The alarming graduation statistics in higher education institutions have resulted in an increased demand on finding ways to improve the learning environment for students to help them graduate in a successful and timely manner. With the rise of data available about past students, machine learning researchers have been able to learn predictive models that solve different problems in the education domain. In this paper, we focus on the problem of course recommendation that aims to recommend to each student a set of courses from which they can register for in their following term, for the ultimate goal of improving the student’s GPA and graduation time. We first propose a different definition for the course recommendation problem statement, by focusing on recommending courses on which the students’ expected grades will help maintain or improve their GPAs. We then leverage two widely-known representation learning techniques, in order to learn the sequence by which students take courses and create better personalized rankings for students. Our experiments on a large dataset obtained from the University of Minnesota that includes students from 23 different majors show that the methods based on the proposed problem statement can better recommend courses on which the students are expected to perform well and that align with their degree programs. Moreover, the results show that the proposed methods achieve statistically significant results for course recommendation over the current methods.

1. Introduction

The average six-year graduation rate across four-year higher education institutions has been around 59% over the past 15 years [Kena et al., 2016, Braxton et al., 2011], while less than half of college graduates finish within four years [Braxton et al., 2011]. These statistics pose challenges in terms of workforce development, economic activity and national productivity. This has resulted in a critical need for analyzing the available data about past students in order to provide actionable insights to improve college student graduation and retention rates.

With the increasing amount of available educational related data, researchers have been able to apply different machine learning and statistical methods in order to analyze this data and improve the learning environment for the different stakeholders involved in education, e.g., students, advisors, and instructors. Some examples of the problems that have been investigated are: course recommendation [Elbadrawy and Karypis, 2016]...
In this paper, we focus on the problem of course recommendation. Many academic degree programs offer flexible plans, each containing a set of required (core) courses that need to be taken by all students, as well as a set of elective and liberal art courses, from which each student can choose a subset to take that interests them. Given a student in a higher education institution, the goal of course recommendation is to recommend a small set of courses to register for in their following term that align with their degree program requirements. This can help students and their advisers during the process of course selection during each term for the ultimate goal of successful and timely graduation.

Previous approaches for course recommendation (Bendakir and Aïmeur, 2006; Lee and Cho, 2011, for eg.) aimed to recommend a set of courses for each student for which they would most likely register in their following term, ignoring the grades that the student is expected to achieve on them. So, they treated the whole sequence taken by all previous students as the ground-truth data on which their different methods were learned. For example, Elbadrawy and Karypis (2016) developed several domain-aware methods for course recommendation, including methods based on popularity ranking, student-based collaborative filtering and matrix factorization. However, when estimating the parameters of these methods, they did not take into account the student’s performance in those courses and as such, these methods can recommend courses on which the student may actually perform poorly.

In this paper, we first re-visit the problem statement of course recommendation and propose a different definition that focuses on recommending only the good courses for students, i.e., the courses on which the student’s expected grades will most likely maintain or improve their overall GPA. The proposed definition aims to help students make better course selection during course registration than the traditional definition, by ranking higher the set of courses on which the student is expected to perform well on than those on which the student is expected to have low performance on. Then, based on this new problem statement, we investigate the use of two widely-known representation learning techniques, by customizing them for the proposed course recommendation problem. Specifically, the courses taken by each student are treated as a sequence of sets of courses, and each of the two representation learning techniques are trained on these students’ sequences in order to learn the sequencing of courses as taken by them. The course representations learned by these models are then used to create personalized rankings of courses for students that are designed to include courses that are relevant to the students’ degree program and in which students are expected to perform well. Specifically, each of these approaches learns two types of dense low-dimensional embeddings for each course, namely, previous and subsequent course embeddings, which represent the course when treated as a previous and a subsequent course, respectively. The representations of the previous courses taken by
each student are used to represent the individual student’s degree state, which is then used to rank the subsequent courses for that student. The first representation learning approach is based on Singular Value Decomposition (SVD), which is a linear model that learns a low-rank approximation of a given matrix. The second approach, which we refer to as Course2vec, is based on neural networks, and uses a log-linear model to formulate the problem as a maximum likelihood estimation problem.

We performed an extensive set of experiments on a real-world 16-year dataset obtained from the University of Minnesota, which includes students who belong to 23 different majors. The experimental results show that: (i) learning course sequencing using representation learning significantly outperforms other competing methods for course recommendation; (ii) differentiating between good and bad courses during learning the course recommendation methods is extremely useful for recommending courses on which the students are expected to perform well and also align with their degree programs; and (iii) SVD outperforms Course2vec in most cases.

1.1. Contributions
The main contributions of this work are as follows:

- We propose a new definition for the course recommendation problem in higher education in order to better suit the education domain. The proposed definition is intended to help improve the student’s GPA and graduation time than the current definition.

- We introduce representation learning to the course recommendation problem, by customizing two widely-used techniques to model the sequence by which the students take courses. To our knowledge, this is the first work to model course sequencing from students’ degree plans using representation learning techniques and use the learned course representations for personalized course recommendation.

1.2. Organization
The rest of the paper is organized as follows. Section 2 discusses the related work to this paper. Section 3 explains the proposed course recommendation problem statement as well as the proposed modifications on the existing methods in detail. Section 4 shows the experimental evaluation methodology. Section 5 discusses the experimental results. Finally, Section 6 provides some concluding remarks.

2. Related Work
In this section, we review the work done in areas of: course recommendation, course sequence discovery and recommendation, and representation learning.

2.1. Course Recommendation
Different machine learning methods have been recently developed for course recommendation. For example, Bendakir and Aimeur (2006) used association rule mining to discover significant rules that associate academic courses from previous students’ data. Lee and Cho (2011) ranked the courses for each student based on the course’s importance within
his/her major, its satisfied prerequisites, and the extent by which the course adds to the
student’s knowledge state.

