Department of Computer Science
George Mason University Technical Reports

4400 University Drive MS#4A5
Fairfax, VA 22030-4444 USA
http://cs.gmu.edu/ 703-993-1530

Unsupervised Academic Curricula Evaluation Through Topic Modeling

Jean Michel Rouly jrouly@gmu.edu
Huzefa Rangwala rangwala@cs.gmu.edu
Aditya Johri ajohri3@gmu.edu

April 30, 2015

Abstract

Identifying the concepts covered in a university course based on a high level description is a necessary step in the evaluation of a university's program of study. To this end, data describing university courses is readily available on the Internet in vast quantities. However, understanding natural language course descriptions requires manual inspection and, often, implicit knowledge of the subject area. Additionally, a holistic approach to curricular evaluation involves analysis of the prerequisite structure within a department, specifically the conceptual overlap between courses in a prerequisite chain. In this work we apply existing topic modeling techniques to sets of course descriptions extracted from publicly available university course catalogs. The inferred topic models correspond to concepts taught in the described courses. The inference process is unsupervised and generates topics without the need for manual inspection. We present an application framework for data ingestion and processing, along with a user-facing web-based application for inferred topic presentation. The software provides tools to view the inferred topics for a university's courses, quickly compare departments by their topic composition, and visually analyze conceptual overlap in departmental prerequisite structures.

1 Introduction

Computer Science education is an increasingly important field of growth at many universities [21, 14]. As departments grow and change, it becomes necessary to automate the comparison and evaluation processes. However, much of the published data about departments is non-standard, natural language text that is not easy to process automatically. There are many parties impacted by the lack of up to date, automatic, and simple to understand information about the characteristics of universities across the country.

Prospective college students and their parents seek out information on college courses to compare curricula in a meaningful way, based on content, in order to find their best fit. A typical approach to this task is an information gathering and subsequent program comparison process duplicated many thousands of times across the population of rising college-going freshmen. Hewner [9] conducted a qualitative study of CS students and found that students in CS need to make a variety of decisions about what courses they take. Invariably, they have limited knowledge when making these decisions. They also often make these decisions based on whether the classes will be enjoyable and assumed that since courses are required, they will have useful content. We believe that a system such as ours can assist students with making more strategic goal-oriented decisions which many of the students want to take. It will also allow them to better see the connection between courses and if possible make decision based on the skillset they want to develop.

Additionally, accrediting bodies (e.g., ABET) typically require a department to cover a given set of standardized topics as a criterion for evaluation [1]. The accreditation process can take up to 18 months to complete [1]; automating the departmental evaluation process would greatly reduce time spent measuring a CS department's coverage of a specific set of areas.

Programs of study at institutions of higher education can be represented as a chain composed of the courses required to complete a degree. These component courses in turn are composed of the topics or concepts they are intended to cover. Evaluation of the courses within a particular program is necessary for the evaluation of an overall academic curriculum. Analyzing the structure of a program's prerequisite chain, for example, requires an understanding of each constituent course and any overlap of covered topics between courses and their prerequisites. Additionally, inter-institutional curricular comparison requires an aggregate evaluation of the courses within each institution's program. However, comparing and evaluating different courses requires expert knowledge in the relevant field. No two courses can be measured for similarity based only on inherent, measurable properties. A domain expert is required to inspect the description of the courses and determine their conceptual overlap.

Automating the information retrieval process to identify core concepts covered in any particular course removes the need for a domain expert. By analyzing course descriptions from a corpus spanning fields and institutions, topic modeling can provide a method to generate a statistical representation of core course concepts. Specifically, unsupervised latent variable models present a method of identifying the core concepts (i.e., topics) covered in a course. This introduces the possibility of applications in automated course and program evaluation methods. The form of topic modeling employed in this work is Latent Dirichlet allocation (LDA) [5].

The overall goal of this work is the development of a system to digest large quantities of university course information, specifically academic course descriptions, and to process and ultimately generate interactive descriptions of the core concepts covered within institutional programs as illustrated by inferred topics. Learned topics will be presented in a web-based application allowing inspection from multiple perspectives.

