Coursim: A Tool For Detecting Course Dependency

Yang Liu, Zhenge Zhao

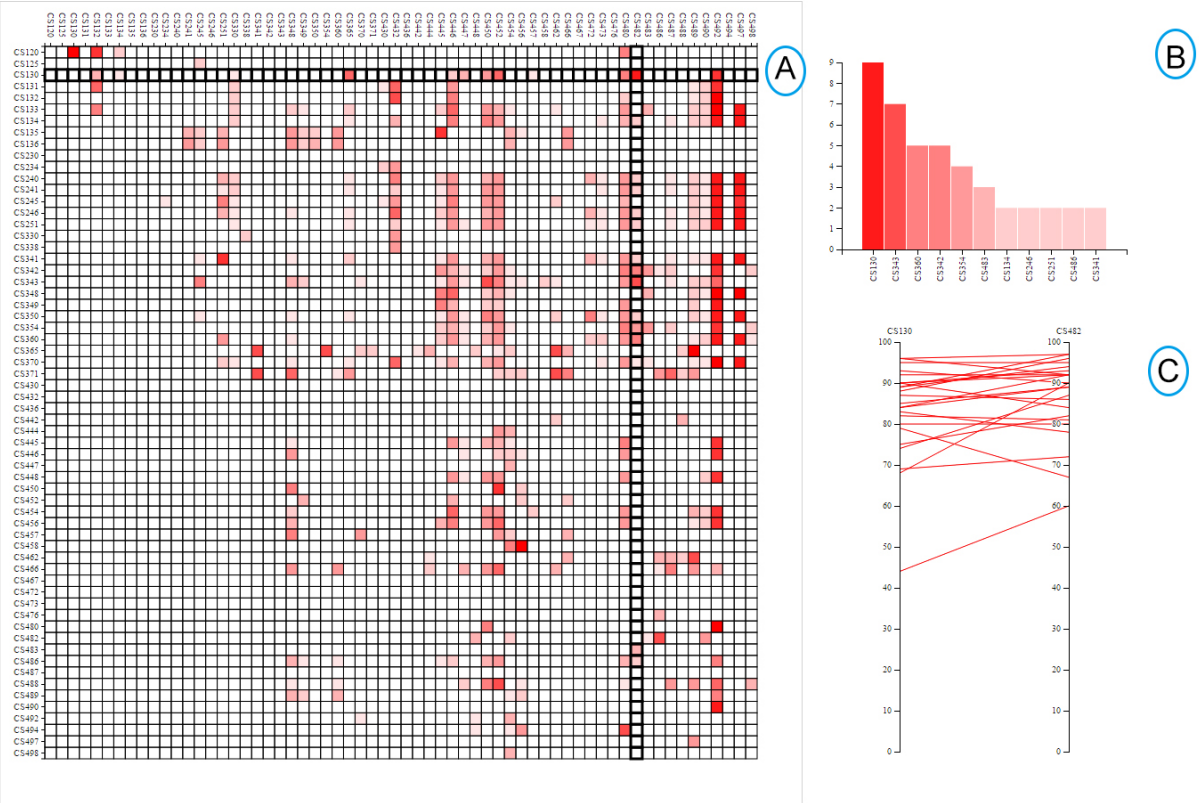

Fig. 1: The interface of Coursim: (A) adjacency matrix view to show the overall associations between all the courses in CS department; (B) bar chart view to depict top 10 courses which benefit the chosen course; (C) parallel coordinate view to demonstrates all the transition of scores of students taking both two courses

Abstract— Course Dependency can greatly affect a student's grade because of their content. Making one course the prerequisite of another may improve the student's performance in the latter course dramatically. But given the vast number of courses in a university, finding a good prerequisite calls for expertise in all the courses. It's a challenge for not only the student herself, but sometimes the academic advisor if he happens to know little about the particular course.
We proposed to compare students' historical grades to see the effect of setting prerequisite course. The visualization tool will use this as the weight in the correlation matrix. The tool includes a main view that shows the global relationship between each pair of courses, and a side view that shows details of one particular pair chosen by user.
We haven't got to the evaluation part yet. Plans will be discussed in Section 5

Index Terms—Visualization tool, Benefit Measurement, Acdemic Advising

1 Introduction

Navigating the curriculum of educational institution, fulfilling prerequisite and choosing between course options has been a feature of the educational environment dating back to Platos inscription "Let no one ignorant of geometry enter!" inscribed at the entrance of his academy [1]. In efforts to remain competitive, course options at institutes of higher learning have exploded, often offering hundreds of course possibilities that satisfy their general education requirements [8].