Another set of recommendation methods proposed in (Parameswaran and Garcia-
Molina, 2009; Parameswaran et al., 2010; Parameswaran et al., 2010; Parameswaran et al.,
2011) focused on satisfying the degree plan’s requirements that include various complex
constraints. The problem was shown to be NP-hard and different heuristic approaches
were proposed in order to solve the problem.

Recently, Elbadrawy and Karypis (2016) proposed using the academic features to
improve the performance of three popular recommendation methods in the education
domain, namely: popularity-based ranking, user-based collaborative filtering and matrix
factorization. The academic features proposed are both student- and course-based, i.e.,
for the student, the features used are his/her major and academic level, and for the course,
the features used are its subject and level. These features are used to define finer groups of
students and courses and were shown to improve the recommendation performance of the
three aforementioned methods than using coarser groups of students. Both the popularity
ranking and matrix factorization methods that utilize the finer groups of students were
shown to achieve the best performance for course recommendation compared to the other
proposed methods with other groups.

2.2. Course Sequence Discovery and Recommendation

Though our focus in this paper is to recommend courses for students in their following
term, and not to recommend the whole sequence of courses for all terms, our proposed
models try to learn the sequencing of courses such that they predict the next-term’s good
courses based on the previously-taken set of courses.

Cucuringu et al. (2017) utilized several ranking algorithms, e.g., PageRank, to extract a
global ranking of the courses, where the rank here denotes the order in which the courses
are taken by students. The discovered course sequences were used to infer the hidden
dependencies, i.e., informal prerequisites, between the courses, and to understand how/if
course sequences learned from high- and low-performing students are different from each
other. This technique learns only one global ranking of courses from all students, which
cannot be used for personalized recommendation.

Xu et al. (2016) proposed a course sequence recommendation framework that aimed to
minimize the time-to-graduate, which is based on satisfying the pre-requisite requirements,
course availability during the term, the maximum number of courses that can be taken
during each term, and the degree requirements. They also proposed to do joint optimiza-
tion of both graduation time and GPA by clustering students based on some contextual
information, e.g., their high school rank and SAT scores, and keeping track of each stu-
dent’s sequence of taken courses as well as his/her GPA. Then, for a new student, he/she
is assigned to a specific cluster based on their contextual information and the sequence of
courses from that cluster that has the highest GPA estimate is recommended to him/her.
This framework can work well on the more restricted degree programs that have little
variability between the degree plans taken by students, given that there is enough support
for the different degree plans from past students. However, as illustrated in Section 3.2,
the more flexible degree programs have much variability in the degree plans taken by
their students. This makes an exact extraction system like the one above inapplicable for
their students, unless there exists a huge dataset that covers the many different possible sequences with high support.

2.3. Representation Learning

Representation learning has been an invaluable approach in machine learning and artificial intelligence for learning from different types of data such as text and graphs. Objects can be represented in a vector space via local or distributed representations. Under local (or one-hot) representations, each object is represented by a binary vector, of size equal to the total number of objects, where only one of the values in the vector is one and all the others are set to zero. Under distributed representations, each object is represented by a dense or sparse vector, which can come from hand-engineered features that is usually sparse and high-dimensional, or a learned representation, called “embeddings” in a latent space that preserves the relationships between the objects, which is usually low-dimensional and more practical than the former.

A widely used approach for learning object embeddings is Singular Value Decomposition (SVD) (Golub and Reinsch, 1970), which factorizes a given matrix into a low-rank approximation. SVD has been widely used in many areas, e.g., in Information Retrieval to learn latent semantic factors from a document-term matrix (Deerwester et al., 1990) for eg., and in recommender systems to learn the latent factors associated with a user-item rating matrix that uncover the observed ratings (Sarwar et al., 2000; Bell et al., 2007; Patorcik, 2007; Koren, 2008) for eg.).

Recently, neural networks have gained a lot of interest for learning object embeddings in different fields, for their ability to handle more complex relationships than SVD. For instance, neural language models for words, phrases and documents in Natural Language Processing (Huang et al., 2012; Mikolov et al., 2013; Le and Mikolov, 2014; Pennington et al., 2014; Mikolov et al., 2013) are now widely used for different tasks, such as machine translation and sentiment analysis. Similarly, learning embeddings for graphs, such as: DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015) and node2vec (Grover and Leskovec, 2016) were shown to have performed well on different applications, such as: multi-label classification and link prediction. Moreover, learning embeddings for products in e-commerce and music playlists in cloud-based music services have been recently proposed for next basket recommendation (Chen et al., 2012; Grbovic et al., 2015; Wang et al., 2015).

3. Good-course-based Recommendation

3.1. Problem Definition

The current problem definition for course recommendation in the literature (Bendakir and Aimeur, 2006; Lee and Cho, 2011; Elbadrawy and Karypis, 2016) is as follows:

**Definition 1. (Current)** Given a student who has taken at least one term with at least one course, the goal is to recommend a set of courses for that student which they are most likely to register for in their following term.

This definition does not take into account the grade that the student is expected to obtain on the recommended courses. Consequently, the various methods that were previously
developed used all the previous students’ data to learn their models. However, not all students make good course selection choices, in the sense that they may not have the knowledge required for the courses that they have registered for in order to perform well on them, and as a result, students will perform poorly on these courses. This is illustrated in Figure 1 that shows the histogram of student-course grade difference from his/her average previous grade, for the dataset used in our experiments (see Table 1). The figure shows that more than 10% of the student-course grades are a full letter grade poorer, i.e., \( \leq -1 \), than the corresponding students’ previous average grades (here, the letter grading system has 11 letter grades (A, A-, B+, B, B-, C+, C, C-, D+, D, F) that correspond to the numerical grades (4, 3.667, 3.333, 3, 2.667, 2.333, 2, 1.667, 1.333, 1, 0). The course selection choices that lead to the students’ such poor performance would potentially lead to undesired consequences. For instance, that student would have to repeat the same course when he/she is better prepared for it in order to achieve a higher grade on it. Another possible consequence is that the student would be forced to take more credits than the required ones to improve their GPA to what they would prefer. Both of these consequences would increase the student’s frustration and time-to-graduate.