2 Background on LDA

Topic modeling, a form of latent variable modeling, is an unsupervised machine learning method which attempts to recreate the distribution of so-called "topics" an author used to generate a corpus of documents. The term topic is used to describe a frequency distribution of terms within a vocabulary. In this use, a topic can be understood to represent an academic concept covered within the context of a course. The topics discovered in a corpus can be used to categorize documents and provide structure to an otherwise unknown dataset.

Latent Dirichlet allocation (LDA) is a specific type of topic modeling which assumes that a mixture of multiple topics exist within a single document in some proportion (i.e., were used to generate that document) [5]. LDA assumes a generative process where, for each word in the document, the algorithm selects a distribution over topics, selects a topic, and then selects a vocabulary term [5]. By picking a distribution over topics, multiple possible topics can be blended into a single document. Reversing this generative process is significantly more difficult because the topic distributions are unknown; this is what the "hidden model" or "latent model" refers to.

LDA can best be understood through its generative process. Given the set of distributions as input, generating the corpus topics is a probabilistic process. Taking the variables $\theta_{d,k}$ (topic proportion for topic k in document d), $\beta_{1:k}$ (topic k), $z_{d,n}$ (topic assignment for word n in document d), and $w_{d,n}$ (the n^{th} word in document d), LDA calculates the posterior probability in Equation 2.1 [4].

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D} | w_{1:D}) = \frac{\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}}{w_{1:D}} \qquad (2.1)$$

Using a variety of probabilistic methods to calculate or approximate the denominator (i.e., evidence), LDA results in a usable set of vocabulary frequency distributions or topics. Specifically, Gibbs Sampling, a variety of Bayesian Inference, is used to approximate the LDA posterior probability [5].

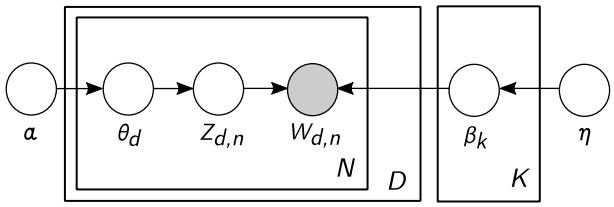

Figure 2.1: LDA graphical diagram adapted from [4].

A graphical "plate" diagram of LDA is given in Figure 2.1, adapted from [5]. Circular nodes represent latent variables while rectangular plates represent duplication. This representation is identical to Equation 2.1, with N the number of words in a document and D the number of documents.

2.1 Related Work

Our research complements other efforts within Computer Science education that are directed towards categorization of content to improve pedagogy, e.g., Hubwieser et al. [10]. We believe that our project contributes both, by identifying content (i.e., topics) being taught across institutions and by identifying gaps and unique contributions. This information can be compared to teacher competencies and used to design assessment and instruments to measure them. Another area in which this work can assist is in identification of concepts and their classification, especially "threshold concepts" [19]. Overall, we believe our data-driven approach complements other qualitative efforts by building on them and by automating some aspects of the research.

Other work has also attempted to extract concept information from course data. Yang et al. employ four distinct techniques to map courses into a conceptual space and then learn prerequisite relationships between similar courses [20]. Two of their conceptual mapping techniques generate latent features, which have the downside of not being human-readable as in LDA. The remaining two techniques generate human-readable topics, but rely either on an outside source (Wikipedia) or simply represent concepts as the vocabulary of the document. The benefit of LDA as an information retrieval tool is its ability to generate pseudo human-readable topics while acting in a fully unsupervised manner on a single, large data set. Our approach naively targets a dataset of fixed universities and customizes web scrapers specifically for their computer science departments. Effland et al. introduces a robust web crawler system to automatically search for, identify, and extract course descriptions from disparate locations on the Internet [8]. Application of similar technology in this work was considered, and would greatly improve the scale of the analyzed data.