The real world scene is that coursework contents can be related, making one course as the prerequisite of the other will help students better learn knowledge and obtain better grades. Academic advisor, for example, deals with these issues a lot.

While some course dependencies have already been explicitly annotated in practice, more remain unclear and potential. In this work, wed like to design a visualization view to group related courses in clusters based on student enrollment history, and another view to show correlations of two related courses based on the discrete grades of students who have enrolled both courses.

The aims of this research are:

- provide a tool to visualize coursework contents similarity based

• *E-mail:[yangliu2014,zhengezhao]@email.arizona.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

on student performance.

- explore different diagrams to handle the complexity of information
- study similarity metrics of two courses based on grade history of one student and multiple students.

2 Background

Collaborative-filtering is a recommendation approach that uses similarity between users and the benefit they have received from items in the past to make recommendations. Three main approaches to collaborative filtering are memory-based, model-based and hybrid methods [9] . Memory-based approaches, specifically item-based recommender 2, use an item-user rating matrix to compute pairwise similarities between items. In the contexts of courses, enrollment roster, previous courses students taken and grades course give, is a natural way to reason about course similarity.

	CS101	CS202	CS301	CS401
User1	80	99	87	??
user2	79	88	86	85
User 3	76	95	91	86
User 4		78		
User 5	98	80	80	70
User 6			78	

Compute the score of CS401 by user 1?

Fig. 2: Typical student course grade table

However the recommendation approaches are not appropriate for our problems, since they are used for making predictions which means, the approach is designed to fill in the missing values in the matrix. For our case, however, we want to figure out how two courses are correlated with each other. We dont want to fill the matrix since it will blur the original data. What more, for each two courses we are only interested in the users who take both of them. Visualization is an effective and intuitive way to approach our goal.

Therere many existing similarity measures to compare two entities. Well known metrics include Euclidean similarity, Jaccard(Tanimoto) similarity, Pearson Correlation Coefficient, cosine similarity and Log-likelihood similarity etc. Euclidean similarity measures the distance between courses grading vectors. Jaccard(Tanimoto) similarity is calculated by dividing the intersection of the sets by the union of those sets [7]. Cosine Similarity envision users ratings as points in space and measures the cosine of the angle between these lines drawn from origin to each point. The Log-likelihood similarity is a measure of how often items from 2 sets appear together versus how often they appear apart. Pearson Correlation Coefficient is a number between -1, 1. It measures the tendency of the rating vectors, paired one by one, and its typically used in early research papers. Its formula is given as $pearson-correlation(u,w) = \frac{cov(R_u,R_w)}{\sigma x \sigma y}$ where cov stands for covariance and σx stands for standard deviation of x. We are interested in the strong positive correlation and the positive correlation as illustrated in 3.

For our work, Euclidean similarity is not suitable because courses taken by more students will be added more distances. Jaccard(Tanimoto) similarity and Log-likelihood dont count grades students get. We choose Pearson Correlation coefficient to indicate the similarity between two courses and based on that, we build our item-based recommender system and provide a way to measure the benefits between two courses.

3 Related Work

The problem of coursework similarity has been studied in the context of course recommendation system. Bendakir et al. [2] proposed a recommendation system based on decision tree of course history. Their approach, however, does not consider students' grades at all. Thus, their tool may wrongly correlate totally different courses simply due to historical mistakes. Sandvig et al. [7] did use the GPA information, but GPA, as an average metric, doesn't say much about each specific class.