Hence, in order to better assist the students during course selection in each term, we propose the following course recommendation problem statement:

**Definition 2. (Proposed)** Given a student who has taken at least one term with at least one course, the goal is to recommend a set of courses for that student which they are most likely to register for in their following term and on which their expected grades will maintain or improve their overall GPA.

Here, we assume, without loss of generality, that the student will spend the same effort on the recommended courses as they did on previous courses. Hence, as a result of Definition 2, we need to modify the way we use the previous students’ data in order to learn a better course recommendation method. As such, for a student \( s \), we define a course \( c \) to be **a good subsequent course** if the student’s grade in it is close to or higher than
their average previous grade, i.e., if

\[ g_{s,c} \geq \mu_s + \tau, \tag{1} \]

where \( g_{s,c} \) denotes s’s grade on course \( c \), \( \tau \) denotes a small pre-defined (positive or negative) offset value, and \( \mu_s \) denotes s’s average previous grade prior to taking \( c \). We define a course \( c \) to be a bad subsequent course for \( s \) if the above condition is not met. The goal of Definition \( 2 \) is thus to recommend to each student a set of good courses only.

For extracting good and bad subsequent courses from the data, we experimented with the parameter \( \tau \) in Eq. \( 1 \) with the values \( \{-0.333, 0.0, 0.333\} \), which allows a course to be considered good if the student’s grade on it was higher than his/her average previous grade by \(-0.333\), \(0\), and \(0.333\), respectively, when using the \(0-4\) grading system.\(^1\) Note that as the value of \( \tau \) increases, the more restricted the model will be in recommending good courses, but also the higher the expected performance of the student on them will be. This also allows us to compare the performance of the different methods when using different good-to-bad course ratios.

3.2. Methods

In higher education institutions, each degree program offers a large set of courses, which can be divided into core and elective courses. The core courses are required to be taken by all students, while the elective courses include a wide range of selection that students can take a subset from based on their interests. There is usually a sample four- or five-year plan suggested by the department. At the University of Minnesota, however, students usually follow different sequences from these suggested plans, which we have verified using the following simple experiment. We defined the similarity between two degree plans in terms of the ordering of common courses in both of them as:

\[
\text{sim}(d_1, d_2) = \frac{\sum_{(x,y) \in O} T(t_{1,x} - t_{1,y}, t_{2,x} - t_{2,y})}{|O|}, \tag{2}
\]

where \( O \) denotes the set of pairs of courses that are common between degree plans \( d_1 \) and \( d_2 \), and \( t_{i,x} \) is the time, i.e., term number, that course \( x \) was taken in \( d_i \), e.g., the first term is numbered 1, the second is numbered 2 and so forth. Function \( T(dt_1, dt_2) \) is defined as:

\[
T(dt_1, dt_2) = \begin{cases} 
1, & \text{if } dt_1 = dt_2 = 0 \\
\exp(-\lambda(|dt_1 - dt_2|)), & \text{if } dt_1 \times dt_2 \geq 1 \\
0, & \text{otherwise}.
\end{cases} \tag{3}
\]

where \( \lambda \) is an exponential decay constant.\(^2\) Function \( T \) assigns a value of 1 for pairs of courses taken concurrently, i.e., during the same term, in both plans, and assigns a value of 0 for pairs of courses that are either: (i) taken in reversed order in both plans, or (ii) taken concurrently in one plan and sequentially in the other. For pairs of courses taken in

---

1Note that for positive values of \( \tau \), the condition in Eq. \( 1 \) is modified to be \( g_{s,c} \geq \min(\text{max-grade}, \mu_s + \tau) \), where max-grade is the highest possible grade in the grading system used, e.g. 4.0 in the \(0-4\) grading system.

2In our analysis, we chose a small exponential decay constant \( \lambda = \frac{1}{5} \) for a slow decay effect.
the same order, it assigns a positive value that decays exponentially with $|dt_1 - dt_2|$. Our underlying assumption behind such an approach is that, when courses $x$ and $y$ are taken concurrently or in the same order with similar time difference in both $d_1$ and $d_2$, then we assume that this is a more similar ordering of both courses than when there is a larger time difference in both plans, and that a different ordering of $x$ and $y$ in the plans does not contribute to their similarity score.

Using Eq. 2 we plotted the histogram of the pairwise similarities of the degree plans of students belonging to the Computer Science (CS) major, as shown in Figure 2a. We filtered the courses as described in Section 4.1., and we made sure that for each pair of plans compared, at least 40% of the courses taken are in common between the two (see Figure 2b for the distribution of the percentage of common courses between pairs of students). In addition, since the degree requirements and courses change from year to year at the University of Minnesota, we only considered pairs of students of the same cohort, i.e., those who entered college in the same term. The figure shows that the average pairwise degree similarity is 0.61, which shows that there are different possible sequencing for the same set of courses. Similar trends were observed for other majors as well.

![Figure 2: Distributions of pairwise degree similarities (2a) and percentage of common courses (2b) among CS major students.](image)

Motivated by these observations, we investigated two representation learning approaches in order to learn the different possible sequencing of courses and use the learned course representations for course recommendation. The first approach uses truncated Singular Value Decomposition as the underlying model (Section 3.2.1.), while the second one uses a neural network based log-linear model (Section 3.2.2.).

