Predicting Future Student Performance on Intelligent Tutoring Systems

Personalizing Educational Curriculum for Young Students

Qi Pan

Submitted in partial fulfillment of the requirements for Bachelors of Applied Sciences in Industrial Engineering at The University of Toronto

Department of Mechanical & Industrial Engineering
University of Toronto
Canada
April 13th, 2017

Abstract

Technical tools like personal computers are becoming commonplace in education and are often an integral part to learning nowadays. Intelligent Tutoring Systems (ITS) are automated programs designed to help students identify their weaknesses in the subject matter they are learning. At the heart of a good ITS is a good internal predictive model which can infer student knowledge on subject matter subtopics by accurately predicting future student answer correctness. In this thesis, an ITS dataset from the 2010 KDD Data Mining Competition is used to showcase a novel testing paradigm that mimics real life ITS use. Models used to showcase the paradigm are built on Nearest Neighbors based Collaborative Filtering (NN based CF) methods which were shown to have a performance of $\approx +2$ percentage points of accuracy vs. baseline macro averages. On top of having above baseline performance, further analysis showed that encoding incorrect answers as negative and separate from absent answers was crucial to any NN based CF methods' success. Additionally, having high recall on incorrects and factoring in the volume of questions answered were also important to NN based CF methods' successful performance in predicting ITS student correctness under the new testing paradigm.

Acknowledgements

I'd like to thank my thesis supervisor Professor Scott Sanner for his patience, time, and dedication while guiding me through my first large research project. The honest and timely feedback helped give me perspective on what it means to be a researcher and scientist, something I couldn't have learned without a diligent and understanding advisor. Thank you to Rohn Jackson, Ashton Wu, Ashley Lawrence, Krystle Pang, and the rest of the FAST team for your support of my passions and ideas (including doing this thesis). I was truly spoiled to have such a wonderful and accepting team. I'd also like to thank Jade Wang for her unconditional support and consultation during this thesis, it helped me to push through the harsh times. Lastly I'd like to thank my parents without which I would not have the privilege of higher education and other awesome opportunities. Thank you for your sacrifices and care.

Declaration

I hereby declare that this thesis is my own work and that, to the best of my knowledge contains no material previously published or produced by other parties except where acknowledgement are made.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

When coming up with an educational curriculum, it is not easy to know what content to cover, the pace at which it should be taught, or even what to emphasize in order to optimally teach a class of students with diverse needs. Traditional curriculums are one size fits all, the whole class of students are exposed to the same course structure in a rigid inflexible manner that is completely based on prior assumptions of the students' knowledge levels. What is the best way to tailor the delivery of educational content to each student such that it is at their pace and focuses on their gaps of knowledge? How can we discover where each student stands in terms of their subject matter knowledge to give them the educational content they need?

With increasing accessibility to computing resources for students, it is only natural for technical solutions that tailor curriculum to individual students to become more commonplace (Dede 1996). Intelligent Tutoring Systems (ITS) are one of the many technologies supplementing traditional curriculums today. ITS are computer systems that provide customized instruction/feedback to students automatically; they are also the interface and medium through which student data on subject matter knowledge can be gathered by assessing students on practice problems. Consequently, ITS can then apply quantitative models to the collected data in order to predict future student performance and through these predictions, infer student knowledge levels and their similarity with their peers (Figure 1.1). Thus, how effective an ITS is at delivering the right content to students depends on the quality of their internal prediction models and how applicable they are to the subset of students using it.

Practically speaking, knowing what makes a good model for ITS can save students hours of ineffective studying time ($\approx 10\%$ time savings according to (Cen, Koedinger, and Junker 2006)) by modifying the standard curriculm to areas where they have gaps. Scientifically, a good predictive model can be a gateway into how humans learn and absorb information. This thesis addresses these important practical and scientific questions by focusing on developing a good ITS answer prediction model under realistic use cases where the predictions must be made at different, discrete time periods of the curriculum.

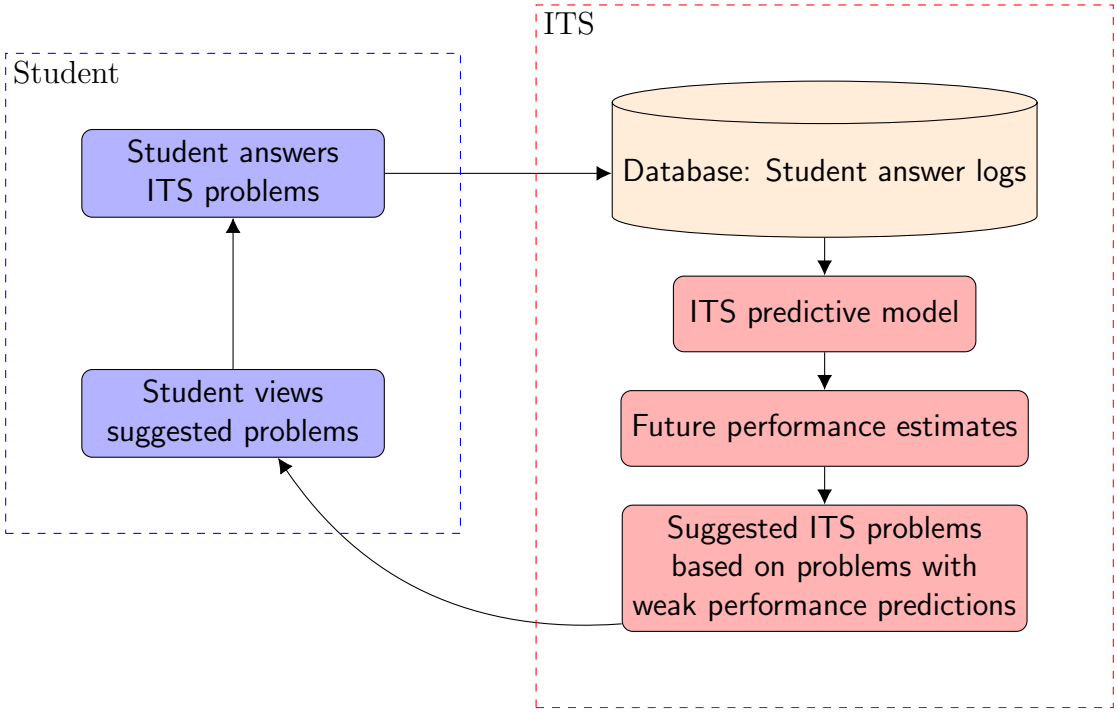

Figure 1.1: Overview: ITS interactions with students

1.2 Contributions

This thesis focuses on an ITS dataset that comes from the 2010 KDD Cup Educational Data Mining Challenge (Stamper et al. 2010). The goal of the original competition was to accurately predict future student performance (which is in line with our motivations) but competitors were given the whole semesters worth of data (minus a few weeks) to do so. While the contestants provided valuable insight for developing good ITS predictive models, their results do not correspond with the most realistic use case for ITS. ITS need to have predictions every day of the semester, which means the model needs to perform reasonably well during the early stages of the semester when students have provided less data to the ITS. In this thesis, the most realistic ITS use case is emulated by using a new testing paradigm. This testing paradigm discretizes the data and trains on subsets ranging from the beginning of the semester, to later on in the semester while predicting for the immediate questions that follow. To the best knowledge of this thesis' author, this testing framework is novel for ITS predictive models on the 2010 KDD Cup Educational Data Mining Challenge dataset.

Model wise, this thesis hones in on good Nearest Neighbors models for ITS predictions over multiple discrete time intervals. The Collaborative Filtering framework is leveraged on Nearest Neighbors methods to find similar students and cleverly make weighted predictions based on the similarity scores. The experiments resulted in a 2 percentage point improvement over the best baselines. Through the best model, it was discovered that encoding incorrect student answers as a "negative" towards their similarity score was more effective than treating them the same as unanswered (neutral). Not only are realistic testing paradigms that follow real ITS use cases presented in this thesis, but above baseline predictive models and feature engineering insights within the education domain have also been uncovered.

The rest of the report is organized as follows. Chapter 2 goes over the existing literature on Nearest Neighbors based Collaborative Filtering Recommendation models and does an exploratory data analysis on the specific data being used in this thesis. Chapter 3 formalizes some baseline methods to be used for comparison with Nearest Neighbors based Collaborative Filtering methods on the ITS dataset. Chapter 4 goes over the many variations of Nearest Neighbors based Collaborative Filtering methods experimented with in this paper. Chapter 5 elaborates on the empirical results obtained through the new testing paradigm and compares the baseline methods with the Nearest Neighbors methods. Some examples of student pairs who were considered not similar, somewhat similar, and very similar are also analyzed for further insight. Chapter 6 Summarizes the contributions of this thesis and talks about future work in ITS prediction using Nearest Neighbors based Collaborative Filtering.

Chapter 2

Background

2.1 Literature Review

Nearest Neighbors (NN) methods and Collaborative Filtering (CF) frameworks have been popularized in recent decades both independently and when utilized together. NN methods in general are nonparametric regression techniques that predict future responses based on the distance of the input to other samples in the feature space. When used for classification, the predicted class is often the majority vote of class labels from the closest samples in the feature space to the input. NN methods have been shown to be especially powerful when the regression function has a completely unknown form (Altman 1992) and have the additional advantage of being simple to describe to stakeholders or novice scholars.

While NN methods handle classification after the data has been encoded, the CF framework is a way to set up the feature space after data has been collected. Each user in a CF framework can be thought of as a feature vector of judgements/ratings where each element is a unique domain specific item (Table 2.1). To predict the judgement a user would give a specific item, a CF framework matches other users with similar feature vectors and uses their judgements on that specific item to predict for the target user. CF frameworks have seen success on internet based services like Amazon (Linden, Smith, and York 2003), typically when used for recommending new products based on user ratings. CF frameworks also provide the advantage of being able to recommend items to users that are not completely based on their own behavior, often surprising users with things they didn't think they were interested in, but end up finding useful/interesting.

	Prob-Step A	Prob-Step B	Prob-Step C	Prob-Step D	...
Student 1	1	?	?	0	...
Student 2	0	?	1	?	...
Student 3	?	?	0	?	...
Student 4	1	0	1	1	...
Student 5	1	1	?	0	...
...	...	...	...	...	...

Table 2.1: An example of a CF framework setup for an ITS problem recommender based on if students got historical problems correct (1) or incorrect (0). ? indicates the student has not attempted the problem-step yet and are what is to be predicted

One of the early supporting papers that showed the usefulness of combining CF frameworks with NN methods was (Herlocker et al. 1999). Herlocker showed positive empirical results when combing CF and NN methodologies to recommend within movies, videos, and music domains. Their results led them to believe that there is no reason a NN based CF recommend would not generalize to other domains (ie. recommending new problems for students to better target their weaknesses in the subject matter) but they did not have empirical results to prove it at the time. More recent and prolific examples of CF frameworks achieving success have been outlined in (Koren 2009). The competition winners validated the frameworks power by winning the Netflix prize challenge of accurately recommending movies to users based on vectors of ratings using various types of CF frameworks.

Some teams who competed in the 2010 KDD Cup also submitted reports on the methods they used to achieve the best predictions on the ITS dataset. The competition emphasized minimizing Root Mean Squared Error (RMSE) on a test set that contained data temporally later than the training data provided. The winning paper by (Yu et al. 2010) utilized an ensemble of linear models such as linear SVM, linear regression, and logistic regression since they directly minimize RMSE, the competition criterion. However, since the competition allowed contestants to train on nearly a full semesters worth of data before prediction, it is not the most realistic indicator of how a practical ITS prediction model would perform during the semester when less training data is available. Even though RMSE has been shown to be a good measure of accuracy, it is scale dependent (Hyndman and Koehler 2006) which poses a problem when using it to evaluate models that will receive incrementally larger sets of training data as with the the typical ITS use case throughout a school semester. Another competition team (Toscher and Jahrer 2010) also used NN based CF methods and managed to achieve third place in the competition, but only after combining it in an ensemble with many other classification methods. This does however, show promise for a NN based CF method being used for ITS prediction of the KDD dataset and this paper will explore it under the non-scale dependent evaluation metric of simple accuracy. As mentioned previously, this paper will also use a new testing paradigm where the model will incrementally receive training data to simulate different points of the semester.

2.2 Exploratory Data Analysis

Before diving into the core problem of creating a good ITS predictive model, an exploration of the data is performed. The dataset is from the 2010 KDD Cup Educational Data Mining Challenge, specifically the 2008-2009 Bridge to Algebra dataset (Stamper et al. 2010). It contains data for **6043** students over a full year semester dated from **2008/09/06 to 2009/06/26**. As the algebra name implies, this dataset was taken from students in a algebra/mathematics course and consequently the ITS used focuses on math problems.

2.2.1 Data Schema

A summary of the dataset schema can be found in (Table 2.2) along with descriptions of each column. Columns that are highlighted are the columns that will be used in the analysis for the rest of this paper. Each row in the dataset represents a step of a problem; a step is the lowest level on the hierarchy of how problems are organized

in the ITS (see Figure 2.1).

2.2.2 Unique Problems and Steps Attempted

Out of all possible problems in the ITS system, the proportion of unique problems each student will complete by the end of semester is surprisingly low. (Figure 2.2) shows that the majority of students only completed $< 1\%$ of all possible problems in the ITS. Since a unique problem is only counted if a student attempted it, this means the ITS is recommending a very niche set of problem to every student. This also means that we can expect the data to be very sparse. A similar trend can be observed when looking at a granular level with steps. (Figure 2.3) also shows that most students have only completed $< 1\%$ of all possible steps in the ITS.

2.2.3 Student Accuracy

Besides knowing how many of the problems students answered in the ITS, it is important to know how often they are correct to gauge the general level of difficulty of ITS problems. (Figure 2.4) shows distribution of average accuracy per student to be mostly left skewed indicating that students tend to get a step correct more often than incorrect, the mode student has an accuracy of $\approx$ 87%.