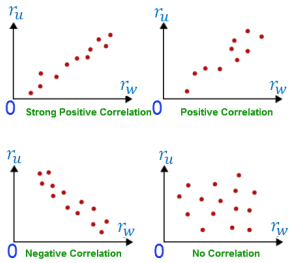

Fig. 3: Pearson Correlation Illustrated

When it comes to the visualization problem. Since our goal is to cluster similar classes together, a node-link diagram naturally jumps into our mind. D3 library has a force-directed graph that is close to our needs. But we are hesitant about its fisheye distortion and curved link variant because these variants make it hard to click on nodes or edges for further details. We are also aware that force directed drawing is criticized for local minima. A multilevel approach [11] might fix it but we are not focusing on algorithmic style improvement in this proposal.

Other well-known techniques include Rheingold-Tilford Tree, whose tree hierarchy is too constrained to express clustered nodes [6]; arc diagram, whose purpose is to highlight existing cycles. We investigate but decide not to use them.

Eventually, we decided to use adjacency matrix instead of node-link diagrams, inspired by the study of matrix and graphs [5, 10]. We also looked at the admirable work by F.Du et al. [4]. Their tool looks similar to ours at first glance but aims at student progress perspective rather than our course content centered perspective. However, we find their visual design interesting and could be one potential future direction of our tool.

We heavily use the well-known d3 library [3] to create our tool.

4 Visual Design

Our ultimate goal, is to design a visualization tool for understanding courses interactions. Our target users are professors, students and academic advisors. We talk with graduate students and professors about what they need to summarize our tasks as below:

T1 Overview of clusters of correlated courses

T2 For a specific course, find the courses which potentially benefit it and compare the importance of these courses

T3 For two selected courses, show details of student grades

We preprocess our original data and choose to use all the course records of computer science department for our implementations. We carefully select the visualization techniques which can be easily understood by users. Our system has three major view: the adjacency matrix, bar chart view and parallel coordinates view. In this section. We describe all the works we have done and the details of Coursim.

4.1 Data Processing

Towards this end, we performed a pilot study using student data from a Canadian research university. This data included all students who had taken a computer science course at that university between September 2001 and December 2011.

The data was in a comma-separated file made up of rows with 6 fields: a unique, anonymized student identifier, the term (which could be Spring, Summer or Fall and the year), the subject ID (such as CS or ENGL), the course code (such as 101), the percentile grade received and the students major.

The data needed to be preprocessed in preparation for the similarity calculation and the visualization. A unique, sequential identifier was added to each line. There was a variety of identifiers for problems

students had in a course, such as WD indicating withdrawal, DNR indicating the final exam was not written. These were replaced with 0 so that there would be a consistent, natural number domain for the grades. Some of the course codes had an optional letter suffix, which would indicate if it was offered on-line or at another campus (for example). These were removed and all courses with the same code were treated as the same value.

We removed courses and corresponding records which less than 10 students take in the data set on the basis that there was insufficient pairwise to measure the interactions between two courses. After these adjustments, the training set consisted of 37,392 students with data about 468,632 courses they took. There were 2,326 unique courses in the dataset.Specifically, for the computer science department, there are 61 courses, 2861 students, in total 38751 records.

ID	studentId	Term	CourseId	CourseNumber	Mark
4357	107482	1081	BUS	121	58
4358	107482	1081	MATH	127	60
4359	107482	1089	CS	100	86
4360	107537	1089	MATH	137	60
4361	107537	1091	CS	115	63
4362	107537	1091	ECON	101	68
4363	107537	1091	ECON	102	0
4364	107537	1091	MATH	135	71
4365	107537	1091	MATH	138	43
4366	107537	1095	AFM	101	69
4367	107537	1095	ECON	102	68
4368	107537	1095	MATH	136	62
4369	107537	1095	MATH	138	76
4370	107537	1099	CS	116	76
4371	107537	1099	ECON	201	50

Fig. 4: Original data table

4.2 Adjacency Matrix View

The Adjacency Matrix view enables users to grasp a global view of the all the courses. We use D3 library which contains the existing adjacency matrix template [3]. Each grid in the matrix represents the potential benefits the column course may get from the row course. Each row shows that, for a specific course, all the courses it may have a positive influence on. Each column depicts that, for a chosen course, all the courses which may benefit it. We build a mouse over in this view, when the users move their mouse on the grid, the column and the row associated with that gird chosen will be highlighted with a bold border which will make it easier to obverse the column and the row. The view also consists a zoomable and navigation feature which can be used to filter the uninterested data based on the demand.