### 3.2.1. Singular Value Decomposition (SVD)

SVD is a traditional low-rank approximation method that has been used in many fields. It factorizes a given matrix $X$ by finding a solution to $X = U\Sigma V^T$, where the columns of $U$ and $V$ are the left and right singular vectors, respectively, and $\Sigma$ is a diagonal matrix containing the singular values of $X$. The $d$ largest singular values, and corresponding singular vectors from $U$ and $V$, is the rank $d$ approximation of $X$ ($X_d = U_d\Sigma_d V_d^T$). This technique is called truncated SVD.
SVD has been widely used in recommendation systems, where typically a user-item rating matrix is decomposed into the user and item latent factors that uncover the observed ratings in the matrix (Bell et al., 2007; Paterek, 2007; Koren, 2008, for eg.). Since we are interested in learning course sequencing as taken by past students, we exploit SVD on a previous-subsequent co-occurrence frequency matrix, where the entry in the $i$th row and $j$th column represents the frequency between a previously-taken course, up to a certain number of prior terms, and a subsequent course over all students. This allows us to learn two different latent representations for each course, namely, the previous and subsequent representations when they are treated as a previous and subsequent course, respectively. Note that the previous courses taken by a student together implicitly construct the student’s profile.

Following our proposed definition of good and bad courses, we formed two different matrices, as follows. Let $n_{ij}^+$ and $n_{ij}^-$ be the number of students who have taken course $i$ before course $j$, where course $j$ is considered a good course for the first group and a bad course for the second one, respectively. The two matrices are:

1. $F^+$: where $F_{ij}^+ = n_{ij}^+$.
2. $F^{+-}$: where $F_{ij}^{+-} = n_{ij}^+ - n_{ij}^-$.

We scaled the rows of each matrix to $L1$ norm and then applied truncated SVD on them. We will append a “+” or a “+-” sign based on which frequency matrix they use. For instance, the SVD model that is applied on the $F^+$ matrix will be denoted as $\text{SVD}(+)$. The course embeddings are then given by $U_d \sqrt{\Sigma_d}$ and $V_d \sqrt{\Sigma_d}$ for the previous and subsequent courses, respectively.

3.2.2. Course2vec

The above SVD model works on pairwise, one-to-one relationships between previous and subsequent courses. We also model course sequencing using a many-to-one relationship, which is motivated by the recent word2vec Continuous Bag-Of-Word (CBOW) model that was proposed for learning distributed representations for words (Mikolov et al., 2013). The CBOW model works on sequences of individual words in a given text, where a set of nearby (context) words (i.e., words within a pre-defined window size) are used to predict the target word. In our case, the sequences would be the ordered terms taken by each student, where each term contains a set of courses, and the previous set of courses would be used to predict future courses for each student.

Model Architecture We formulate the problem as a maximum likelihood estimation problems. Let $T_i = \{c_1, \ldots, c_n\}$ be a set of courses taken in some term $i$. A sequence $Q_s = \langle T_1, \ldots, T_m \rangle$ is an ordered list of $m$ terms as taken by some student $s$, where each term can contain one or more courses. Let $W \in \mathbb{R}^{|C| \times d}$ be the courses representations when they are treated as previous courses, and let $W' \in \mathbb{R}^{d \times |C|}$ be their representations when they are treated as “subsequent” courses, where $|C|$ is the number of courses and $d$ is the number of dimensions in the embedding space. We define the probability of

---

3We also tried TF-IDF weighting scheme and it gave similar performance.
observing a future course $c_t$ given a set of previously-taken courses $c_1, \ldots, c_k$ using the softmax function, which is a log-linear classification model, i.e.,

$$Pr(c_t | c_1, \ldots, c_k) = y_t = \frac{\exp(w'_{c_t} h)}{\sum_j \exp(w'_{c_j} h)},$$

(4)

where $h$ denotes the aggregated vector of the representations of the previous courses, where we use the average pooling for aggregation, i.e.,

$$h = \frac{1}{k} W^T (x_1 + x_2 + \cdots + x_k),$$

(5)

where $x_i$ is a one-hot encoded vector of size $|C|$ that has 1 in the $c_i$'s position and 0 otherwise. The architecture of the above framework can be represented by a neural network with one hidden layer, which is shown in Figure 3. Note that one may consider more complex neural network architectures, e.g., non-linear aggregation operation, deep networks with many hidden layers, or recurrent neural networks. This is left for future work.

Similar to the two matrices proposed for the SVD model that follow Definition 2, we propose the two following Course2vec-based models:

1. **Course2vec(+)**. This model maximizes the log-likelihood of observing only the good subsequent courses that are taken by student $s$ in some term given his/her previously-taken set of courses. The objective function of Course2vec(+) is thus as follows:

$$\maximize_{W, W'} \sum_{s \in S} \sum_{T_i \in Q_s} \left( \log Pr(G_{s,i}|P_{s,i}) \right),$$

(6)

where: $S$ is the set of students, $G_{s,i}$ is the set of good courses taken by student $s$ at term $i$, and $P_{s,i}$ is the set of courses taken by student $s$ prior to term $i$, up to a
2. **Course2vec(+-)**. This model maximizes the log-likelihood of observing good courses and minimizes the log-likelihood of observing bad courses given the set of previously-taken courses, up to a pre-defined number of prior terms. The objective function of Course2vec(+-) is thus:

\[
\text{maximize } \sum_{s \in S} \sum_{T \in Q_s} \left( \log Pr(G_{s,i}|P_{s,i}) - \log Pr(B_{s,i}|P_{s,i}) \right), \tag{7}
\]

where: \(B_{s,i}\) is the set of bad courses taken by student \(s\) at term \(i\), and the rest of the terms are as defined in Eq. 6.