2.2.4 Knowledge Components Attempted

A distribution of the how many knowledge components were exposed to a certain proportion of all students can be found in (Figure 2.5). As described in (Table 2.2), a knowledge component is simply what subject matter the problem covers. Specifically for this dataset, what math skills the problem requires to solve it. Similar to the amount of unique problems each student has been exposed to, most of the students aren't exposed to the full range of mathematical skills the ITS challenges them on.

2.3 Subset of Data Used

Due to the limited computing resources, a subset of the total dataset from the 2008-2009 Bridge to Algebra dataset (Stamper et al. 2010) is used. To select the subset, the following steps were followed:

1. Find which knowledge component was a part of in the largest proportion of all ITS problems
2. Take only those problems that contained the this knowledge component as the subset of data

This subset contains **4593** unique students and **30043** unique problem-step combinations. All experiments in later chapters will use this subset of data. For details of the relative ranking of knowledge components, see (Table 2.3) for the top 10 knowledge components.

2.4 Notation

Throughout the paper, various constants will be referenced that show the size of the dataset. A summary of these constants can be found in (Table 2.4). Numerical

Column	Data Type	Description
Row	Integer	unique row number
Anon Student Id	String	unique alphanumerical id
Problem Hierarchy	String	the hierarchy of curriculum levels for this given problem (the section and unit)
Problem Name	String	unique string identifier for a problem
Problem View	Integer	Running count of how many times student has encountered this problem
Step Name	String	unique string identifier for the step GIVEN a problem
Step Start Time	TMstamp	starting time of the step
First Transaction Time	TMstamp	time of first transaction towards the step
Correct Transaction Time	TMstamp	time of the correct attempt towards the step, if there was one
Step End Time	TMstamp	time of the last transaction towards this step
Step Duration (sec)	Integer	total time spent on step
Correct Step Duration (sec)	Integer	step duration if the first attempt was correct
Error Step Duration (sec)	Integer	step duration if the first attempt was incorrect
Correct First Attempt	Boolean	true if correct on first attempt of the step, false otherwise. **This is the response variable being predicted**
Corrects	Integer	running count of total correct attempts by the student on this step
Incorrects	Integer	running count of total incorrect attempts by the student on this step
Hints	Integer	running count of total hints requested by the student for this step
KC	String	identified skills used in this problem
Oppotunity	Integer	running count of encounters with the given KC

Table 2.2: 2008-2009 Bridge to Algebra Data Schema

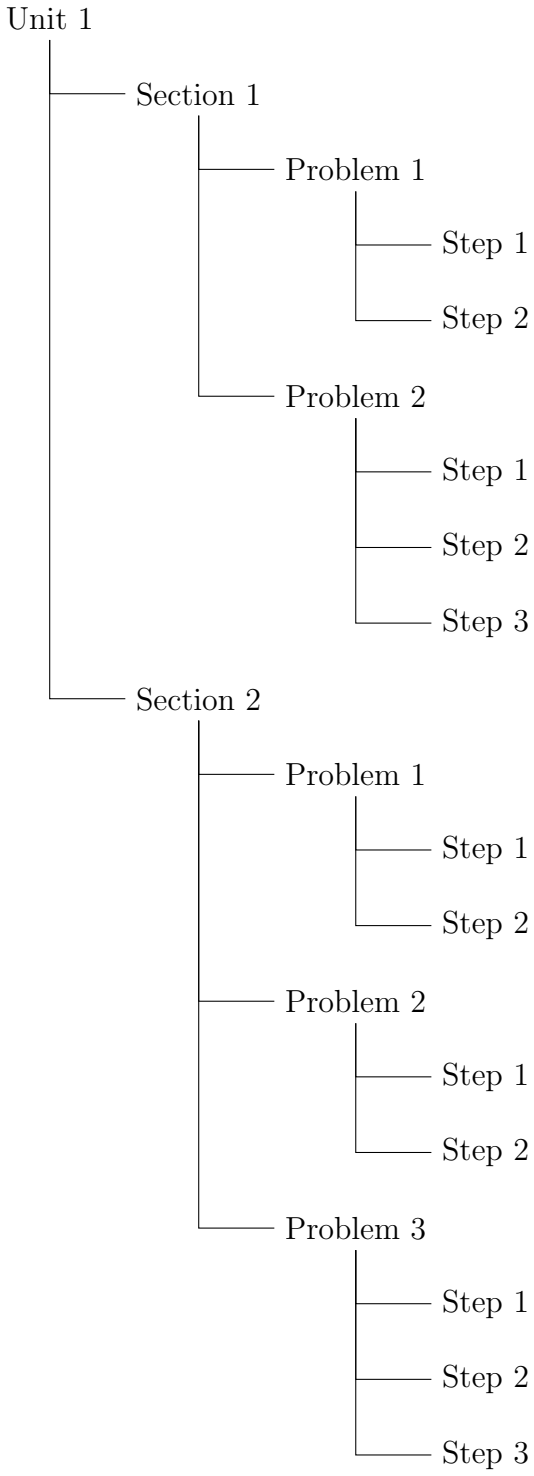

Figure 2.1: Example hierarchy of how problems are organized in the 2010 KDD ITS

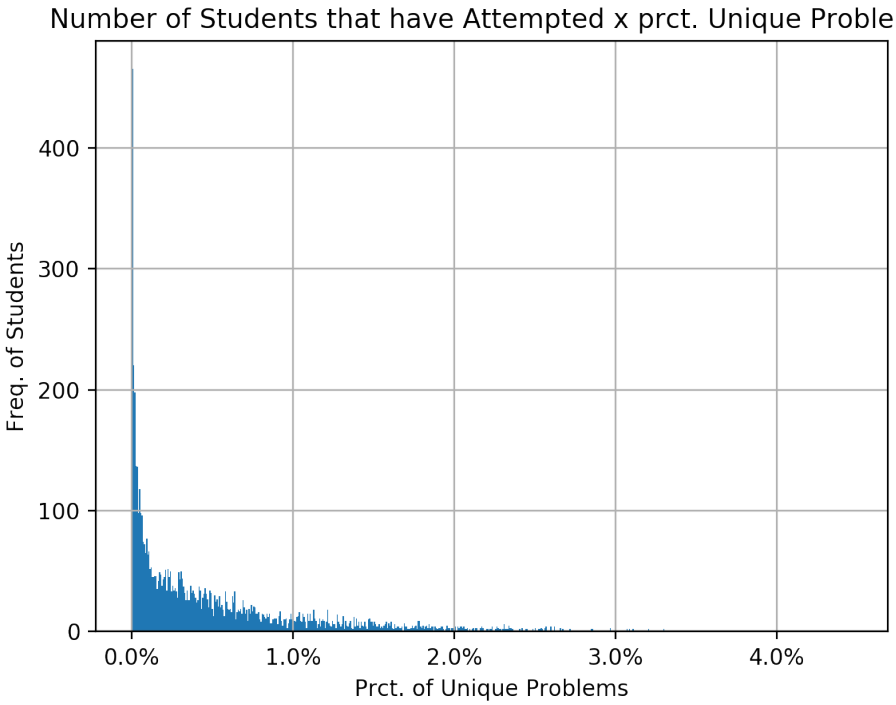

Figure 2.2: Freq. of students that attempted a % of all unique problems in the ITS

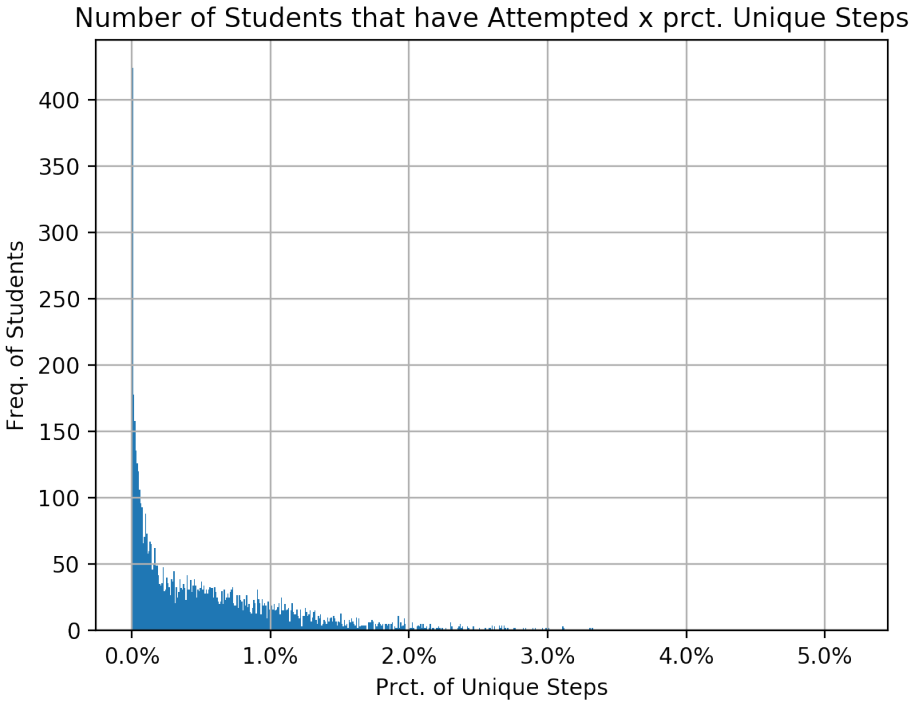

Figure 2.3: Freq. of students that attempted a % of all unique steps in the ITS

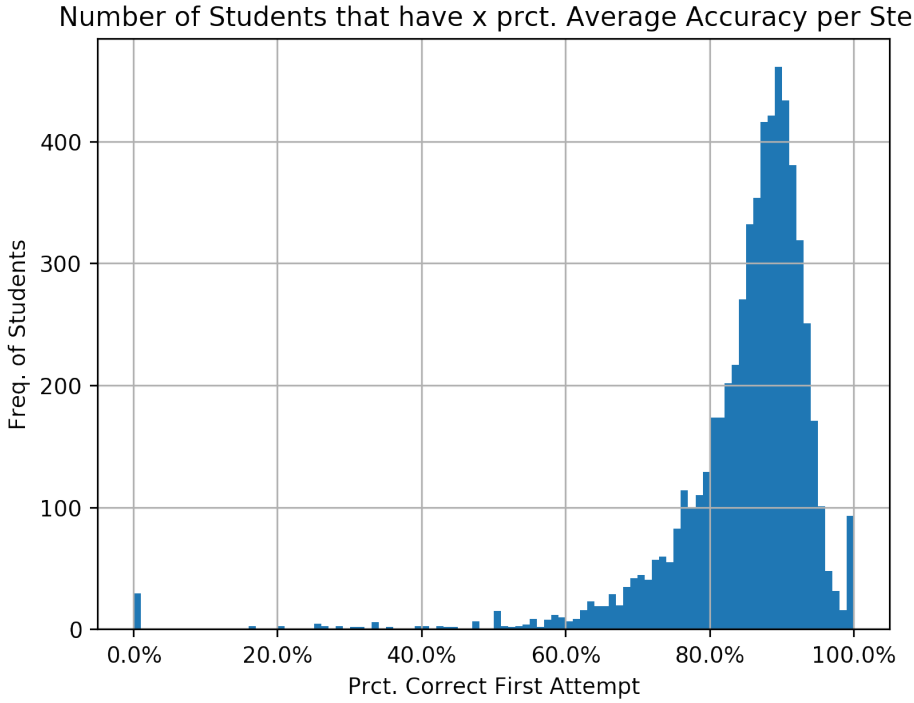

Figure 2.4: Freq. of students who have a certain % average accuracy on all steps they've attempted

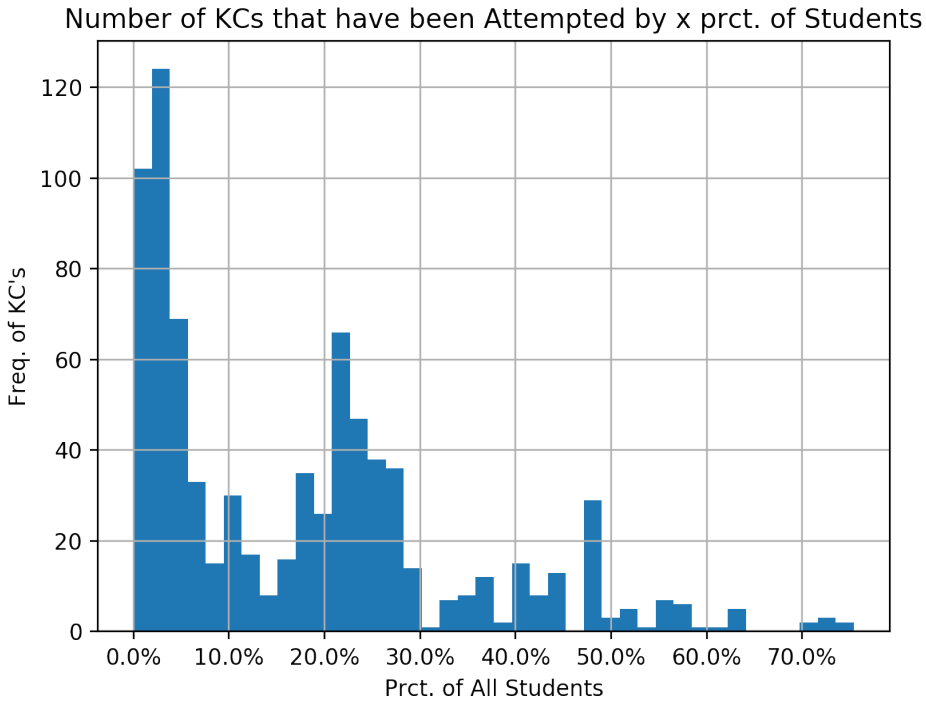

Figure 2.5: Freq. of students that attempted a % of all unique knowledge components

Rank	Knowledge Component	Proportion of Problems
1	Calculate sum digit – no carry-1	0.753930
2	Identify operator-1	0.750952
3	Calculate difference digit from 2 digits-1	0.729274
4	Calculate difference digit – no borrow-1	0.728777
5	Calculate sum digit from 2 digits-1	0.726957
6	Calculate difference digit from 1 digit-1	0.712560
7	Calculate sum digit from 1 digit-1	0.710905
8	Calculate sum digit – carry out-1	0.640079
9	Calculate sum digit – carry in-1	0.640079
10	Calculate difference digit – borrow in-1	0.634122

Table 2.3: Top 10 knowledge components that were contained in the highest proportions of problems in the full dataset

Constant	Description
M	number of unique students
N	number of unique problem-step combinations
T	number of discrete time intervals

Table 2.4: Table of constants

features are often represented as a 3D tensor. The template of 3D tensors in this thesis are shown by equation 2.1. Each axis of the 3D tensor represents either a unique student, problem-step, or discrete time interval. The values of the tensors are a given feature's numerical value, which feature is indicated by the subscript.

$$\mathbf{X}_{feature\_name} \in R^{M \times N \times T} \tag{2.1}$$

M =number of unique students, N =number of unique problem-steps, and T =number of discrete time interval. The following subsections provide concrete definitions of all 3D tensors to be used in the various methods in this paper.