Adjacency Matrix view can clearly show the every possible connection between each pair of courses. Compared to node-link diagram or dagre layout [3], it has obvious advantages in terms of recognition and orientation. When the size of the data get big, the force-directed node link diagram or dagre layout will include a lot of cross edges and become messy. It is hard for users to localize the actual pair of courses they want, and identify the actual relations between them. But adjacency matrix can always navigate users to the right pair of data and is easy for users to explore the data set and find interesting patterns.

The color scale ranging from white to dark red indicates the similarity between each pair of courses. White grids mean either there are less than 10 students or the influence the one course has on the later course is negative. We talk about how we calculate the similarities measurement in the next section.

4.3 Benefits Measurement

In order to demonstrate the benefits a course(Pre1) may give to another(Course1), we want to build a measurement way based on students performance in both of the courses. The intuition is that if a student gets high scores in Pre1 and in the future semesters, they also get a high

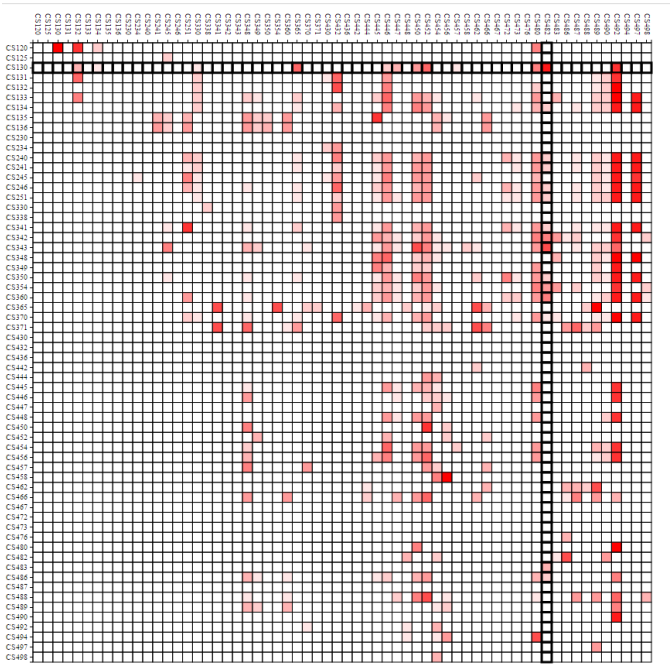

Fig. 5: Adjacency Matrix View

score in Course1, which may indicate the benefits between Pre1 and Course1. On the contrary, if the high score in Pre1 ends up with bad performance in Course1, the benefits may be few. Pearson-correlation seams a considerable method. But it turns out that it is not suitable in our case. The problem is that if for Pre1 and Course1, we extract the grades of the students who have taken both two courses in sequence and put them into two vectors. The results we get are not convincing. There are a few problems when applying Pearson-correlation. The first is that a lot of students performance is consistent. A student who can get high scores in one course will also achieve good scores in later courses. This makes the Pearson-correlation for each course very similar (around 0.7), which is hard to identify the benefits. Another problem is, the distribution of scores for different courses are also different. In some courses, 80 points may have been the one of best performance among all the students. The actual points depend on different professors and the different courses contain diverse standards.

In order to avoid the two problems mentioned above, we introduce a new benefit measurement based on recommender system. The motivation is that if a student(Jack) whose overall performance is not that good gets higher score in Course1 after taking Pre1 than the students whose overall performance are similar to Jack but without taking Pre1, we say Pre1 may be a benefit for course A. We choose Collaborative Filtering which can predict a students grade on one course based on the similarity to other students. More specifically, we choose item-based recommender which is more efficient than the user-based recommenders since we have much fewer courses than the number of students. To measure the benefit Pre1 may bring to Course1, we do as follows:

- Step 1: For each course pair (in order, for example, Pre1, Course1) taken by more than 10 students, remove the actual grades of Course1 of those students and take the rest of the records of the training data.
- Step 2 : Running an Item-based recommender using Pearson Correlation to predict the grades the students will get for Course1.
- Step 3: Calculating the average of the differences between the actual and predicted grades and take that as the similarity measurement.