**Model Optimization.** In order to solve the objective functions of the above Course2vec-based models (Eqs. 6 and 7), we make a standard conditional independence assumption. We factorize the likelihood by assuming that the likelihood of observing a good or bad course is independent of observing any other course taken in the same term, i.e.,

\[
Pr(G_{s,i}|P_{s,i}) = \prod_{c \in G_{s,i}} Pr(c|P_{s,i}), \tag{8}
\]

and the same for \(Pr(B_{s,i}|P_{s,i})\). In the future, we plan to investigate the effect of relaxing this assumption and taking the concurrency of courses into account.

The objective functions in Eqs. 6 and 7 can be solved using Stochastic Gradient Descent (SGD), by solving for one subsequent course at a time. The computation of gradients in the two equations requires computing Eq. 4 for all courses for the denominator, which requires knowing whether a course is to be considered a good or a bad subsequent course for a given context. However, not all the relationships between every context (previous set of courses) and every subsequent course is known from the data. Hence, for each context, we only update the subsequent course vector when the course is known to be a good or bad subsequent course associated with that context. In the case that some context does not have a sufficient pre-defined number of subsequent courses with known relationships, then we randomly sample a few other courses and treat them as bad courses.

Note that in Course2vec(+-), since a course can be seen as both a good and a bad subsequent course for the same context in the data (for different students), then, in this case, we randomly choose whether to treat that course as good or bad each time according to a uniform distribution that is based on its good and bad frequency in the dataset. In addition, for both Course2vec(+) and Course2vec(+-), if the frequency between a context and a subsequent course is less than a pre-defined threshold, e.g., 20, then we randomly choose whether to update that subsequent course’s vector in the denominator each time it is visited. The code for Course2vec can be found at: [https://goo.gl/uCCqie](https://goo.gl/uCCqie).

---

4Our experiments have shown that using all previous courses improves the performance of the baseline and proposed methods, so we only report the results of using all previous courses in Section 5.
3.3. Recommendation

With the learned “previous” and “subsequent” vectors of courses, course recommendation using the SVD- and Course2vec-based models is done as follows. Given a student $s$ with his/her previous set of courses taken in a pre-defined number of prior terms, $c_1, \ldots, c_k$, who would like to register for his/her following term, we compute the probability $Pr(c_t|c_1, \ldots, c_k)$ for each candidate course $c_t \in C$ according to Eq. 4. We then rank the courses according to their probabilities, and select the top courses as the final recommendations for $s$. Note that since the denominator in Eq. 4 is the same for all candidate courses, the ranking score for course $c_t$ can be simplified to the dot product between $w'_c$, and $h$.

4. Experimental Evaluation

4.1. Dataset Description and Preprocessing

The data used in our experiments was obtained from the University of Minnesota, where it spans a period of 16 years (Fall 2002 to Summer 2017). From that dataset, we extracted the degree programs that have at least 500 graduated students until Fall 2012, which accounted for 23 different majors from different colleges. For each of these degree programs, we extracted all the students who graduated from this program and extracted the 40 most frequent courses taken by the students as well as the courses that belonged to frequent subjects, e.g., CSCI is a subject that belongs to the Computer Science department at the University. A subject is considered to be frequent if the average number of courses that belong to that subject over all students is at least three. We removed any courses that were taken as pass/fail.

Using the above dataset, we split it into train, validation and test sets as follows. All courses taken before Spring 2013 were used for training, courses taken between Spring 2013 and Summer 2014 inclusive were used for validation, and courses taken afterwards (Fall 2014 to Summer 2017 inclusive) were used for test purposes.

At the University of Minnesota, the letter grading system has 11 letter grades (A, A-, B+, B, B-, C+, C, C-, D+, D, F) that correspond to the numerical grades (4, 3.667, 3.333, 3, 2.667, 2.333, 2, 1.667, 1.333, 1, 0). For each (context, subsequent) pair in the training, validation, and test set, where the context represents the previously-taken set of courses by a student, the context contained only the courses taken by the student with grades higher than the “D+” letter grade. The statistics of the 23 degree programs are shown in Table 1.

4.2. Baseline and Competing Methods

In our experiments, we compared the performance of the SVD- and Course2vec-based models against the previously developed course recommendation methods, as well as against variants of the SVD- and Course2vec-based models when they follow Definition 1.