2.4.1 X_correct and X_views

Once timestamps have been discretized, at any given time a student will have answered a given problem-step and therefore viewed that question. X_correct is a boolean indicating whether they got that problem-step correct or not, while X_views is the running count of views of that specific problem-step up to time t. These definitions are formalized in Equations 2.2, 2.3, and 2.4.

$$\mathbf{X}_{correct} \in R^{M \times N \times T}, \mathbf{X}_{views} \in R^{M \times N \times T} \tag{2.2}$$

$$X_{correct}^{(i,\ j,\ t)} = \begin{cases} 0, & \text{if student } i \text{ [is wrong about OR did not do]} \\ & \text{problem-step } j \text{ at time } t \\ 1, & \text{if student } i \text{ got problem-step } j \text{ correct at time } t \end{cases} \tag{2.3}$$

$$X_{views}^{(i,\ j,\ t)} = \begin{cases} \text{times student } i \text{ has seen} & \text{if this problem-step was} \\ \text{problem-step } j \text{ at time } t, & \text{attempted at time } t \\ 0, & \text{else} \end{cases} \tag{2.4}$$

$$\forall i = 1, \ldots, M \ \forall j = 1, \ldots, N \ \forall t = 1, \ldots, T$$

2.4.2 X_correct_latest and X_views_latest

Because students can try problem-step combinations more than once, their latest "correctness" on a given problem-step may change. $X_{correct\_latest}$ stores the most recent "correctness" at time t for all problem-steps they've attemptd up to time t. Similairly, whereas X_{views} only contained the views for a given problem-step exectued at time t, we are also interested in the total views a student had on all problem-steps they've attempted up to time t. The mathematical definitions are formally stated in Equations 2.5, 2.6, and 2.7.

$$\mathbf{X}_{correct\_latest} \in R^{MxNxT}, \mathbf{X}_{views\_latest} \in R^{MxNxT} \tag{2.5}$$

$$X_{correct\_latest}^{(i,\ j,\ t)} = \begin{cases} 0, & \text{if student } i \text{ [was wrong on their last attempt of OR} \\ & \text{did not do] problem-step } j \text{ up to time } t \\ 1, & \text{if student } i \text{ got problem-step } j \text{ correct on their last} \\ & \text{attempt up to time } t \end{cases} \tag{2.6}$$

$$X_{views\_latest}^{(i,\ j,\ t)} = \text{number of times student } i \text{ has attempted problem-step } j \text{ up to time } t \tag{2.7}$$

$$\forall i = 1, \ldots, M \ \forall j = 1, \ldots, N \ \forall t = 1, \ldots, T$$

2.4.3 X_correct_cnt_latest and X_incorrect_cnt_latest

For a given student i, what is their running count of corrects/incorrects for a given problem-step j at time t. Corrects are converted into binary while incorrects are left as counts. The mathematical definitions are formally stated in Equations 2.8, 2.9, and 2.10.

$$\mathbf{X}_{correct\_cnt\_latest} \in R^{MxNxT}, \mathbf{X}_{incorrect\_cnt\_latest} \in R^{MxNxT} \tag{2.8}$$

$$X_{correct\_cnt\_latest}^{(i,\ j,\ t)} = \begin{cases} 0, & \text{if student } i \text{ has never been correct on problem-step } j \text{ at time } t \\ 1, & \text{if student } i \text{ has been correct at one point on problem-step } j \text{ at time } t \end{cases} \tag{2.9}$$

$$X_{incorrect\_cnt\_latest}^{(i,\ j,\ t)} = \text{number of times student } i \text{ has been incorrect on problem-step } j \text{ at time } t \tag{2.10}$$

$$\forall i = 1, \ldots, M \ \forall j = 1, \ldots, N \ \forall t = 1, \ldots, T$$

Chapter 3

Baseline Methods

To evaluate the NN-based CF methods in Chapter 4, some baseline methods are established for comparison. The following methods in this chapter will be used as baselines for the empirical results.

3.1 Global Average

The most naive baseline method, the Global Average method just predicts all 0 or 1 based on whether the average of all non-sparse elements (non-absent answers for problem-steps) in $X_{correct\_latest}$ is ≥ 0.5, more formally given by Equation 3.1.

$$\mathcal{A} = \text{set of } (i,\ j,\ t) \text{ indices where student } i \text{ has attempted problem-step } j \text{ at or before time } t$$

$$X_{pred}^{(i,\ j,\ t)} = \begin{cases} 1, & \text{if } \frac{\sum_{i',\ j' \in \mathcal{A}^{(t)}} X_{correct\_latest}^{(i',\ j',\ t)}}{|\mathcal{A}^{(t)}|} \geq 0.5 \\ 0, & \text{else} \end{cases} \tag{3.1}$$

3.2 Within Student Average

A slightly more sophisticated way of doing the average is to predict the average across all students on a given problem-step. This happens to correspond to taking the average of a column in our 2D sparse matrices for a given time t, formally this is given by Equation 3.2.

$$X_{pred}^{(i,\ j,\ t)} = \begin{cases} 1, & \text{if } \frac{\sum_{i' \in \mathcal{A}^{(j,\ t)}} X_{correct\_latest}^{(i',\ j,\ t)}}{|\mathcal{A}^{(j,\ t)}|} \geq 0.5 \\ 0, & \text{else} \end{cases} \tag{3.2}$$

3.3 Within Problem-Step Average

Similar to the Within Col except now for rows, predict using the historical average of the student. This happens to correspond to taking the average of a row in our 2D sparse matrices for a given time t, formally this is given by Equation 3.3.

$$X_{pred}^{(i,\ j,\ t)} = \begin{cases} 1, & \text{if } \frac{\sum_{j' \in \mathcal{A}^{(i,\ t)}} X_{correct\_latest}^{(i,\ j',\ t)}}{|\mathcal{A}^{(i,\ t)}|} \geq 0.5 \\ 0, & \text{else} \end{cases} \tag{3.3}$$

	Within Student Avg.	**Within Prob-Step Avg.**
1.)	$\mathbf{X}^{(t)}_{correct\_latest} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & ? \end{bmatrix}$	$\mathbf{X}^{(t)}_{correct\_latest} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & ? \end{bmatrix}$
2.)	Within Stud Avg. $= \frac{1+1+0}{3} \approx 0.66$	Within P-S Aveg. $= \frac{0+0}{2} = 0$
3.)	Since $0.66 \geq 0.5$ predict 1	Since $0 < 0.5$ predict 0
4.)	$\mathbf{X}^{(t)}_{pred} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}$	$\mathbf{X}^{(t)}_{pred} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$

Table 3.1: Example of predicting the missing value (indicated with a ?) of the 4th Problem-Step (column) for the 3rd student (row) at time t using the Within Student Average (left) and Within Problem-Step Average (right)

An example of the Within Student Average and Within Problem-Step is shown in (Table 3.1). This is an example assuming prediction for the discrete time t and for an ITS with only 3 students and 4 unique problem-steps.

Chapter 4

Nearest Neighbors Methods

This chapter describes all the NN-based CF methods that were experimented with. Each one is briefly described and formalized with equations. All final prediction output values are compared against a threshold to determine whether the problem-step should be classified as correct (1) or incorrect (2) and will be tuned for performance.

4.1 Cosine Similarity NN with {0, 1} encoding

The first NN method attempted involved only utilizing the latest corrects and incorrects of a student for any given problem-step. Corrects are encoded as 1 while incorrects and missing answers are encoded as 0. The cosine similarity tensor is defined as $\mathbf{S}$ (Equation 4.1), the tensor is pairwise between students (rows) for a given time slice and is based on the default $0, 1$ encoding defined by $\mathbf{X}_{correct\_latest}$ earlier (Equation 2.6).

$$\mathbf{S} \in R^{M \times M \times T}$$

$$S^{(i_1,\ i_2,\ t)} = \frac{X^{(i_1,\ \cdot,\ t)}_{correct\_latest} \cdot X^{(i_2,\ \cdot,\ t)}_{correct\_latest}}{\|X^{(i_1,\ \cdot,\ t)}_{correct\_latest}\|\ \|X^{(i_2,\ \cdot,\ t)}_{correct\_latest}\|} \tag{4.1}$$

$$\forall i_1 = 1, \ldots, M\ \forall i_2 = 1, \ldots, M\ \forall t = 1, \ldots, T$$

Next, the base prediction matrix is obtained and contains the sum products of the i^{th} students similarity row from $\mathbf{S}$ and the corresponding j^{th} problem-step column in $\mathbf{X}_{correct\_latest}$ This is then element wise divided by the sum of similarity scores for a given student i and problem-step j, but only if those have students attempted the problem-step before or on current time t. The resulting prediction matrix is formalized in Equations 4.2, 4.3, 4.4, and 4.5.

$$\mathbf{X}^{(t)}_{base\_pred} = \mathbf{S}^{(t)} \cdot \mathbf{X}^{(t)}_{correct\_latest} \tag{4.2}$$

$$X^{(i,\ j,\ t)}_{attempt\_latest} = \begin{cases} 1, & \text{if student } i \text{ has attempted problem-step } j \text{ before or at time } t \\ 0, & \text{else} \end{cases} \tag{4.3}$$

$$\forall i = 1, \ldots, M\ \forall j = 1, \ldots, N\ \forall t = 1, \ldots, T$$

$$\mathbf{X}^{(t)}_{weight\_divisor} = \mathbf{S}^{(t)} \cdot \mathbf{X}^{(t)}_{attempt\_latest} \tag{4.4}$$

$$\mathbf{X}^{(t+1)}_{pred} = \mathbf{X}^{(t)}_{base\_pred} \circ \frac{1}{\mathbf{X}^{(t)}_{weight\_divisor}} \tag{4.5}$$

Since the point of a CF framework is to leverage the results of others, a students similarity to themselves must be set to 0 so that they are not considered from prediction. As a tweak to the formulation, the similarity matrix $\mathbf{S}$ will have 0 down the diagonal in order to prevent the students past performance on a problem-step contribute to their prediction. The final prediction matrix at time $t+1$ is shown in Equation 4.9.

$$\mathbf{S}^{(i_1,\ i_2,\ t)}_{noself} = \begin{cases} 0, & \text{if } i_1 = i_2 \\ \mathbf{S}^{(i_1,\ i_2,\ t)}, & \text{else} \end{cases} \tag{4.6}$$

$$\mathbf{X}^{(t)}_{weight\_divisor\_noself} = \mathbf{S}^{(t)}_{noself} \cdot \mathbf{X}^{(t)}_{attempt\_latest} \tag{4.7}$$

$$\mathbf{X}^{(t+1)}_{pred} = \mathbf{X}^{(t)}_{base\_pred} \circ \frac{1}{\mathbf{X}^{(t)}_{weight\_divisor\_noself}} \tag{4.8}$$

$$X^{(i,\ j,\ t+1)}_{pred\_final} = \begin{cases} 1, & \text{if } X^{(i,\ j,\ t+1)}_{pred} \geq \text{threshold} \\ 0, & \text{else} \end{cases} \tag{4.9}$$

$$\forall i = 1, \ldots, M\ \forall j = 1, \ldots, N\ \forall t = 1, \ldots, T$$

Methods in Sections 4.2 and 4.3 use the formulation outlined in this section but with slight tweaks. To exemplify how this formulation works, consider an ITS database with 4 students and 5 unique problem-steps in (4.10). Lets say the ITS needs to predict for the 1st student, 5th problem-step (highlighted in red).

$$\mathbf{X}^{(t)}_{correct\_latest} = \begin{bmatrix} 1 & 1 & 1 & 0 & ? \\ 0 & 1 & ? & 0 & 1 \\ ? & ? & 1 & ? & 1 \\ 1 & 1 & 0 & ? & 0 \end{bmatrix} \tag{4.10}$$

The first step is to calculate student 1's cosine similarity with all other students. All missing values become encoded as 0's and each row of (4.10) represents a different student. The similarity calculation between student 1 and student 2 is shown in Equations 4.11 and 4.12.

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} \Rightarrow X^{(stud1,t)}_{correct\_latest} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad X^{(stud2,t)}_{correct\_latest} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\frac{X^{(stud1,t)}_{correct\_latest} \cdot X^{(stud2,t)}_{correct\_latest}}{\|X^{(stud1,t)}_{correct\_latest}\| \|X^{(stud2,t)}_{correct\_latest}\|} = \frac{(1 \cdot 0) + (1 \cdot 1) + (1 \cdot 0) + (0 \cdot 0) + (0 \cdot 1)}{\sqrt{3} * \sqrt{2}} \tag{4.11}$$

$$= \frac{1}{\sqrt{6}} \approx \mathbf{0.41} \tag{4.12}$$

The same steps are followed for similarities between student 1 v.s. (3 and 4). The resulting similarities are shown in Equation 4.13, this is equivalent to the top row of the similarity matrix $\mathbf{S}^{(t)}$.

$$\mathbf{S}^{(stud1,\ \cdot,\ t)} = \begin{bmatrix} 0 & 0.41 & 0.41 & 0.82 \end{bmatrix} \tag{4.13}$$

Since all students other than student 1 have answered problem-step 5 before, the predicted value (before considering threshold) is the weighted average of their answers using similarity as a weighting (Equations 4.14 and 4.15).