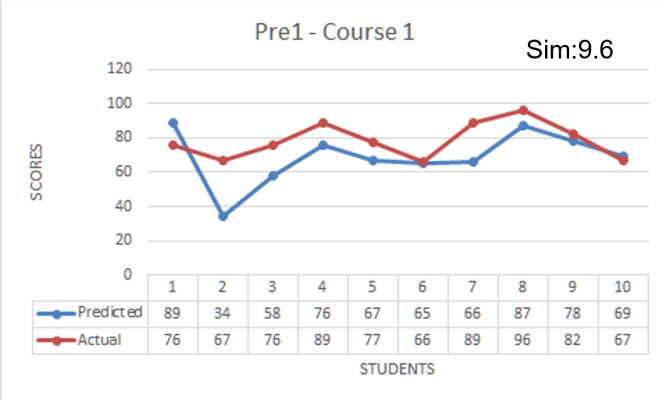

Fig. 6: Our method based on item-based recommender system, Pre1 clearly benefit Course1 with a beachmark 9.6

From Fig. 6, we can clearly see that most of the 10 students who have taken both Pre1 before Course1 have better actual scores than the prediction which is based on the overall performance. In other words, compared with the students whose overall performance is similar but without taking Pre1, these students have gotten 9.6 more points on Course1 averagely, which is a great improvment. Therefore we can say, Pre1 should benefit Course1 even without knowing the contents of both two courses.

This new measurement gives us a better way to measure the actual benefits a course may give to the other. In stead of using Pearsonal Corrleations, it is a more solid way because it can tolerate the influence of the students consisitent performance and also avoid the direct comparsion using scores of different courses. Using this method, we get a much better distribution of the overall benefits measurement. Specifically, for the computer science department, there are 1191 pairs of courses have benefit benchmarks.The benefits points range from -17.9 to 12.6(Fig 7). Minus points mean the negative influence and positive points mean active influence. In the current version of our adjacency matrix , we ignore the negative points, which we also also color it as a difference color. This maybe one of our future work.

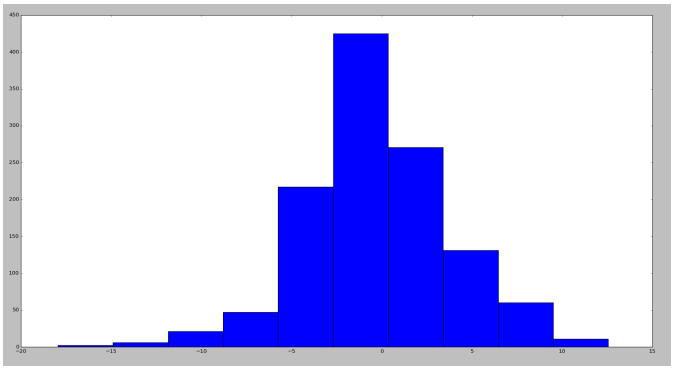

Fig. 7: The distribution of benefits benchmarks for CS courses, ranging from -17.9 to 12.6,bigger the number, more benefits may exit

4.4 Bar Chart View

The bar chart view details the course users are interested in. The interaction is built between the adjacency matrix and the bar chart. When the user finds one course he is interested and wants to see all the courses may benefit this course. They can click on any grid along the column of that course. The bar chart will pop up. It reveal the top 10 courses which may benefits the selected course and order them by the benefit benchmarks. The color of each bar is the same as the corresponding grid in the adjacency matrix to make sure the consistency and. This style avoid the possible confusion when users are navigating to the bar chart view. Using this bar chart view, users can easily find out for the specific course, which one has the highest benefit benchmark. Also, for a pair of courses, whats the rank of the pre-course, is that actually a good choice? The view helps users to figure out these problems and make good decision for choosing good prerequisites.