1. grp-pop(++) This is the “Group Popularity Ranking” method that was proposed by Elbadrawy and Karypis (2016). It ranks the courses based on how frequently they were taken by students of the same major and academic level as the target student.Though this is a simple ranking method, it was shown to be among the best
Table 1: Dataset statistics.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>#students</th>
<th>#courses</th>
<th>#grades</th>
<th>%good-courses (when $\tau =$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCT</td>
<td>861</td>
<td>54</td>
<td>17,614</td>
<td>18</td>
</tr>
<tr>
<td>AEM</td>
<td>605</td>
<td>60</td>
<td>13,280</td>
<td>29</td>
</tr>
<tr>
<td>BIOL</td>
<td>1,369</td>
<td>58</td>
<td>21,590</td>
<td>28</td>
</tr>
<tr>
<td>BSE</td>
<td>730</td>
<td>52</td>
<td>11,389</td>
<td>29</td>
</tr>
<tr>
<td>BMIE</td>
<td>700</td>
<td>56</td>
<td>13,808</td>
<td>28</td>
</tr>
<tr>
<td>CHEN</td>
<td>887</td>
<td>71</td>
<td>19,219</td>
<td>19</td>
</tr>
<tr>
<td>CHEM</td>
<td>789</td>
<td>66</td>
<td>13,814</td>
<td>25</td>
</tr>
<tr>
<td>CIVE</td>
<td>814</td>
<td>61</td>
<td>15,992</td>
<td>32</td>
</tr>
<tr>
<td>COMM</td>
<td>1,361</td>
<td>55</td>
<td>17,135</td>
<td>41</td>
</tr>
<tr>
<td>CSE</td>
<td>1,114</td>
<td>71</td>
<td>17,520</td>
<td>29</td>
</tr>
<tr>
<td>ECE</td>
<td>926</td>
<td>73</td>
<td>18,781</td>
<td>30</td>
</tr>
<tr>
<td>ELEM</td>
<td>787</td>
<td>45</td>
<td>13,303</td>
<td>46</td>
</tr>
<tr>
<td>ENGL</td>
<td>1,219</td>
<td>86</td>
<td>13,451</td>
<td>35</td>
</tr>
<tr>
<td>FIN</td>
<td>1,248</td>
<td>55</td>
<td>24,150</td>
<td>18</td>
</tr>
<tr>
<td>GCD</td>
<td>712</td>
<td>57</td>
<td>11,726</td>
<td>28</td>
</tr>
<tr>
<td>JOUR</td>
<td>2,322</td>
<td>78</td>
<td>31,549</td>
<td>35</td>
</tr>
<tr>
<td>KIN</td>
<td>1,197</td>
<td>120</td>
<td>26,451</td>
<td>44</td>
</tr>
<tr>
<td>MKTG</td>
<td>1,297</td>
<td>50</td>
<td>22,084</td>
<td>26</td>
</tr>
<tr>
<td>MECH</td>
<td>1,650</td>
<td>65</td>
<td>31,608</td>
<td>31</td>
</tr>
<tr>
<td>NURS</td>
<td>936</td>
<td>59</td>
<td>18,239</td>
<td>30</td>
</tr>
<tr>
<td>NUTR</td>
<td>626</td>
<td>50</td>
<td>12,400</td>
<td>40</td>
</tr>
<tr>
<td>POL</td>
<td>1,339</td>
<td>89</td>
<td>13,904</td>
<td>34</td>
</tr>
<tr>
<td>PSY</td>
<td>1,980</td>
<td>67</td>
<td>25,299</td>
<td>38</td>
</tr>
</tbody>
</table>

%good-courses denotes the percentage of good courses in the training set when using the corresponding $\tau$ values (see Eq. 1).

performing methods proposed by the authors. This is due to the domain restrictions, where each degree program offers a specific set of required and elective courses for the students to choose a subset from, and a pre-requisite structure exists among most of these courses. Moreover, these courses follow an academic-level-based numbering approach that students usually follow, e.g., courses at the University of Minnesota are numbered as 1XXX–5XXX, where the leftmost number denotes the suggested academic student level (freshman – graduating senior). Note that this method follows Definition 1.

2. $\text{grp-pop}(+-)$ and $\text{grp-pop}(+)$. We modified the above Group Popularity Ranking method in order to follow Definition 2. For each course $c$, let $n^+_c$ and $n^-_c$ be the number of students that have the same major and academic level as the target student $s$, where $c$ was considered a good subsequent course for the first group and a bad one for the second group. Then, the ranking of each course in $\text{grp-pop}(+)$ is computed as: $n^+_c$, while in $\text{grp-pop}(+-)$, it is computed as: $n^+_c - n^-_c$.

3. $\text{SVD}(++)$. This is the SVD-based model when applied on the previous-subsequent co-occurrence frequency matrix $F^{++}$, where $F^{++}_{ij} = n^+_i + n^-_i$, so it treats both good and bad subsequent courses as good ones (see Section 3.2.1 for the definition of $n^+_i$ and $n^-_i$). Similar to SVD(+) and SVD(+-), we applied the same weighting schemes on the $F^{++}$ matrix.

4. $\text{Course2vec}(++)$. This is the Course2vec-based model that maximizes the log-likelihood of observing all courses taken by student $s$ in some term given the set of
previously-taken courses, regardless of being good or bad subsequent courses. The objective function for this model is thus as follows:

$$\max W, W' \sum_{s \in S} \sum_{T_i \in Q_s} \left( \log Pr(G_{s,i} | P_{s,i}) + \log Pr(B_{s,i} | P_{s,i}) \right).$$

Note that we append a (++), (+), or (+-) suffix to each method based on how they treat the subsequent courses during learning. A (++)-based method treats both good and bad courses as good ones (following Definition 1). A (+)-based method treats good courses as good ones and ignores the bad ones, while a (+-) based method treats good courses as good ones and bad courses as bad ones, both following Definition 2 in different ways.

4.3. Evaluation Methodology and Metrics

We evaluated the performance of the different methods for course recommendation using three different metrics, namely Recall(good), Recall(bad), and Recall(diff). The first, Recall(good), measures the fraction of the actual good courses that are retrieved. The second, Recall(bad), measures the fraction of the actual bad courses that are retrieved. The third, Recall(diff), measures the overall performance of the recommendation method in ranking the good courses higher than the bad ones.

The first two metrics are computed as the average of the student-term-specific corresponding recalls. In particular, for a student $s$ and a target term $t$, the first two recall metrics for that $(s, t)$ tuple are computed as follows:

1. Recall(good)$(s, t) = \frac{|G_{s,n(s,t)}|}{n^g_{(s,t)}}$.
2. Recall(bad)$(s, t) = \frac{|B_{s,n(s,t)}|}{n^b_{(s,t)}}$.