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} \Rightarrow \mathbf{X}^{(\cdot,\ probstep5,\ t)}_{correct\_latest} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{X}^{stud1,\ probstep5,\ (t+1)}_{pred} = \mathbf{S}^{(stud1,\ \cdot,\ t)} \cdot \mathbf{X}^{(\cdot,\ probstep5,\ t)}_{correct\_latest} \tag{4.14}$$

$$= \begin{bmatrix} 0 & 0.41 & 0.41 & 0.82 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \frac{}{0 + 0.41 + 0.41 + 0.82} = \mathbf{0.5} \tag{4.15}$$

If $0.5 \geq$ threshold then we predict that student 1 will get problem-step 5 correct (1), else we predict they will get it incorrect (0). The final result is therefore described in Equation 4.16.

$$X^{(stud1,\ probstep5,\ t)}_{pred\_final} = \begin{cases} 1(\text{correct}), & \text{if } 0.5 \geq \text{threshold} \\ 0(\text{incorrect}), & \text{else} \end{cases} \tag{4.16}$$

4.2 Cosine Similarity NN with {-1, 0, 1} encoding

Instead of encoding absent and incorrect problem-steps as the same value, they are now distinguished by encoding incorrects as -1, penalizing the weighted average accordingly and providing more "resistance" to correct responses by similar students in the final prediction calculation. The formulation is thus the same as Section 4.1 but now all incorrects in $\mathbf{X}_{correct\_latest}$ turn from 0 to -1 (Equation 4.17).

$$X^{(i,\ j,\ t)}_{correct\_latest} = \begin{cases} -1 & \text{if student } i \text{ was wrong on their last attempt of problem-step } j \text{ up to time } t \\ 0 & \text{if student } i \text{ did not do problem-step } j \text{ up to time } t \\ 1 & \text{if student } i \text{ got problem-step } j \text{ correct on their last attempt up to time } t \end{cases} \tag{4.17}$$

$$\forall i = 1, \ldots, M\ \forall j = 1, \ldots, N\ \forall t = 1, \ldots, T$$

After this modification of $\mathbf{X}_{correct\_latest}$, the formulation is identical to section 4.1.

4.3 L1 Similarity NN with 0, 1 encoding

The formulations is the same as 4.1 but the **S** matrix will now be measured using $L1$ similarity instead (Equation 4.21). Prediction happens in the same way after the similarity matrix is calculated.

$$\mathbf{S} \in R^{MxMxT} \tag{4.18}$$

$$\begin{aligned} S^{(i_1,\ i_2,\ t)} =& (\max\left(X^{(t)}_{correct\_latest}\right) - \min\left(X^{(t)}_{correct\_latest}\right)) \\ & - \frac{\|X^{(i_1,\ \cdot,\ t)}_{correct\_latest} - X^{(i_2,\ \cdot,\ t)}_{correct\_latest}\|_1}{N} \end{aligned} \tag{4.19}$$

$$S^{(i_1,\ i_2,\ t)} = (1 - 0) - \frac{\|X^{(i_1,\ \cdot,\ t)}_{correct\_latest} - X^{(i_2,\ \cdot,\ t)}_{correct\_latest}\|_1}{N} \tag{4.20}$$

$$S^{(i_1,\ i_2,\ t)} = 1 - \frac{\|X^{(i_1,\ \cdot,\ t)}_{correct\_latest} - X^{(i_2,\ \cdot,\ t)}_{correct\_latest}\|_1}{N} \tag{4.21}$$

4.4 L1 Similarity NN with -1, 0, 1 encoding

This method is the same formulation as Section 4.3 but now $X_{correct\_latest}$ is defined to include incorrects as -1 (such as Equation 4.17 in Section 4.2).

4.5 Running Total Similarity

This method defines a distance, D, between users based on the current running count of corrects and incorrects defined in Section 2.4. Formally, a vector of size N (number of unique problem-steps) is created between each student pair, and then aggregated (the aggregation function can change) to create a distance as stated in Equation 4.23.

$$\begin{aligned} d^{(i_1,\ i_2,\ t,\ j)} =& w_{cor} \cdot |X^{(i_1,\ j,\ t)}_{correct\_cnt\_latest} - X^{(i_2,\ j,\ t)}_{correct\_cnt\_latest}| \\ & + w_{incor} \cdot |X^{(i_1,\ j,\ t)}_{incorrect\_cnt\_latest} - X^{(i_2,\ j,\ t)}_{incorrect\_cnt\_latest}| \end{aligned} \tag{4.22}$$

$$D^{(i_1,\ i_2,\ t)} = agg\_func(\mathbf{d}^{(i_1,\ i_2,\ t,\ \cdot)}) \tag{4.23}$$

Aggregation functions that will be used in this thesis are sum, mean, and l2. A similarity can then be derived by Equation .

$$S^{(i_1,\ i_2,\ t)} = 1 - \frac{D^{(i_1,\ i_2,\ t)}}{\max(\mathbf{D})} \tag{4.24}$$

Since any element of **D** is positive, $\max(\mathbf{D})$ is the same as the range of **D** and thus the similairity calculation is normalized to a range of $\in [0, 1]$. Once the pair-wise similarity matrix is defined by 4.24 the procedure to predict is the same as Section 4.1.

Chapter 5

Empirical Results

In this chapter the methods discussed from Chapters 3 and 4 will be tested and compared. A new testing paradigm will be shown and the details of model tuning will be outlined as well. Lastly, some examples from using the Running Total Similarity method are discussed in detail.

5.1 Methodology

First, the testing paradigm on which the methods are being evaluated in this thesis will be shown. Secondly, how each model's hyperparameters (thresholds) were tuned are also discussed.

5.1.1 Testing Paradigm

Each row of data in the ITS dataset represents the answer to a specific problem-step by a certain student. The parent problem of each of these steps have been ordered chronologically for each student such that 1 represents the first problem the student attempted, 2 represented the second problem the student attempted, and so on. Each student therefore has a finite set of problems that are chronologically ordered (example in Table 5.1), this rank will be the time dimension (**tm**) in the testing paradigm. This also means that all steps in a given unique student-problem combination contains the same tm values (e.g. all steps from problem P1 done by student A will have a tm value of x).

Once all the data has been labelled with a discrete tm value, it becomes possible to test the ITS as if it were at different points of the semester. Assuming that

Student	tm	Problem
A	1	P1
A	2	P4
A	3	P2
B	1	P3
B	2	P2
C	1	P4

Table 5.1: An example of 3 students and how tm labels are assigned to problems they have completed: student A completed problems in the order P1, P4, and P2 as indicated by the tm column values of 1, 2, and 3 respectively

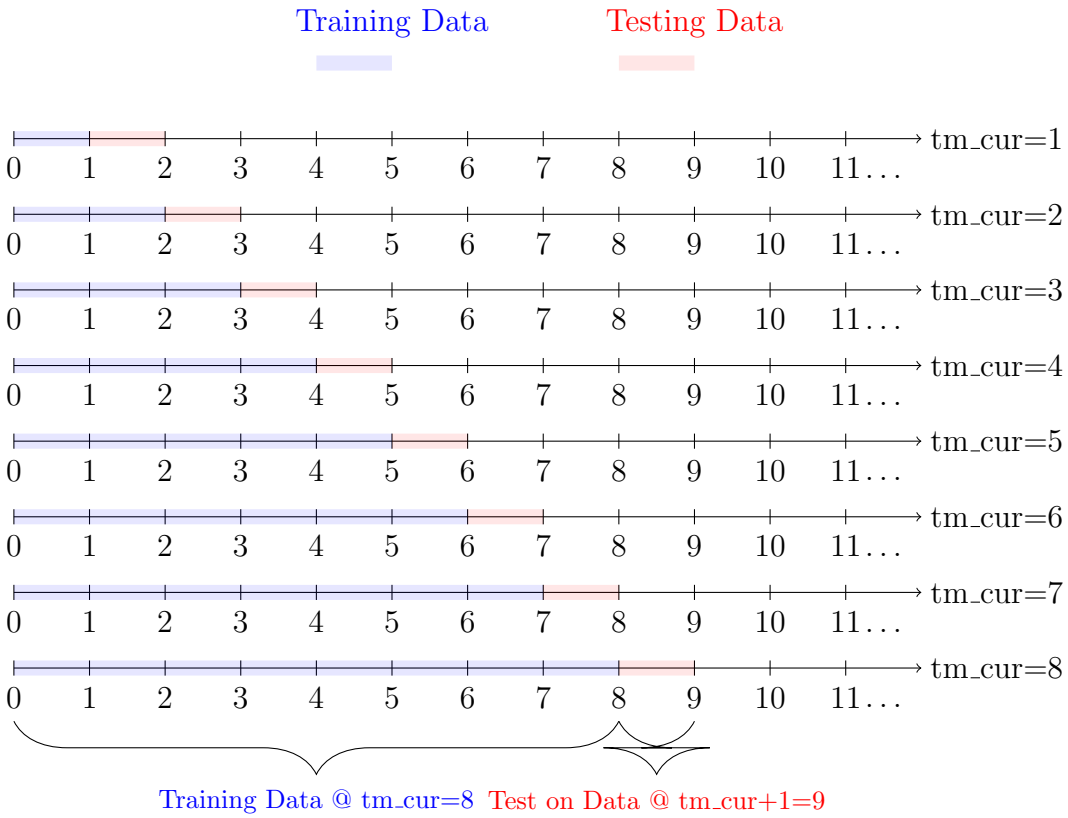

Figure 5.1: Overview: Testing Paradigm

the current time is tm_cur, the testing paradigm incrementally crawls later and later training the models up to the current time (tm_cur) while testing it on the immediate interval after ($tm\_cur + 1$). This way of testing mimics the real life scenario of predicting the correctness of the next problem a student will answer based on the data you have at any given point of the semester. The results are plotted as the accuracy on the predicted answers at $tm\_cur + 1$ (test set) based on training data from $training\_tm \in [1, tm\_cur]$ with the horizontal axis being tm_cur (Figure 5.1).

5.1.2 Hyperparameter Tuning

Different thresholds were attempted for each of the methods using the testing paradigm defined in Chapter 4. The results of the tuning are in figures 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, and 5.8. The chosen hyperparameters were the ones that performed the best in terms of accuracy (dominated over a longer period of time) and are summarized in Table 5.2. Testing was limited to around 5 different values per hyperparameter due to the processing limitations available.

**It should be noted that the hyper parameter results for RTM (Figures 5.6, 5.7, and 5.8) appear to be buggy in implementation as a change in wincor appears to have no effect on the performance. This was unfortunately not fully investigated and a reason was never found for why wincor's impact was effectively non-existent. This presents a good opportunity in the future to more carefully implement RTM based on the mathematical formulations in section 4.5 to reduce the possibility of an erroneous implementation of RTM and have a fair comparison with other methods.*