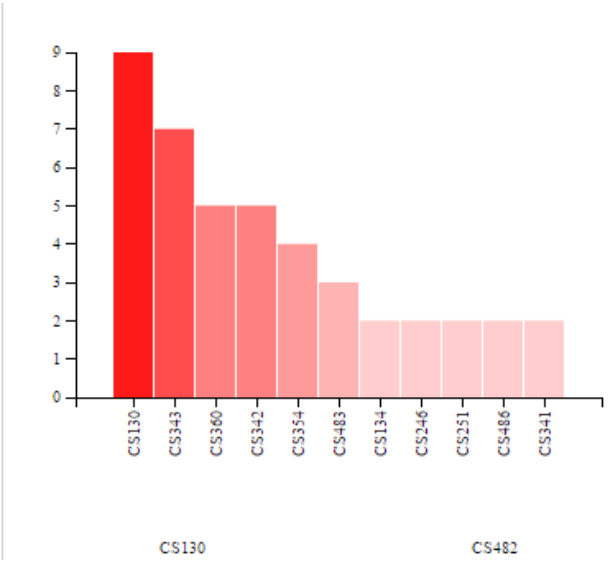

Fig. 8: Bar chart view for course CS482

4.5 Parallel Coordinates View

For a pair of courses we are interested in, we use a parallel coordinates graph to show the specific grades changes for students who have taken both courses. Each lines start point is the score that a student gets in the previous course, and the end point is the score the same students gets in the later course. The interaction is build between the adjacency matrix and the parallel coordinates. When a gird is clicked, a parallel coordinates will be shown in this view. Instead of focusing on the course level and getting a summarized benchmark for each pair of courses. The parallel coordinates will allow users to explore the details and show all the trends of each students performance between theses two courses. Users can clearly observe the changes of each student which can be a good reference when they are considering the benefit between two courses. For example ,from Fig. 9, the overall benefit CS240 may bring to CS492 is high positive. But we can also find some obvious outliners: a student getting almost 98 points gets 65 in the CS492, another student whose score for CS240 is 78 ends up getting 8 points in CS492. These are some anomalies which users can easily find out with the parallel coordinates. Fo r the future work, different colors are utilized to show the grade section for the course. And we also need to build interaction between bar chart and parallel coordinates.

5 Evaluation

In this section, we show some user studies and the results of your visualization. Since our data comes from a real world university, we can actually know the contents of each course and by consoling computer scientists, we can see if Coursim actually gives us some good perquisites for a chosen course. We also show our evaluation plan here for doing an actual user study.

5.1 Usage Scenario

This case study shows three different usages of this tool. First, when a student comes to his advisor and asked him a question like what courses do I need to take this semester if I eventually want to take CS482? The professor doesnt know how much about CS482, actually the professor is just a Math Doctor which his student takes the CS as a minor. But using our tool, when can find the course CS482 in the