$G_{s,n(s,t)}$ and $B_{s,n(s,t)}$ denote the set of good and bad courses, respectively, that were taken by $s$ in $t$ and exist in his/her list of $n(s,t)$ recommended courses, $n(s,t)$ is the actual number of courses taken by $s$ in $t$, and $n^g_{(s,t)}$ and $n^b_{(s,t)}$ are the actual number of good and bad courses taken by $s$ in $t$, respectively. Note that each of Recall(good) and Recall(bad) is only computed for an $(s, t)$ tuple if that tuple contains good or bad courses, respectively, otherwise it is ignored. Since our goal is to recommend good courses only, we consider a method to perform well when it achieves a high Recall(good) and a low Recall(bad).

Recall(diff) is computed as the difference between Recall(good) and Recall(bad), i.e.,

3. Recall(diff) = Recall(good) - Recall(bad).

Recall(diff) is thus a signed measure that assesses both the degree and direction to which a recommendation method is able to rank the actual good courses higher than the bad ones in its recommended list of courses for each student, so the higher the Recall(diff) value, the better the recommendation method is.

Note that, for each $(s, t)$ tuple, the recommended list of courses using any method are selected from the list of courses that are being offered at term $t$ only, and that were not
Table 2: Prediction performance of the different methods.

<table>
<thead>
<tr>
<th>τ = -0.333</th>
<th>%good courses</th>
<th>%good terms</th>
<th>%bad terms</th>
<th>grp-pop (+) (+) (+-)</th>
<th>SVD (+) (+) (+-)</th>
<th>Course2vec (+) (+) (+-)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall(good)</td>
<td>0.423 0.421 0.399</td>
<td>0.475 0.481 0.431</td>
<td>0.487 0.471 0.420</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall(bad)</td>
<td>0.437 0.377 0.270</td>
<td>0.548 0.446 0.280</td>
<td>0.530 0.451 0.312</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall(diff)</td>
<td>-0.014 0.044 0.129</td>
<td>-0.073 0.035 0.151</td>
<td>-0.044 0.020 0.107</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>τ = 0.0</td>
<td>0.427 0.433 0.375</td>
<td>0.471 0.483 0.407</td>
<td>0.475 0.475 0.390</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall(good)</td>
<td>0.449 0.361 0.196</td>
<td>0.544 0.412 0.214</td>
<td>0.535 0.438 0.257</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall(bad)</td>
<td>-0.023 0.071 0.179</td>
<td>-0.073 0.070 0.193</td>
<td>-0.060 0.037 0.133</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall(diff)</td>
<td>0.430 0.451 0.335</td>
<td>0.476 0.499 0.352</td>
<td>0.483 0.478 0.380</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>τ = 0.333</td>
<td>0.462 0.358 0.141</td>
<td>0.538 0.387 0.146</td>
<td>0.519 0.415 0.223</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recall(good)</td>
<td>-0.033 0.092 0.194</td>
<td>-0.061 0.111 0.206</td>
<td>-0.036 0.063 0.157</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

τ is defined in Eq.1. %good courses denotes the percentage of good subsequent courses in the test set. %good terms and %bad terms denote the percentages of terms in the test set that contained good and bad subsequent courses, respectively. The underlined entries denote the results with the best performance in terms of Recall(diff).

already taken by $s$ with an associated grade that is either: (i) $\geq C+$, or, (ii) $\geq \mu_s - 1.0$, where $\mu_s$ is the average previous grade achieved by $s$. This filtering technique significantly improved the performance of all the baseline and proposed methods.

4.4. Model Selection

We did an extensive search in the parameter space for model selection. The parameters in the SVD-based models is the number of latent dimensions ($d$). The parameters in the Course2vec-based models are: the number of latent dimensions ($d$), and the minimum number of subsequent courses ($samples$), in the denominator of Eq.4 that are used during the SGD process of learning the model. We experimented with the parameter $d$ in the range $[10 - 30]$ with a step of 5, and with the minimum number of $samples$ in the range $[3 - 9]$ with a step of 2.

The training set was used for learning the distributed representations of the courses, whereas the validation set was used to select the best performing parameters in terms of the highest Recall(diff). The selected parameters were in the range $[20 - 30]$ for $d$ for both the SVD- and Course2vec-based models, and in the range $[3 - 9]$ for $samples$ for Course2vec.

5. Results

We evaluate the effectiveness of the proposed problem statement and methods in order to answer the following questions:

**Q1.** Does learning course sequencing using representation learning lead to better course recommendation?

**Q2.** What are the benefits of differentiating between good and bad subsequent courses?

**Q3.** How do the different methods perform among different cohorts and majors?
5.1. Course Sequencing versus Group Popularity Ranking

Table 2 shows the performance prediction of the proposed and competing methods. Comparing the performance of the SVD- and Course2vec-based models against the grp-pop-based methods, we can see that following the traditional definition for course recommendation, i.e., Definition 1, both SVD(++) and Course2vec(++) outperform grp-pop(++) with $\sim 11\%$ increase in Recall(good). Similarly, following the proposed definition for course recommendation, i.e., Definition 2, we can see that both the SVD- and Course2vec-based models significantly outperform the corresponding grp-pop-based methods in terms of Recall(good), with statistically significant improvements. The group popularity ranking method personalizes rankings based only on the student’s major and academic level, though students of the same major and academic level can take various sets of courses with different sequencing. On the other hand, both the SVD- and Course2vec-based methods model the sequential behavior of courses as taken by past students, which allows them to learn the different possible sequencing of the courses and creates better personalized recommendations based on the previously-taken set of courses.