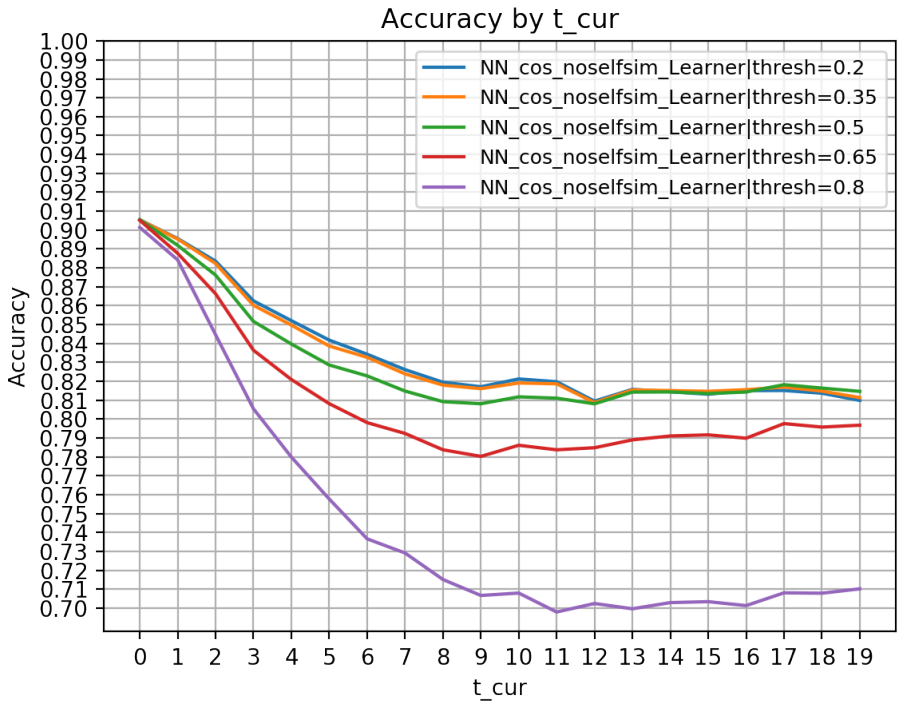

Figure 5.2: Hyperparam Tuning: Cosine Sim with {0, 1} encoding

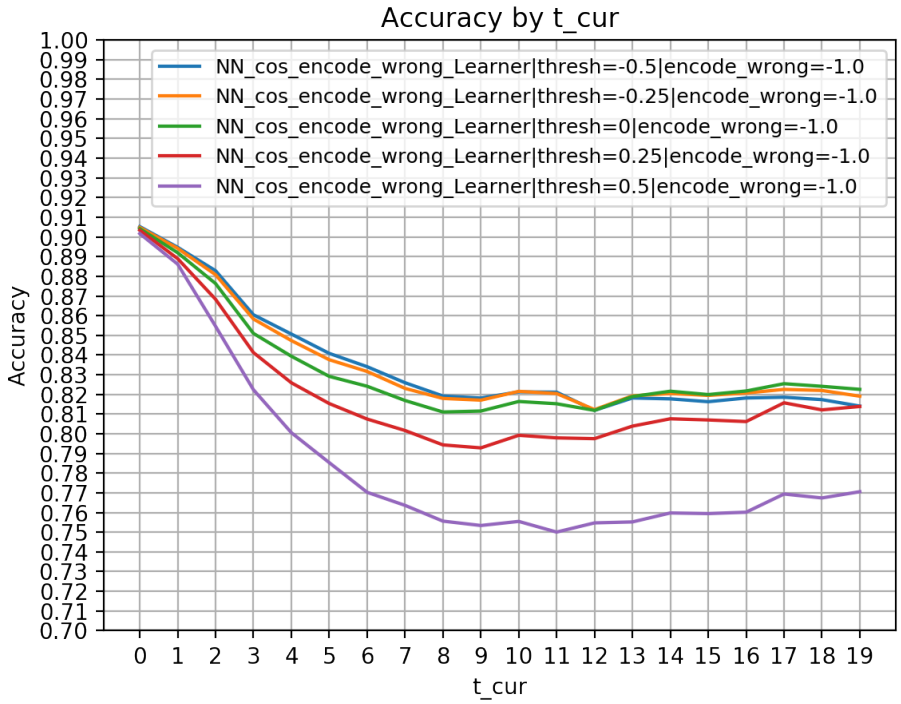

Figure 5.3: Hyperparam Tuning: Cosine Sim with {-1, 0, 1} encoding

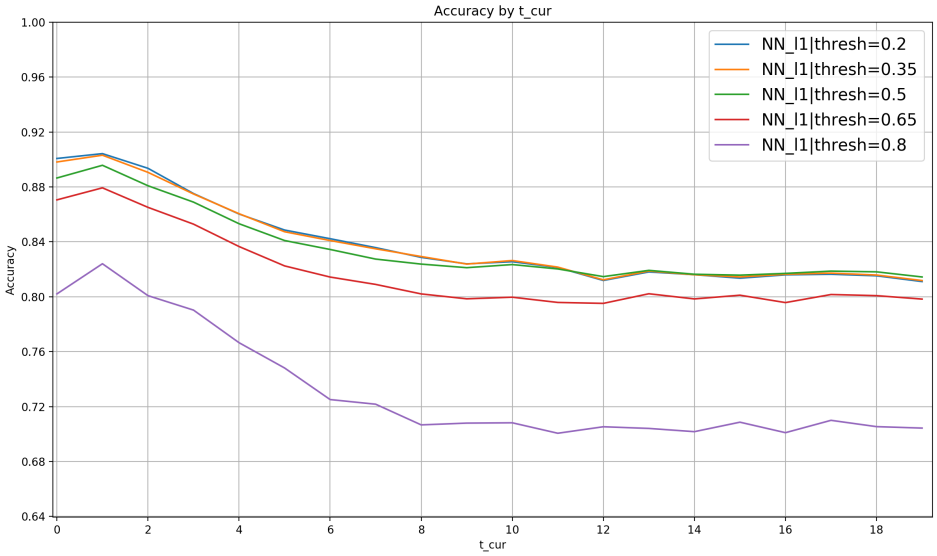

Figure 5.4: Hyperparam Tuning: L1 Sim with {0, 1} encoding

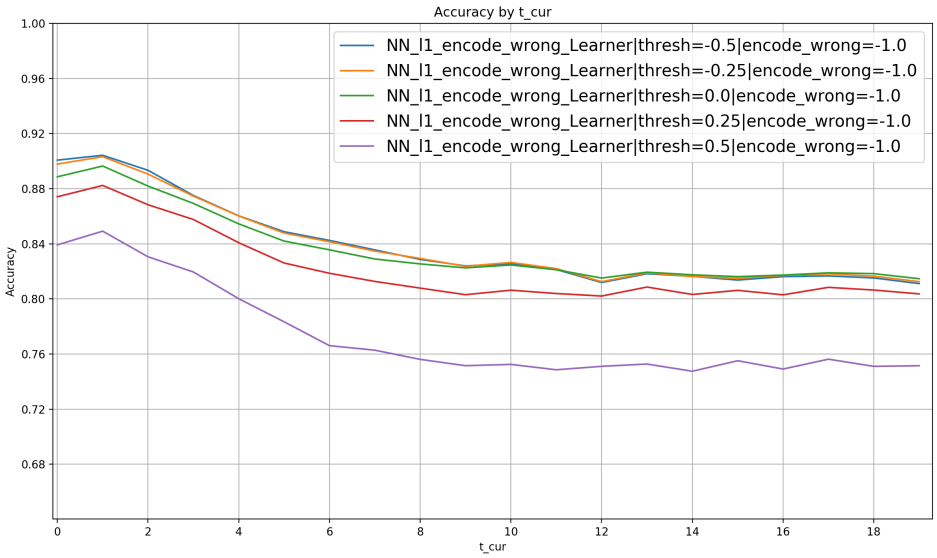

Figure 5.5: Hyperparam Tuning: L1 Sim with {-1, 0, 1} encoding

Method	Best Threshold	Additional Hyperparams
Cosine 0, 1	+0.20	none
Cosine -1, 0, 1	−0.25	none
L1 0, 1	+0.35	none
L1 -1, 0, 1	−0.25	none
Running Total	+0.50	wincor=10, wcor=1, agg=l2

Table 5.2: Chosen hyperparmeters for each method after tuning and finding the ones that give the best accuracy

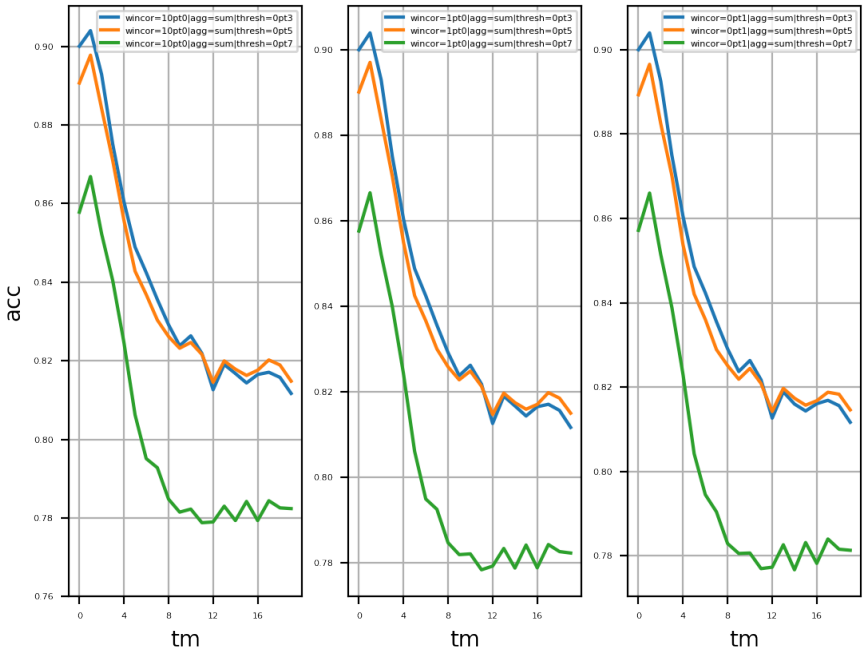

Figure 5.6: Hyperparam Tuning: Running Total Method Agg_Func=sum (from left to right, wincor=10, 1, 0.1)

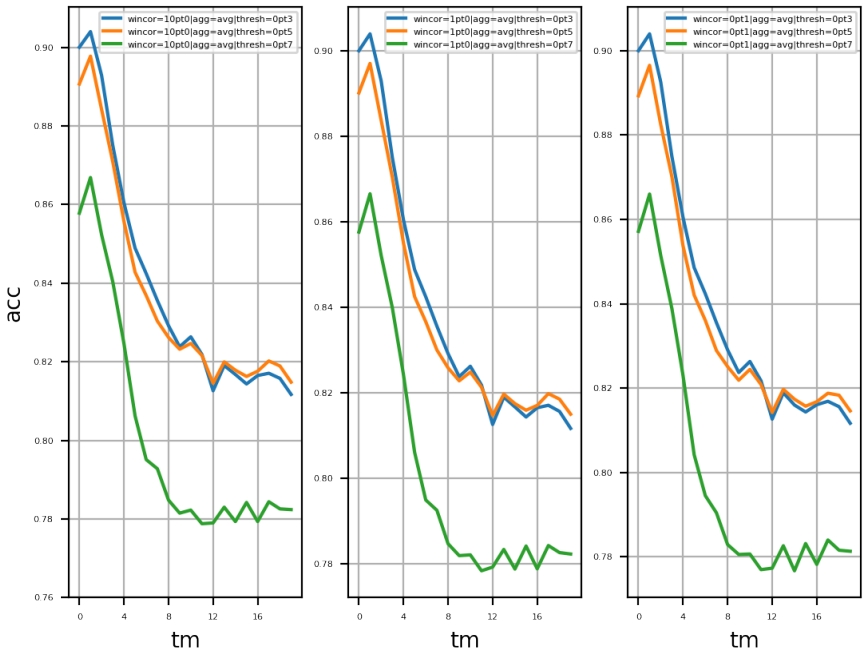

Figure 5.7: Hyperparam Tuning: Running Total Method Agg_Func=avg (from left to right, wincor=10, 1, 0.1)

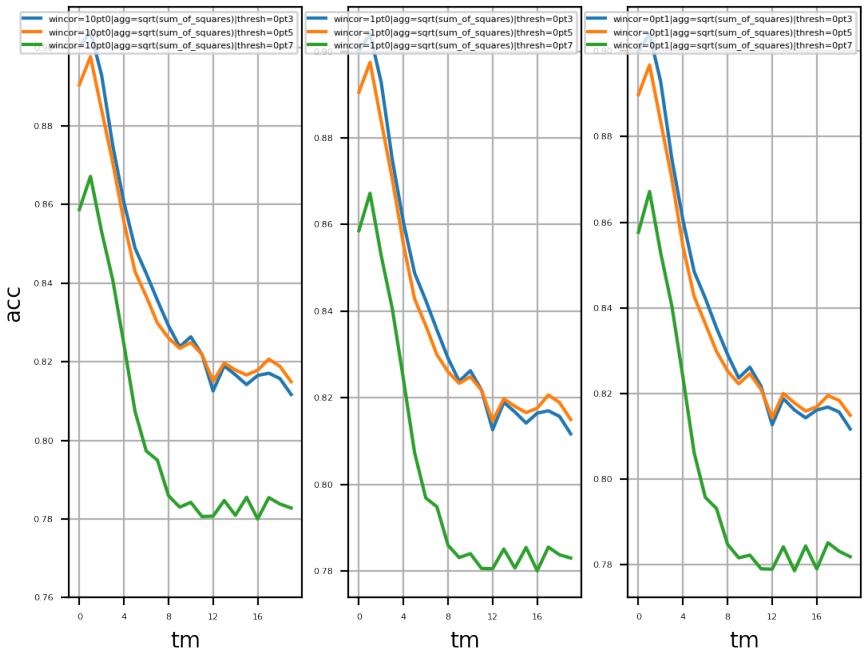

Figure 5.8: Hyperparam Tuning: Running Total Method Agg_Func=l2 (from left to right, wincor=10, 1, 0.1)

5.2 Performance

5.2.1 Comparison Between Methods

The first round of comparisons exclude the Running Total Similarity method as the method was a result of brainstorming after conducting experiments on all other methods. Methods were pitted against the baselines training and testing on the exact same data sets following the training paradigm defined earlier in Section 5.1.1. The Accuracy results are summarized by Figure 5.9.

In addition to the main evaluation metric of basic accuracy, Positive/Negative Predictive Values and True Positive/Negative Rates were also calculated. A summary of the components that make up these ratios are defined in Table 5.3. The ratios themselves are defined by Equations 5.1, 5.2, 5.3, and 5.4. The results for these ratios for all methods can be found in Figures 5.10, 5.11, 5.12, and 5.13. It should be noted that the label "NN_cos_Learner..." applies to the Cosine Similarity NN with {0, 1} encoding method before the self similarities were removed (e.g. Student A still had a similarity of 1 with itself,). This error was discovered later and corrected for subsequent methods (hence why there is another similarly labeled method but with an additional "noselfsim" label) but the results of the test remain on the figures as additional information.

$$\text{Positive Predictive Value} = \frac{TP}{TP + FP} \tag{5.1}$$

$$\text{Negative Predictive Value} = \frac{TN}{TN + FN} \tag{5.2}$$

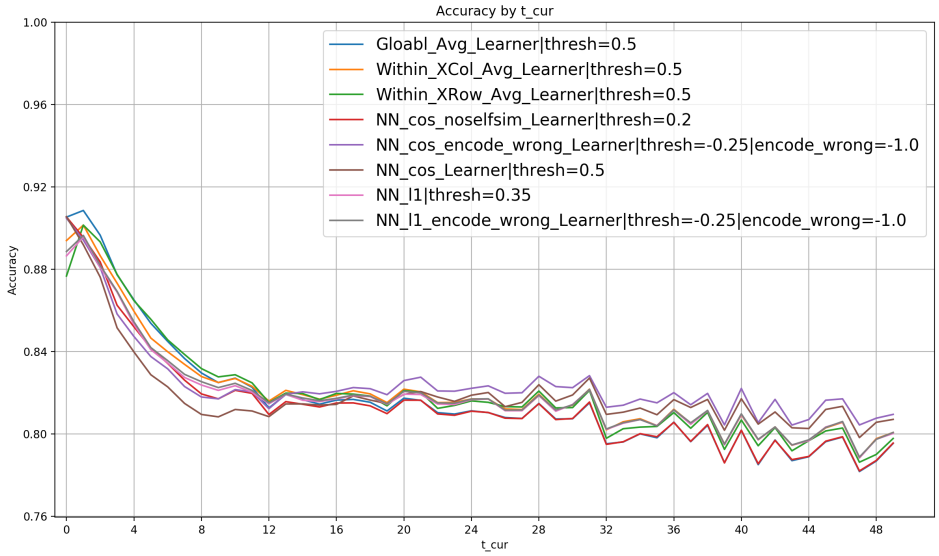

Figure 5.9: Accuracy results following testing paradigm in Section 5.1.1

Actual Value	Predict Correct	Predict (Incorrect)
Correct	True Positive (TP)	False Negative (FN)
Incorrect	False Positive (FP)	True Negative (TN)