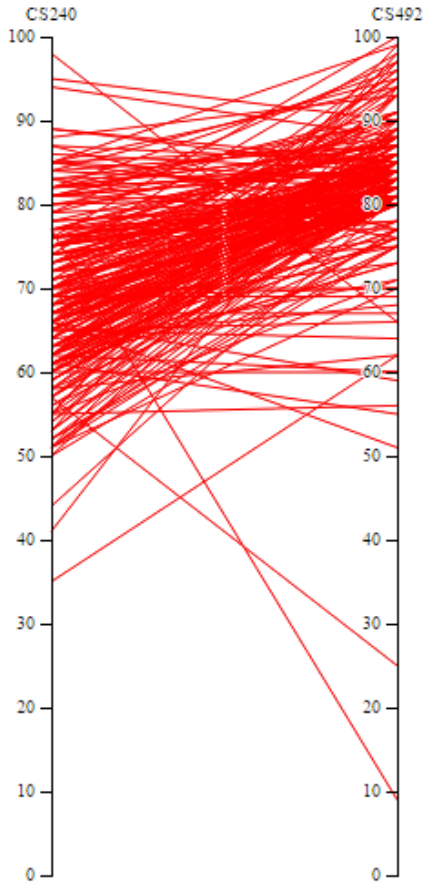

Fig. 9: Parallel coordinates graph for CS240-CS492

adjacency matrix and click the column. Then in the bar chart (Fig. 8), he can easily figure out that the CS130 and CS343 have the highest benchmarks. Then he moves to the matrix and clicks on the gird of pair CS130 CS482 and CS343-CS482. The parallel oordinates tell him the most the students are enhanced if they have taken CS130 and CS343. With this knowledge in mind, he can advise the two courses to his students without knowing what actually the two courses are. What's more, another interesting question a student may be wondering is that if he has taken a course, which course he may take next which will be easiler for him to get a high score. For this purpose, he can find the row of that course, and go through the row and find out the red girds. Then he can click on it and see the details in the bar chart and parallel coordinates. The last example is that it can be used to evaluate the setting of the current prerequisites. We can easily achieve that by finding out the pair of courses and watch its actual ranks in the barchart. Using the bar chart and parallel, a professor can figure out if the current prerequisites is a good. Are there any other hidden prerequisites which have a higher benefit than the current one? Or whats actual performance looks like for students taking the prerequisites?

After plotting the pictures, we want to find out if the benchmarks work well. We pick up some pair of courses and find out the contents and titles of the universitys websites. The results shown are quite interesting. For example CS 365 and CS489, the benchmark we calculate is 11.2. CS365 is Model of Computation and CS 489 is a topic course which includes Big Data Infrastructure, Complexity of Computational Problems and Introduction to Machine Learning, which include a lot of model computation.Some other pairs of courses with high benchmarks are also convinced by different domain specialists there are a lot of connections between these pairs. For instance, CS 135 Designing Functional Programs and CS 445: Software Requirements Specification and Analysis, CS 458: Computer Security and Privacy-CS 456 Computer Networks.

5.2 Evaluation Plan

As we stated in the original proposal, an intermediate evaluation test would be comparing documented dependency to the overview drawn by our tool. The overview should be a superset of documented dependency but not close to a universal set. We can worry less about false positives because the prototype doesn't have a high density of highlighted cells.

The above test can be automated relatively easily. We may not even need to get user involved. The ratio of identified documented dependency over all documented dependency should be the prime and only thing to look at in my opinion.

After that, an evaluation of false positives may be carried out. I see two possibilities: first is to contact the academical advisor from our dataset's university to confirm the possible dependency revealed by our tool; second is to find a local academical advisor who is willing to share her data and apply our tool to it. In fact, doing both would be more convincing.

Eventually, a user study should be done. We should have an academical advisor's feedback and improved the tool at this moment. We then ask the academical advisor to use this tool with students seeking advice, when they choose their courses. We need to be careful about the user groups here. Do we want to work with the same academical advisor all the time? Do we care about gender balance, student year(freshman, senior) balance?

I think we should share this tool with a handful of academical advisors, to avoid the tool being too personalized. When their feedback conflicts, we record their argument first and ask them if they are willing to discuss further. We end up either solving the issue or having one implementation but understanding the alternatives.

For a balanced student group, the question boils down to who's going to actually use this tool. If the tool is going to be used sololy with both student and academical advisor in presence, then we should accept whoever comes to the office and don't worry about balance at all. If the tool is going to be offered in the school system open to all students, we should invite students to evaluate it online in addition to academical office deployment. I'm more inclined to the first approach simply because it is easier to come true.

Existing tools should be considered in doing all these evaluations. A survey should be taken, asking users to rate all three views on a Likert scheme, with an optional text box of why they like or hate any part. Depending on the skewness, we maybe need to throw away the top 5% high and low gradings. The text feedback should be synthesized to explain the gradings of each view and discussed possible improvement.

6 Discussion

The project is successful overall. We learned valuable lessons throughout. They are about both general research and visualization research.

- Knowing what's out there in the field is important. Researchers at entry level often start out experimenting their own thoughts without studying literatures. Reference is more than citation. Readers will not be convinced unless we can relate other people's work to ours and explain why, how, and what we are doing differently.

- During this project, we find it really helpful to summarize the problems of published papers. These problems serve as good reminders of common pitfalls even for experienced researchers and reviewers. We also think going back to the first slides at the beginning of semester to be useful. It helps keep the project going in a regular track.

- For the coding part, d3 is a quite different language from C-style ones. We started out skimming preliminary tutorials and moved on to modify the gallery codes. This approach turned out to be inefficient and painful. I personally misunderstood chaining

methods until mid term. If one happens to be familiar with C-style languages like me, d3 is likely to deserve more time to acquire than you anticipate.

