerequisite relationships among knowledge components. In *Proceedings of Educational Data Mining* (2014), pp. 355–356.

[145] Settles, B. Active learning literature survey. *University of Wisconsin, Madison 52*, 55-66 (2010), 11.

[146] Seung, H. S., Opper, M., and Sompolinsky, H. Query by committee. In *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992* (1992), ACM, pp. 287–294.

[147] Shams, R., and Mercer, R. E. Investigating keyphrase indexing with text denoising. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries* (2012), pp. 263–266.

[148] Socher, R., Chen, D., Manning, C. D., and Ng, A. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems* (2013), pp. 926–934.

[149] Stasaski, K., and Hearst, M. A. Multiple choice question generation utilizing an ontology. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications* (2017), ACL, pp. 303–312.

[150] Sumita, E., Sugaya, F., and Yamamoto, S. Measuring non-native speakers' proficiency of english by using a test with automatically-generated fill-in-the-blank questions. In *Proceedings of the second workshop on Building Educational Applications Using NLP* (2005), ACL, pp. 61–68.

[151] Talukdar, P. P., and Cohen, W. W. Crowdsourced comprehension: predicting prerequisite structure in Wikipedia. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP* (2012), ACL, pp. 307–315.

[152] Tong, S., and Koller, D. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research 2* (2001), 45–66.

[153] Vajjala, S., and Meurers, D. 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* (2012), Association for Computational Linguistics, pp. 163–173.

[154] VanLehn, K. Student modeling. *Foundations of intelligent tutoring systems 55* (1988), 78.

[155] Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2001), vol. 1, IEEE, pp. 511–518.

[156] Vuong, A., Nixon, T., and Towle, B. A method for finding prerequisites within a curriculum. In *Educational Data Mining* (2010).

[157] Wang, J., Yu, L., Zhang, W., Gong, Y., Xu, Y., Wang, B., Zhang, P., and Zhang, D. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *SIGIR* (2017), ACM, pp. 515–524.

[158] Wang, S., Liang, C., Wu, Z., Williams, K., Pursel, B., Brautigam, B., Saul, S., Williams, H., Bowen, K., and Giles, C. Concept hierarchy extraction from textbooks. In *The ACM Symposium on Document Engineering* (2015).

[159] Wang, S., Ororbia, A., Wu, Z., Williams, K., Liang, C., Pursel, B., and Giles, C. L. Using prerequisites to extract concept maps fromtextbooks. In *Proc. CIKM* (2016), ACM, pp. 317–326.

[160] Wang, Z., and Ye, J. Querying discriminative and representative samples for batch mode active learning. *ACM Transactions on Knowledge Discovery from Data (TKDD) 9*, 3 (2015), 17.

[161] Wang, Z., Zhang, J., Feng, J., and Chen, Z. Knowledge graph embedding by translating on hyperplanes. In *AAAI* (2014), vol. 14, pp. 1112–1119.

[162] Welbl, J., Liu, N. F., and Gardner, M. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text, NUT@EMNLP* (2017), ACL, pp. 94–106.

[163] White, R. W., Ruthven, I., and Jose, J. M. The use of implicit evidence for relevance feedback in web retrieval. In *Advances in Information Retrieval.* Springer, 2002, pp. 93–109.

[164] Witten, I., and Milne, D. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy* (2008), pp. 25–30.

[165] Wojatzki, M., Melamud, O., and Zesch, T. Bundled gap filling: A new paradigm for unambiguous cloze exercises. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications* (2016), pp. 172–181.

[166] Wolfe, J. H. Automatic question generation from text-an aid to independent study. *ACM SIGCSE Bulletin 8*, 1 (1976), 104–112.

[167] Wu, Z., and Giles, C. L. Sense-aaware semantic analysis: A multi-prototype word representation model using wikipedia. In *AAAI* (2015), pp. 2188–2194.

[168] Xu, Z., Yu, K., Tresp, V., Xu, X., and Wang, J. Representative sampling for text classification using support vector machines. In *European Conference on Information Retrieval* (2003), Springer, pp. 393–407.

[169] Yang, Y., Liu, H., Carbonell, J., and Ma, W. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining* (2015), pp. 159–168.

[170] Yu, L., Zhang, W., Wang, J., and Yu, Y. Seqgan: sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[171] Zesch, T., and Melamud, O. Automatic generation of challenging distractors using context-sensitive inference rules. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications* (2014), ACL, pp. 143–148.

[172] Zesch, T., Müller, C., and Gurevych, I. Using wiktionary for computing semantic relatedness. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008* (2008), pp. 861–866.

[173] Zhou, Q., Yang, N., Wei, F., Tan, C., Bao, H., and Zhou, M. Neural question generation from text: A preliminary study. In *Natural Language Processing and Chinese Computing - 6th CCF International Conference, NLPCC 2017, Dalian, China, November 8-12, 2017, Proceedings* (2017), pp. 662–671.

Vita

Chen Liang

Education

The Pennsylvania State University, University Park, PA, USA
Ph.D., Information Sciences and Technology, 2018

Tsinghua University, Beijing, China
B.S., Computer Science and Technology, 2013

Experience

Facebook, Software Engineering Intern, Summer 2017

Bosch Research and Technology Center, Research Intern, Summer 2016

Qatar Computing Research Institute, Research Associate, Summer 2015

Tsinghua University, Research Assistant, July 2011 - July 2013

Selected Publications

Liang, Chen, et al. "Distractor Generation for Multiple Choice Questions Using Learning to Rank." In *Proc. BEA@NAACL'18.*

Liang, Chen, et al. "Active Learning of Strict Partial Orders: A Case Study on Concept Prerequisite Relations." arXiv:1801.06481 [cs.LG], January 2018

Liang, Chen, et al. "Investigating Active Learning for Concept Prerequisite Learning." In *Proc. EAAI'18.*

Liang, Chen, et al. "Distractor Generation with Generative Adversarial Nets for Automatically Creating Fill-in-the-blank Questions." In *Proc. K-CAP'17.*

Liang, Chen, et al. "Recovering Concept Prerequisite Relations from University Course Dependencies" In *Proc. EAAI'17*

Liang, Chen, et al. "BBookX: Building Online Open Books for Personalized Learning." In *Proc. AAAI'16.*

Liang, Chen, et al. "Measuring Prerequisite Relations Among Concepts." In *Proc. EMNLP'15.*

Liang, Chen, et al. "BBookX: An Automatic Book Creation Framework." In *Proc. DocEng'15.*