Table 5.3: A summary of what True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) are in the context of ITS prediction

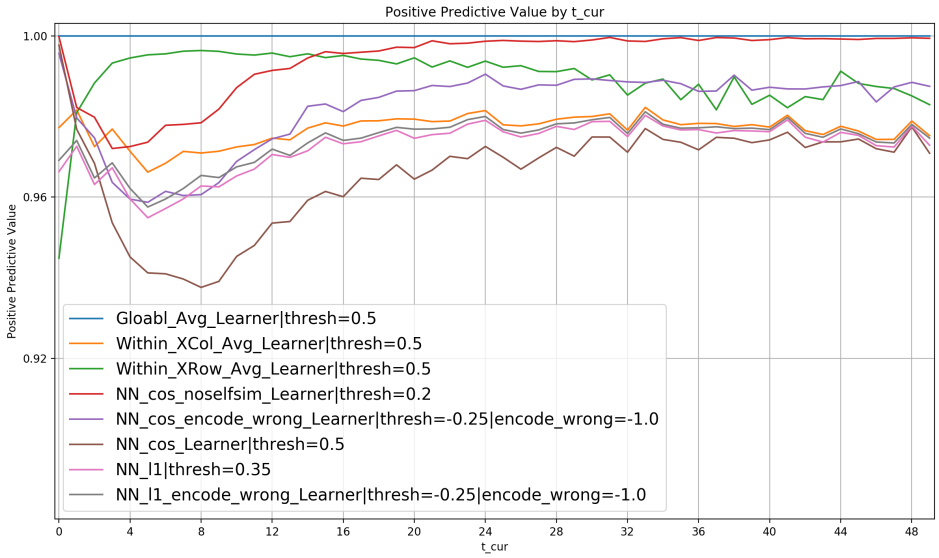

Figure 5.10: Positive Predictive Value results following testing paradigm in Section 5.1.1

$$\text{True Positive Rate} = \frac{TP}{TP + FN} \tag{5.3}$$

$$\text{True Negative Rate} = \frac{TN}{TN + FP} \tag{5.4}$$

The results suggested that the Cosine Similarity NN with -1, 0, 1 (cos_101 from now one) performed the best, as after $t\_cur = 14$ it **consistently beats all other methods, baseline and NN alike on the main evaluation metric accuracy by $\approx 2\%$ in the latter half of t_cur's.** Besides raw model performance comparisons, some other observations about the experiments were:

- Regardless of method, the prediction accuracy was very high during the early tm_cur values suggesting that ITS problems are very simple and elementary at the beginning for most students (this falls in line with the reasonable assumption that courses start off the semester with easier and more basic questions)

- Since students are more often right than wrong (during the exploratory data analysis it was found that students are correct $\approx 87\%$ of the time on average) the methods that performed better tended to have a better ability to find those rare cases where they may be incorrect and thus had higher True Negative Rates (being able to recall incorrect answers) than other worse performing models. The top model (cos_101) in terms of overall accuracy is also the top model in terms of True Negative Rate.

Given the success of the cos_101 method, it seemed that encoding incorrects as negatives is generally more helpful for the task of predicting future ITS answers in students. The Running Total Method (RTM) was then developed as a follow up to the cos_101 method and allows tuning of how much weight to give both corrects and

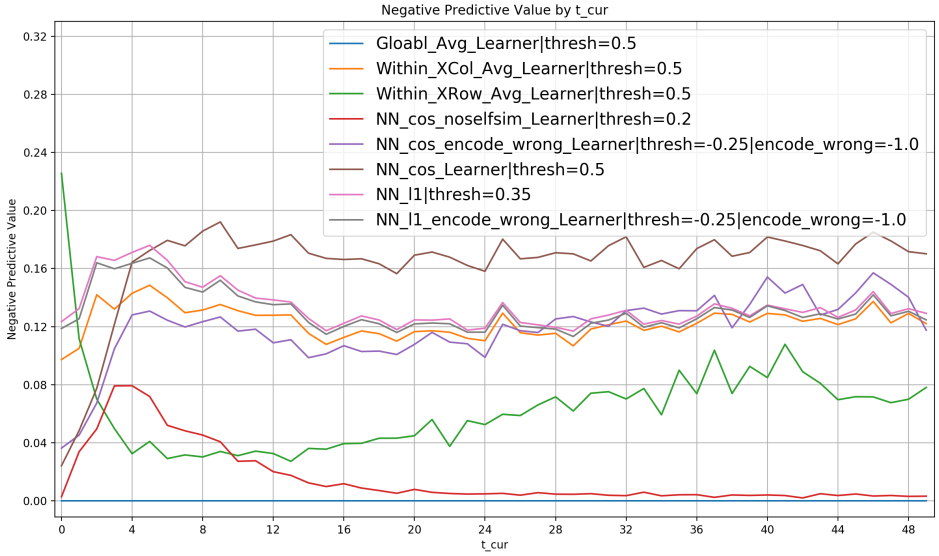

Figure 5.11: Negative Predictive Value results following testing paradigm in Section 5.1.1

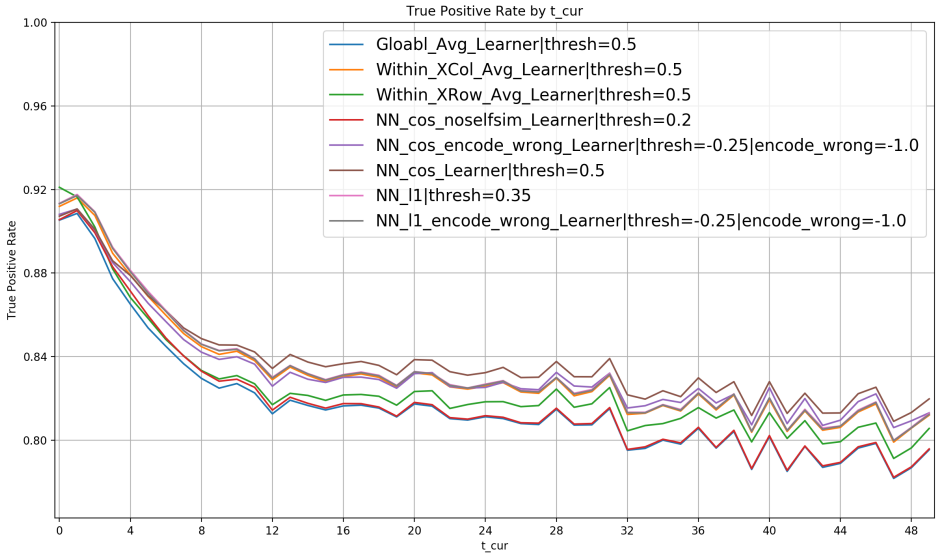

Figure 5.12: True Positive Rate Results following testing paradigm in Section 5.1.1

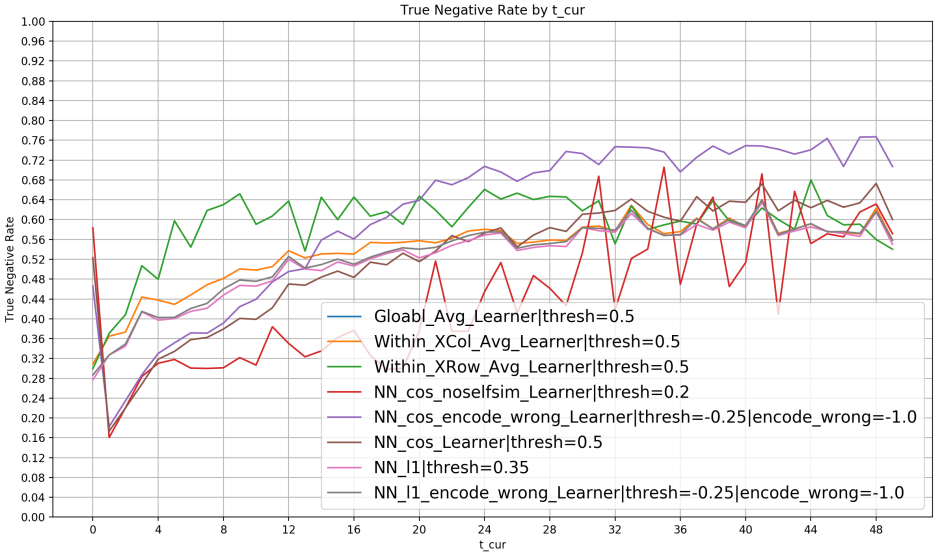

Figure 5.13: True Negative Rate Results following testing paradigm in Section 5.1.1

incorrects. The idea was that since the default equal weighting (corrects as 1's and incorrects as -1's) already saw success, allowing the weighting to be tuned to the data should give even more success. Surprisingly however, the RTM did not perform that much better than the cos_101 method in a overall accuracy comparison (Figure 5.14). RTM did have slightly better performance early on but by a few percentage points ($\approx 1\%$) but later one does worse/equal to cos_101.

The Positive/Negative Predictive Value and True Positive/Negative Rates for cos_101 vs. RTM can be found in Figures 5.15, 5.16, 5.17, and 5.18. In terms of these precision and recall metrics it seems the early advantage gained by RTM mainly comes from higher Negative Predictive Values and True Negative Rates before becoming equally accurate to cos_101 at $tm\_cur \approx 12$. **This again, affirms earlier results where the best methods had good True Negative Rates (recall of incorrect student answers) but this time shows that Negative Predictive Values is also important (precision on incorrect student answers, i.e. not having too many false negatives).**

The results of RTM vs cos_101 prompted further investigation into why it did not perform a lot better by looking at specific examples where the methods (cos_101 and RTM) treat students as not similar at all, somewhat similar, and very similar in Section 5.2.2.

5.2.2 Similarity Exploration

As discussed earlier, RTM was developed in response to the success of the cos_101 method so it was surprising that there was little/no improvement when RTM was implemented. This unexpected resulted prompted a look into specific cases of where both RTM and cos_101 matched students as not similar, somewhat similar, and very similar. The Cosine Similarity NN with {0, 1} encoding method (cos_01) is also included for comparison. The cos_01 method performed worse than cos_101 so it is important to know what characteristics about cos_101 made it better at predicting