- Time management is crucial. It's dangerous to wait till last minute. We usually set for ourselves a dummy deadline, one or two days prior to the real deadline. We thus apply major fixes early and have enough time for minor fixes in a relaxed mood, which actually improves the quality of those minor fixes.

7 Conclusion and Future Directions

Now let's review our tasks listed in Section 1. For task 1, we carefully grouped students to compare courses with only related students. For task 2, we tried standard force-directed layout and several kinds of filtering and decided to use adjacency matrix. For task 3, a recommender system is built to predict benefit of course A as the prerequisite of B.

The prototype looks promising for now, since it reveals that our initial prediction of matrix layout is wrong. This means that it is likely to be wiser than our intuition. We thus conclude that the project is successful.

For the work we should do but don't have time.

- Lock/unlock the bar chart and slope chart on mouse click. When the side views are unlocked, they are updated as mouse moves over the matrix cells.

- Incorporate color map to show negative values in the matrix. In prototype, a negative is not drawn as if two courses don't share common students.

- Compare students taking A and B with students taking A but not B. This should appear as another side view near the slope chart. In prototype, user can only see students taking A and B.

- Add interaction between slope chart and bar chart. If user clicks on one bar in the bar chart, the slope chart should show the corresponding cell in the matrix. In prototype, user has to find the cell by eye and click it to update the slope chart.

For the work we leave as future directions.

- Even with the adjacency matrix representation, the screen gets filled up when we study courses at university wide level. That brings us to focus on courses from one department only(a kind of filtering). Perhaps a selection box be added to support data for other department in future. It is also worth considering, if other selection, aggregation methods exist to show more data in the screen.

- Some universities use letters instead of scores, for scalability, a future work should be carried on how to deal with the grade without scores

- Show the bar chart of the students who don't take the pre-course for comparison

- Also a formal evaluation as discussed already should be carried out.

- Instead of showing just Major+Number as the course ID, we also want to include the course name and a brief introduction of the course so that users will know which course it actually is.

References

[1] W. Anglin. *Mathematics: A Concise History and Philosophy: A Concise History and Philosophy*. Readings in Mathematics. Springer New York, 1994.

[2] N. Bendakir and E. Aïmeur. Using association rules for course recommendation. In *Proceedings of the AAAI Workshop on Educational Data Mining*, vol. 3, 2006.

[3] M. Bostock, V. Ogievetsky, and J. Heer. D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, Dec. 2011. doi: 10.1109/TVCG.2011.185

[4] F. Du, C. Plaisant, N. Spring, and B. Shneiderman. Eventaction: Visual analytics for temporal event sequence recommendation. In *Proceedings of the IEEE Visual Analytics Science and Technology*, VAST '16, pp. 1–10. IEEE, 2016.

[5] M. Ghoniem, J.-D. Fekete, and P. Castagliola. A comparison of the readability of graphs using node-link and matrix-based representations. In *Proceedings of the IEEE Symposium on Information Visualization*, INFOVIS '04, pp. 17–24. IEEE Computer Society, Washington, DC, USA, 2004. doi: 10.1109/INFOVIS.2004.1

[6] E. M. Reingold and J. S. Tilford. Tidier drawings of trees. *IEEE Trans. Softw. Eng.*, 7(2):223–228, Mar. 1981. doi: 10.1109/TSE.1981.234519

[7] J. Sandvig and R. Burke. Aacorn: A cbr recommender for academic advising. Technical report, Technical Report TR05-015, DePaul University, 2005.

[8] B. Schwartz. *The Paradox of Choice: Why More Is Less, Revised Edition*. HarperCollins, 2009.

[9] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, Jan. 2009. doi: 10.1155/2009/421425

[10] F. Van Ham. Using multilevel call matrices in large software projects. In *Proceedings of the Ninth Annual IEEE Conference on Information Visualization*, INFOVIS'03, pp. 227–232. IEEE Computer Society, Washington, DC, USA, 2003.

[11] C. Walshaw. A multilevel algorithm for force-directed graph drawing. In *International Symposium on Graph Drawing*, pp. 171–182. Springer, 2000.