5.2. Focusing on the Good Courses

By differentiating between good and bad subsequent courses and treating them differently during learning, we can see, from Table 2, that each (+)-based method achieves a Recall(good) that is comparable to or better than that achieved by their corresponding (++)-based method. In addition, the (+)-based methods achieve much better (lower) Recall(bad). For instance, when $\tau = 0.0$, SVD(+) and SVD(+-) achieve 0.412 and 0.214 Recall(bad), respectively, resulting in 24% and 61% improvement over SVD(++), respectively. This is expected, since the (++)-based methods treat both types of subsequent courses equally during their learning, and so they recommend both types in an equal manner. This shows that using Definition 2, i.e., differentiating between good and bad courses, in any course recommendation method is extremely helpful for ranking the good courses higher than the bad ones, which will help the student maintain or improve their overall GPA.

By comparing the (+)- and (+-)-based methods, we see that, the (+-)-based model achieves a worse Recall(good), but a much better Recall(bad). For instance, when $\tau = 0.0$, SVD(+-) achieves a 16% decrease in Recall(good) and a 48% decrease in Recall(bad) over SVD(+). This is expected, since adding the bad subsequent courses gives the models more power to learn not to rank these courses high in the recommended list, but it also adds some noise, since different students with the same or similar previous set of courses can achieve different outcomes on the same courses.

5.3. Performance among Different Cohorts and Majors

We compared the performance of the different (+-)-based methods across different cohorts and majors. Figures 4 and 5 show the prediction performance of the (+)-based methods on a per-major and per-cohort basis, respectively, in terms of Recall(good) and Recall(bad).
As shown in Figures 4c and 4d, the performance of the different methods varies considerably among different majors, which was found to be highly positively correlated with both the average pairwise degree similarity (as computed according to Eq. 2) and the average pairwise percentage of common courses among students of these majors, as shown in Figures 4a and 4b, respectively. These correlation values were found in the range.
Table 3: Prediction performance of the different methods on the top- and bottom-5 majors with respect to their average pairwise degree similarity and percentage of common courses.

<table>
<thead>
<tr>
<th></th>
<th>grp-pop(++)</th>
<th>SVD(++)</th>
<th>Course2vec(++)</th>
<th>grp-pop(++)</th>
<th>SVD(++)</th>
<th>Course2vec(++)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Recall(good)</strong></td>
<td>0.593</td>
<td>0.662</td>
<td>0.591</td>
<td>0.574</td>
<td>0.623</td>
<td>0.636</td>
</tr>
<tr>
<td><strong>Recall(bad)</strong></td>
<td>0.366</td>
<td>0.405</td>
<td>0.436</td>
<td>0.397</td>
<td>0.402</td>
<td>0.484</td>
</tr>
<tr>
<td><strong>Recall(good)</strong></td>
<td>0.235</td>
<td>0.271</td>
<td>0.261</td>
<td>0.258</td>
<td>0.288</td>
<td>0.270</td>
</tr>
<tr>
<td><strong>Recall(bad)</strong></td>
<td>0.173</td>
<td>0.230</td>
<td>0.208</td>
<td>0.178</td>
<td>0.222</td>
<td>0.217</td>
</tr>
</tbody>
</table>

DS stands for Degree Similarity, whereas CC stands for Common Courses. Underlined entries denote the best performing results for each metric.

[0.51, 0.86]. This implies that, as the percentage of common courses and degree similarity between pairs of students decrease, accurate course recommendation becomes more difficult, since there is more variability in the set of courses taken as well as the sequencing of the common courses.

Table 3 shows the prediction performance of the different methods on the five most and least similar majors with respect to their average pairwise degree similarity (DS) and percentage of common courses (CC). In terms of Recall(good), both SVD(++) and Course2vec(++) were able to make good recommendations by learning the different sequencing of courses despite the low average pairwise degree similarity and percentage of common courses, with recall values that are statistically significant over grp-pop(++) in the five least similar majors. On the other hand, in terms of Recall(bad), grp-pop(++) performed the best, followed by Course2vec and then SVD(++). Moreover, even for the five most similar majors in terms of both the degree similarity and percentage of common courses, both SVD(++) and Course2vec(++) significantly outperformed grp-pop(++) in terms of Recall(good), while SVD(++) achieved slightly worse Recall(bad) than grp-pop(++) and Course2vec(++) had the worst performance.

Figure 5 shows the performance of the different methods on a per-cohort basis. In terms of Recall(good), Course2vec(++) performs the best for recommending good courses for both sophomores and juniors, whereas SVD(++) outperforms the other methods for the three remaining cohorts. On the other hand, in terms of Recall(bad), SVD(++) outperforms the other methods for the three earliest cohorts, i.e., freshmen, sophomores and juniors, while grp-pop(++) performs the best for seniors and graduating seniors.

6. Conclusion

In this paper, we proposed an improved definition for the course recommendation problem such that the goal would be to recommend “good” courses on which the student’s expected grades will maintain or improve his/her GPA, which can lead to better student satisfaction and better graduation times. In addition, we introduced representation learning to the course recommendation problem by customizing two representation learning approaches so that they learn the good and bad sequencing of courses from the past students’ data. The first approach is based on Singular Value Decomposition, while the other is based on
neural networks.

We conducted an extensive set of experiments on a large dataset obtained from 23 different majors at the University of Minnesota. The results showed that our proposed definition is more helpful for recommending courses on which students are expected to perform well and that align with their degree programs. In addition, the results showed that our proposed methods significantly outperform other competing methods for course recommendation.

In the future, we plan to investigate the synergy and competition among concurrently-taken courses for recommending sets of courses instead of individual ones.

ACKNOWLEDGEMENT

This work was supported in part by NSF (IIS-0905220, OCI-1048018, CNS-1162405, IIS-1247632, IIP-1414153, IIS-1447788) and the Digital Technology Center at the University of Minnesota. Access to research and computing facilities was provided by the Digital Technology Center and the Minnesota Supercomputing Institute, http://www.msi.umn.edu.

REFERENCES