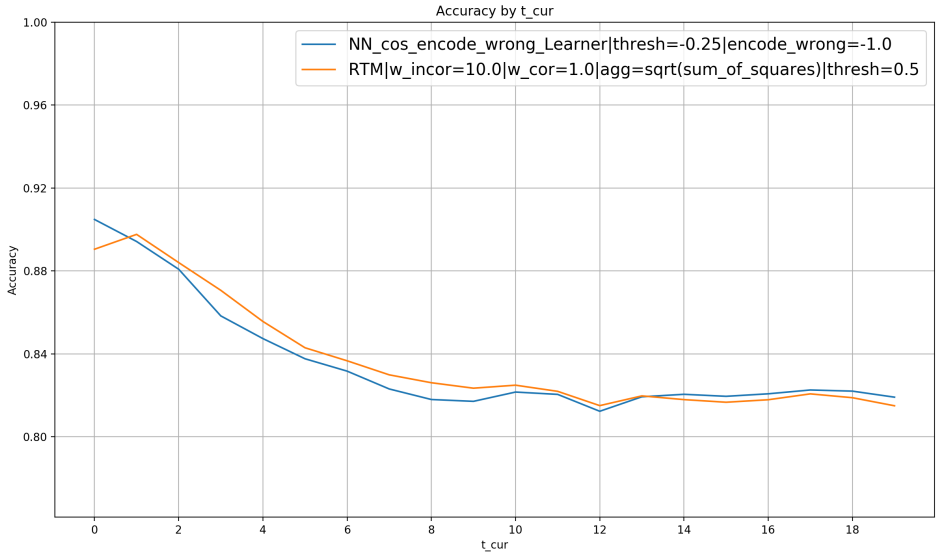

Figure 5.14: Accuracy of the cos_101 method vs the RTM

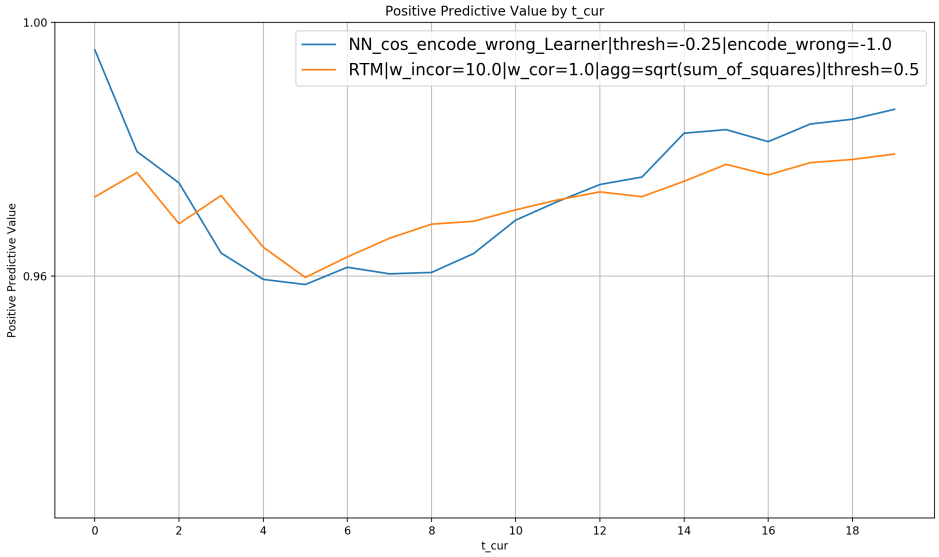

Figure 5.15: Positive Predictive Values of the cos_101 method vs the RTM

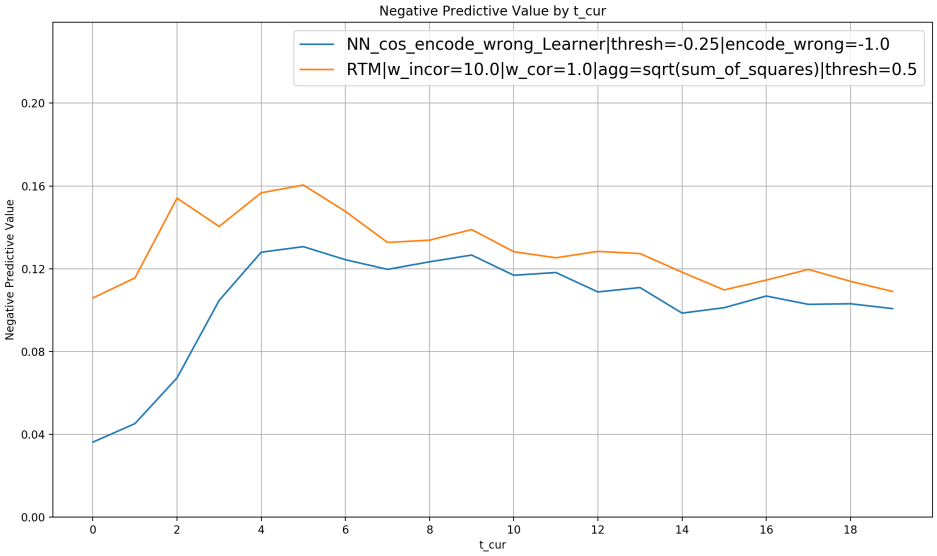

Figure 5.16: Negative Predictive Values of the cos_101 method vs the RTM

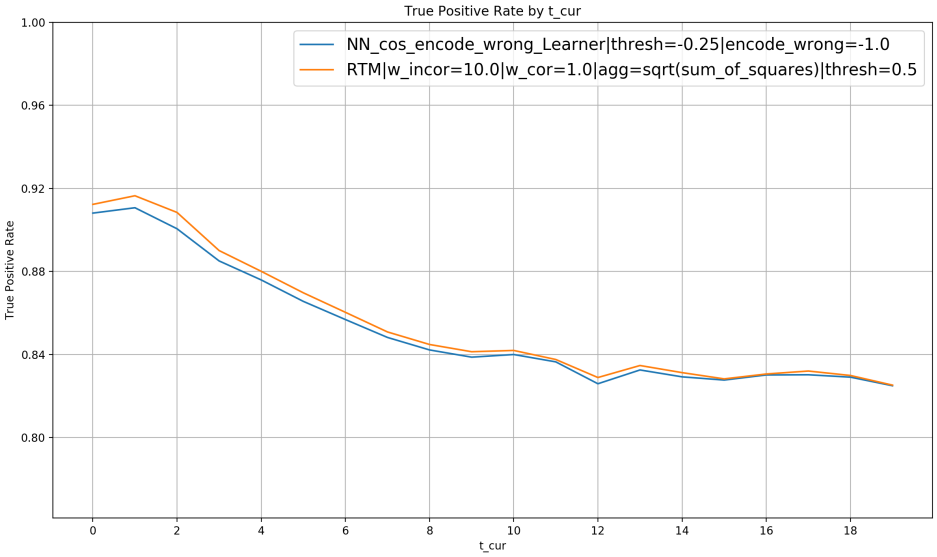

Figure 5.17: True Positive Rates of the cos_101 method vs the RTM

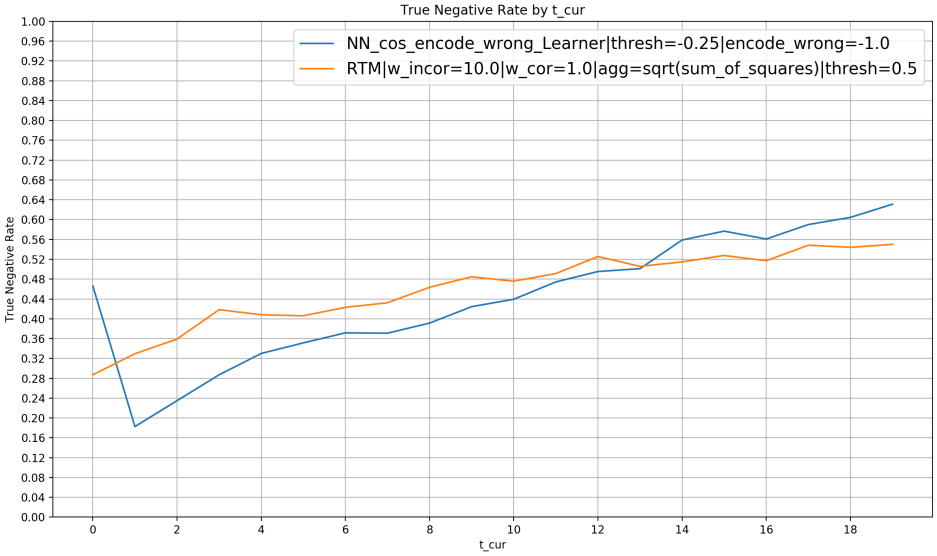

Figure 5.18: True Negative Rates of the cos_101 method vs the RTM

for ITS students.

The distribution of similarity scores between cos_01, cos_101 and RTM are shown in Figures 5.19, 5.20, and 5.21. Each score belongs to a unique student pair during the last iteration of the method ($t\_cur = 18$ predicting for $t\_cur = 19$). Since the cos_101 method allows for negative values in each student vectors, the cosine similarity values are allowed to be < 0. In contrast, cos_01 and RTM similarities are bound between $[0, 1]$. Percentiles in the figures refer to the least, average, and most similar student pairs (1st, 50th, and 99th percentiles respectively).

Most of the similarity scores are actually 0 for the two cosine methods with a large concentration of 0 similarities accounting for nearly 75% of the similarities based on the cumulative distribution. Immediately following this spike of of 0's there is a roughly normally distributed set of similarity scores between $[0, 0.2]$ for cos_01 and between $[0, 0.1]$ for cos_101. The main difference between the two distributions is that cos_101 also contains a few negative similarity scores before the cluster of 0's. **This indicates that having the ability to use negatives scores is likely the reason that the cos_101 method performed better than cos_01 in terms of accuracy.**

In contrast to the cosine methods, the distribution of similarity scores for RTM has a smooth left skewed distribution where a lot of the students are rated as being similar (Even the average 50th percentile similarity is 0.9 on a $[0, 1]$ scale). Since RTM performance is roughly just as good as cos_101, it shows that good performance can also come from similarity distributions that are smooth and not just ones that are concentrated as in cos_101.

Figures 5.22 and 5.23 show samples of student vector pairs for the 1st, 50th, and 99th percentile similarities for the cosine methods in terms of $X_{correct\_latest}$ values. For cos_01, the 1st and 50th percentile student vector pairs are the same due to the fact that $> 50\%$ of student pair similarities were 0. In cos_01, these students that were considered very dissimilar by cos_01 had completely opposite values. Any

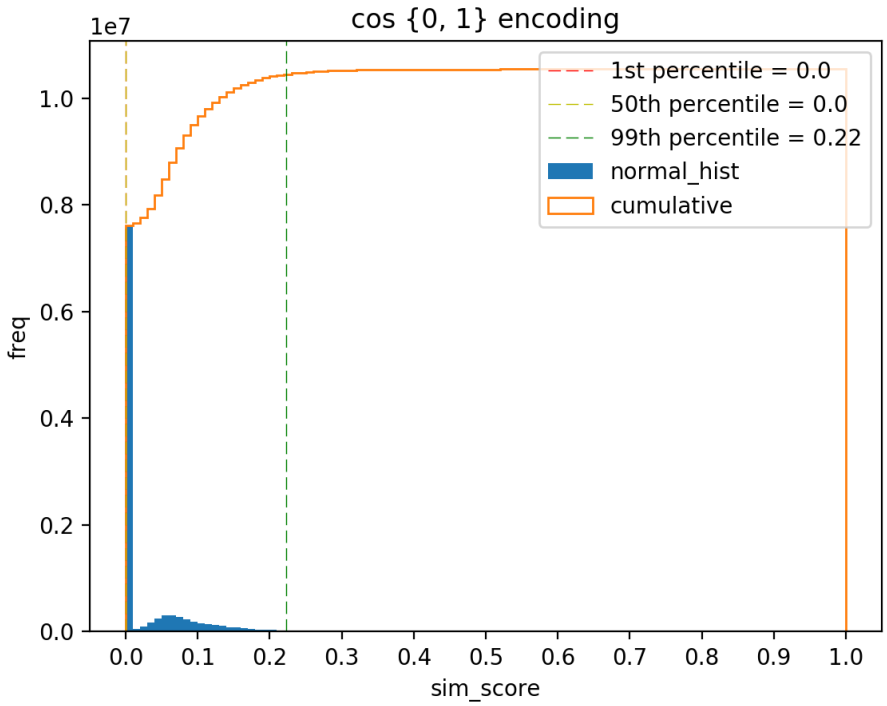

Figure 5.19: Distribution of similarity scores for the cos {0, 1} method

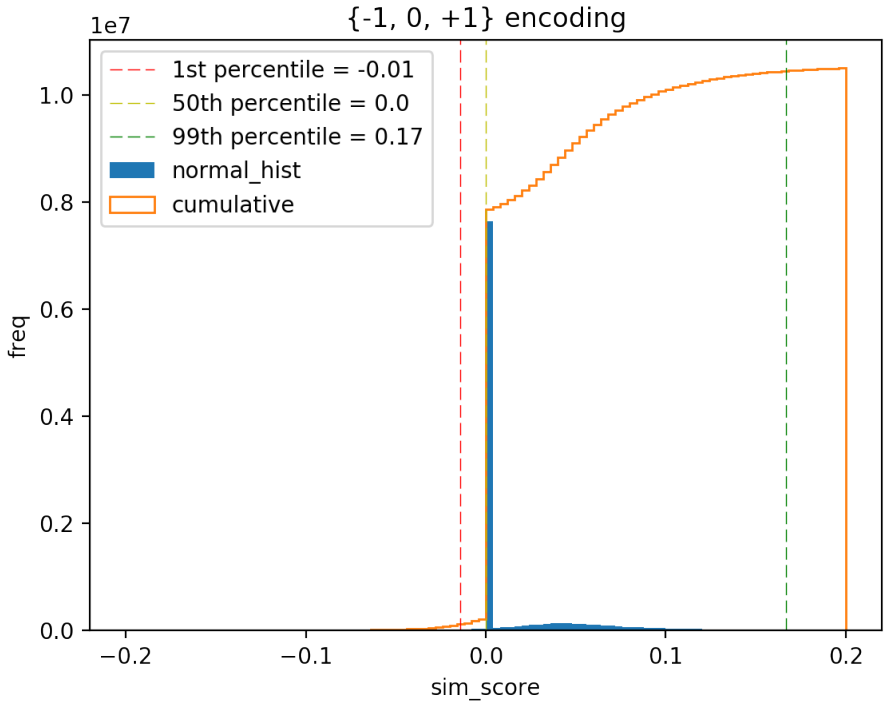

Figure 5.20: Distribution of similarity scores for the cos_101 method

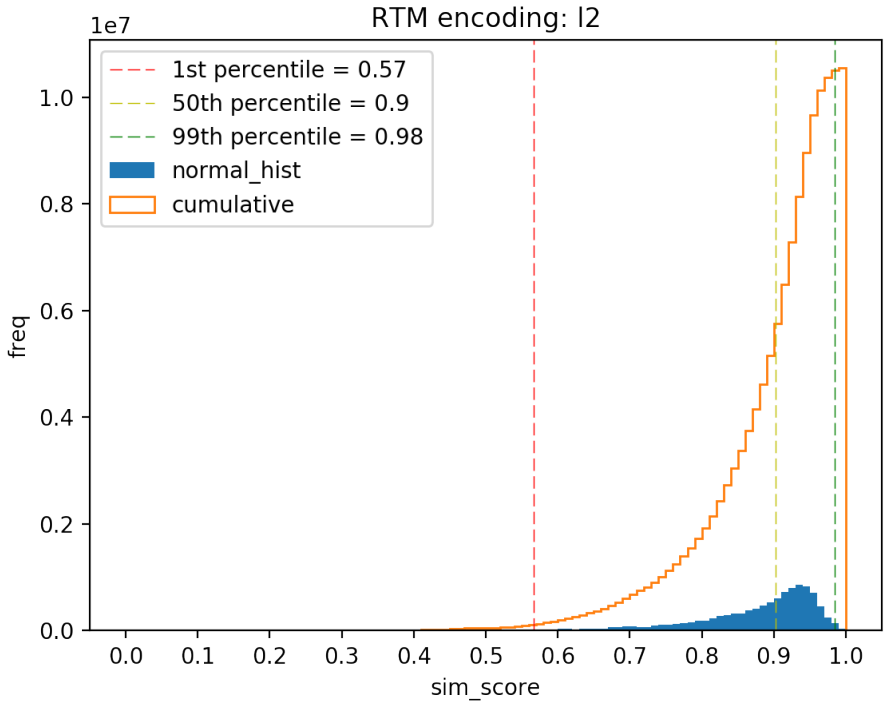

Figure 5.21: Distribution of similarity scores for the RTM

problems one student got correct the other either didn't answer or got wrong. The students that were considered more similar have more overlapping correct answers to the same problem-steps. For cos_101, a good similarity essentially boils down to one student having not answered as many problem-steps (Figure 5.23, top right subplot) which minimizes the difference between student vectors. This is because every problem-step not answered is simply encoded as a 0 which has a difference of 1 regardless of whether or not the other student got the same problem-steps correct or incorrect (1 or -1 respectively). A non-answered problem-step is also the most common encoding since the student vectors are sparse making it higher probabilistically to match element-wise with another students vector.

Figures 5.24 and 5.25 show samples of student vector pairs for the 1st, 50th, and 99th percentile similarities for RTM in terms of $X_{correct\_cnt\_latest}$ and $X_{incorrect\_cnt\_latest}$ values. Similar to cos_101, the most similar student pairs have a low amount of total problem-steps answered wheras in the lower performing cos_01 method this trend is not observed. This is not a surprise since both RTM and cos_101 perform at around the same level in terms of accuracy. **These results from both cos_101 and RTM suggest that the volume of problem-steps (questions) answered by students is an important factor to consider when predicting future student correctness in ITS students.** This is not a total surprise as one can imagine the difference between each students academic performance being closer early one and becoming larger as the semester drags on and their time management, concentration levels, and learning skills are tested on increasingly more advanced topics.

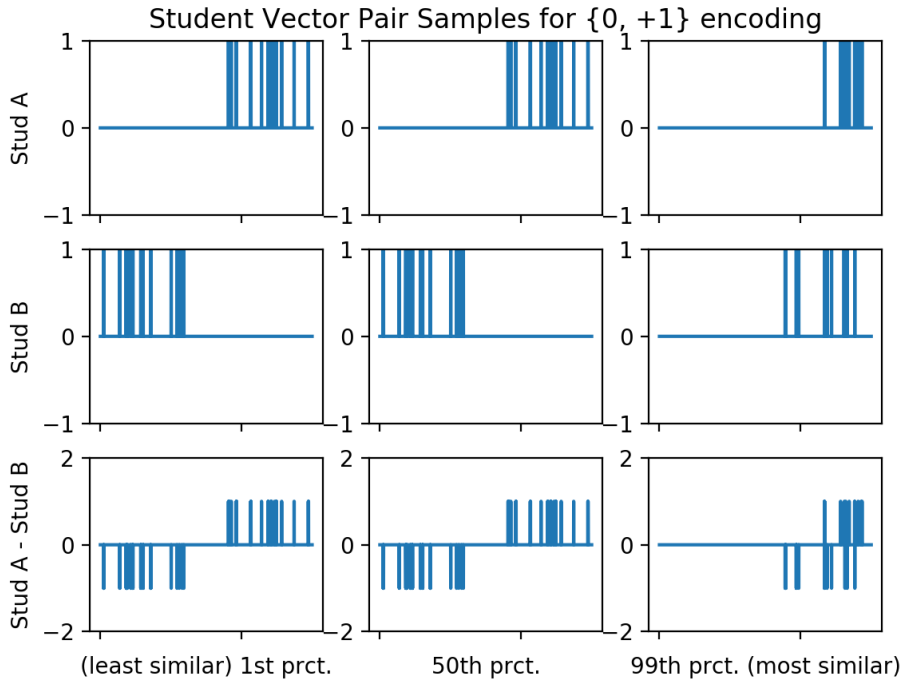

Figure 5.22: Sample student vectors of $X_{correct\_latest}$ from students who were matched at 1st, 50th, and 99th percentiles (left to right) from cos {0, 1}'s similarity matrix. Each graph shows the values of every unique problem-step (along the x-axis) in the KDD ITS for the given student pair (student_A and student_B) that make up the similarity score. From top to bottom the graphs are for student_A, student_B, and their difference.

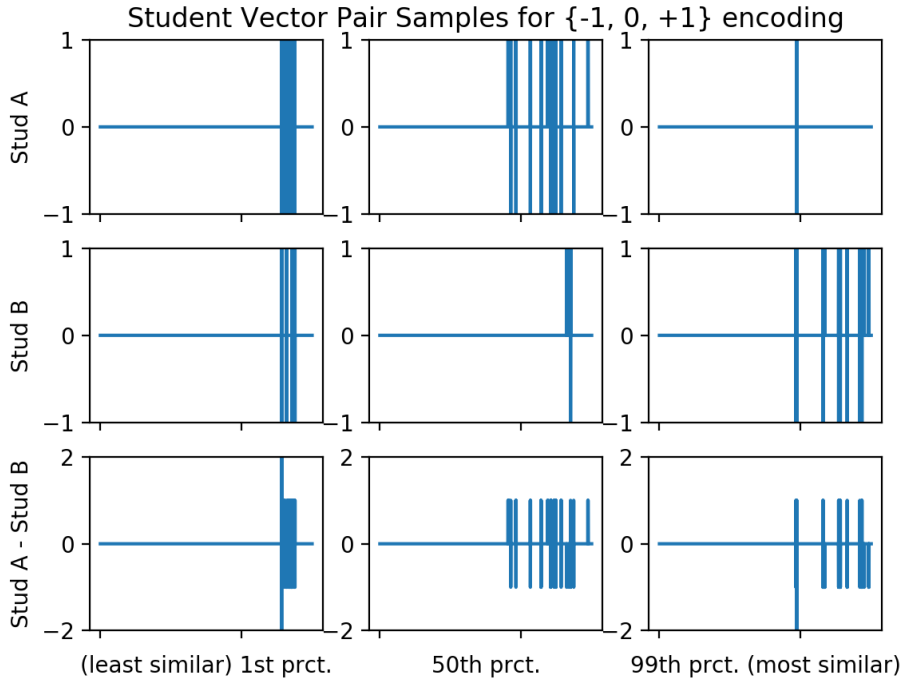

Figure 5.23: Sample student vectors of $X_{correct\_latest}$ from students who were matched at 1st, 50th, and 99th percentiles (left to right) from cos {-1, 0, 1}'s similarity matrix. Each graph shows the values of every unique problem-step (along the x-axis) in the KDD ITS for the given student pair (student_A and student_B) that make up the similarity score. From top to bottom the graphs are for student_A, student_B, and their difference.

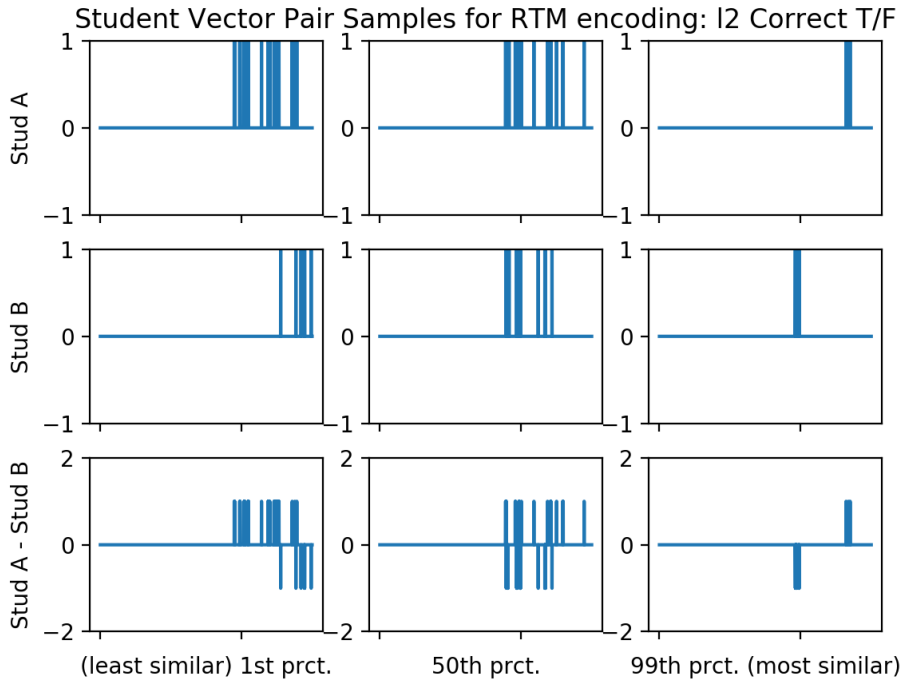

Figure 5.24: Sample student vectors of $X_{correct\_cnt\_latest}$ from students who were matched at 1st, 50th, and 99th percentiles (left to right) from RTM's similarity matrix. Each graph shows the values of every unique problem-step (along the x-axis) in the KDD ITS for the given student pair (student_A and student_B) that make up the similarity score. From top to bottom the graphs are for student_A, student_B, and their difference.

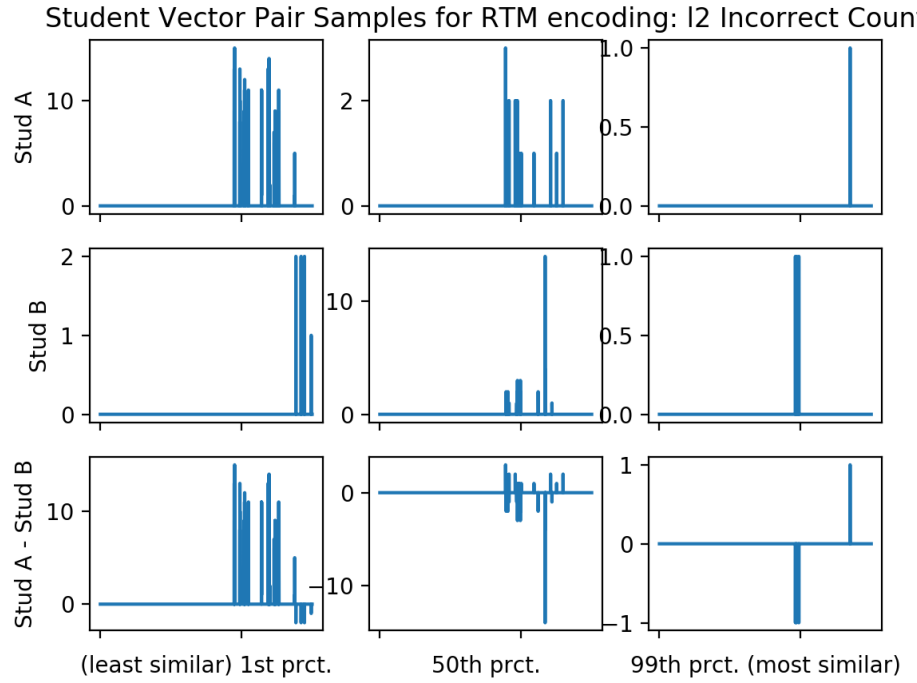

Figure 5.25: Sample student vectors of the boolean $X_{incorrect\_cnt\_latest} > 0$ from students who were matched at 1st, 50th, and 99th percentiles (left to right) from RTM's similarity matrix. Each graph shows the values of every unique problem-step (along the x-axis) in the KDD ITS for the given student pair (student_A and student_B) that make up the similarity score. From top to bottom the graphs are for student_A, student_B, and their difference.

Chapter 6

Conclusion

6.1 Summary of Contributions

In this thesis a new and more realistic way of evaluating predictive models for ITS systems was developed. The 2010 KDD Cup did not provide a realistic testing paradigm by allowing contestants to always train on a near-full semesters worth of data before predicting. In real life, an ITS predictive model has to predict decently well at various points in the school semester. Under this new testing paradigm, multiple NN-based CF methods were experimented with over incrementally larger training sets that simulated the time progression in a normal semester. After all experiments, the best models achieved a $\approx 2\%$ accuracy increase over baselines.

Since most students tended to answer correctly most of the time ($\approx 87\%$ mode accuracy on a per student basis) a good model needed to be able to be able to recall when students are wrong. The recall of predicting incorrects was represented through True Negative Rates (TNR) where the top model did indeed have the highest TNR. This result shows the important of recall when predicting incorrects in the ITS scenarios with innately high student accuracies.

After carefully examining specific student data for some of the top methods (cos_101 and RTM), it was shown that one of the likely factors of success for methods that performed above baseline is their ability to encode incorrect problem-steps as distinct from problem-steps that have not been completed. For future ITS applications it would be wise to treat them as distinct. For example, in the cos_101 method incorrects are treated as equal in weight to corrects but in the opposite direction (-1 vs. +1). In the RTS method corrects and incorrects had independent weight to the "distance" between two students which were tuned to the data.

Lastly, the most similar students on the best performing methods (cos_101 and RTS) achieved high similarity due to a low overall amount of problem-steps answered, but this trend was not observed in worse performing methods (cos_01). This suggests that a method should takes into account the volume of problem-steps each student has taken to achieve good ITS prediction performance, at least on the KDD dataset.

6.2 Directions for Future Work

Unfortunately, the 2010 KDD Competition results were not comparable with the results of this thesis as the methodology for testing predictive model performances were completely different. Since this thesis was the first time using an incremental

testing paradigm on the KDD ITS dataset, future work should adopt this framework when testing new predictive models of student correctness as it is closer to the real use case of ITS. There is definitely work be done with respect to alternative NN-based CF models as not all of them were explored here. As mentioned in section 5.1.2, there was a potential error in the implementation of RTM in this thesis so future work should also verify the effectiveness of RTM by carefully implementing it based on the mathematical definitions presented in section 4.5. The general direction that has seen success in this thesis is to go forward with trying to incorporate incorrects and corrects as independently weighted factors (as seen with cos_101 and RTM). Also, future researchers of NN-based CF models for ITS datasets should account for answer volume as that was seen to be important to the above baseline methods of this thesis. Hopefully, any new work that adopts these suggestions can help make better ITS predictors that work in real life cases to save future generations of students time and frustration when learning about the world.

Bibliography

Altman, Naomi S (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". In: *The American Statistician* 46.3, pp. 175–185.

Cen, Hao, Kenneth Koedinger, and Brian Junker (2006). "Learning factors analysis–a general method for cognitive model evaluation and improvement". In: *International Conference on Intelligent Tutoring Systems*. Springer, pp. 164–175.

Dede, Chris (1996). "The evolution of distance education: Emerging technologies and distributed learning". In: *American Journal of Distance Education* 10.2, pp. 4–36.

Herlocker, Jonathan L et al. (1999). "An algorithmic framework for performing collaborative filtering". In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pp. 230–237.

Hyndman, Rob J and Anne B Koehler (2006). "Another look at measures of forecast accuracy". In: *International journal of forecasting* 22.4, pp. 679–688.

Koren, Yehuda (2009). "The bellkor solution to the netflix grand prize". In:

Linden, Greg, Brent Smith, and Jeremy York (2003). "Amazon. com recommendations: Item-to-item collaborative filtering". In: *IEEE Internet computing* 7.1, pp. 76–80.

Stamper, J. et al. (2010). *bridge_to_algebra_2008_2009. Challenge data set from KDD Cup 2010 Educational Data Mining Challenge*. URL: `http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp`.

Toscher, A and Michael Jahrer (2010). "Collaborative filtering applied to educational data mining". In: *KDD Cup*.

Yu, Hsiang-Fu et al. (2010). "Feature engineering and classifier ensemble for KDD cup 2010". In: *KDD Cup*.