University	Course Count
American University (AU)	32
George Mason University (GMU)	145
Kansas State University (KSU)	83
Louisiana State University (LSU)	59
Portland State University (PDX)	190
Rensselaer Polytechnic Institute (RPI)	61
University of South Carolina (SC)	64
Stanford University (Stanford)	69
University of Utah (Utah)	142
University of Tennessee, Knoxville (UTK)	29
ACM Exemplar Courses (EC)	68

Table 3.1: CS program statistics

3 Methods

The structure of this work is threefold. First, a dataset of university course descriptions is generated using publicly available data from a number of North American universities. This process is discussed in Section 3.1. Second, LDA is applied to the collected data and topics are inferred. This step is discussed in Section 3.3. Finally, the collected data and inferred topics are presented in a user-facing web application, described in Section 3.4.

3.1 Data Acquisition

The primary data manipulated in this study are university catalog course descriptions. The current experimental data sources are described in Table 3.1. Note that only Computer Science departments are included in this dataset, in order to simplify the evaluation process and the required number of inferred topics. Simple web scrapers were written using Python and BeautifulSoup to download publicly available descriptions from university catalogs. These will be made publicly available at `http://github.com/jrouly/trajectory`. Descriptions and catalog webpages appear in a vast variety of different formats, structures, and HTML correctness, so a parser was written for each university to acquire the unstructured text. This text was then passed through a cleaning procedure to remove abbreviations and non-English characters, as well as common English stop words. Finally the text was passed through a stemmer to strip morphology from the words and eliminate duplicate terms. At the same time, departmental course prerequisite data is collected from the catalog as well. Prerequisites are limited to courses in the database, meaning that references older courses that are no longer present in the catalog are ignored. Non-specific prerequisite references (e.g., `400-level`) are ignored as well.

The Python scraping framework developed is structured to allow pluggable web scrapers tooled to specific syllabus repositories. There are a number of existing web scrapers in place pointing to different university course catalogs, but the code can be easily extended in the future to grow the data set. Integrated in the scraping framework is a lightweight relational database layer to store both course description data and university metadata, including names, URLs, and prerequisite information. The database layer also stores inferred topics.

3.2 Preliminary Data Exploration

In addition to LDA, other unsupervised machine learning tools can be applied to the same data set. Simple clustering algorithms (e.g., K-Means) when given the same bag of words corpus as input act to identify groupings of similar documents according to their term frequency vector Euclidean distance [16]. Additional, similar clustering algorithms can be applied in a similar manner.

Preliminary exploratory results are promising. We applied K-Means clustering to a sample dataset of course descriptions selected from the George Mason University Computer Science online catalog across multiple semesters. Using course section IDs as ground-truth labels, we clustered the course descriptions. Table 3.2 summarizes the metrics computed on the resulting clustering. Homogeneity represents the "same-ness" of a cluster, or the degree to which each cluster contains only members of a single type. Completeness represents the "spread" of courses across clusters, or the degree to which every member of the same type is assigned to the same cluster. V-Measure is simply the harmonic mean of the prior two metrics. For each metric, higher is better, and they are bounded from 0 to 1. A distributed implementation of K-Means available in the Python toolkit `scikit-learn` was used to perform the clustering.

The high completeness values are promising: this indicates that many of the same course are assigned under the same cluster prototype. The low value of homogeneity is unsurprising given the initialization parameters used: K-Means was initialized to detect only 20 clusters, a far smaller number than the magnitude of distinct course sections available. The number 20 was chosen arbitrarily as a smaller count than the true number of distinct course sections in order to increase cluster size.

Execution time	0.144s
Homogeneity	0.415
Completeness	0.877
V-measure	0.563

Table 3.2: Preliminary clustering metrics

3.3 Topic Modeling

After the exploratory clustering process, we passed cleaned data into a topic modeling framework by exporting from the database layer to the filesystem in a structured "bucket of files" format. The Java MALLET library is used to perform LDA on the course description data. A data pipeline is

constructed using the MALLET API that reads input data, tokenizes it, and trains an LDA topic model. The inferred topics are then exported to a common Comma Seperated Values (CSV) format and read back into the database layer and applied to existing courses. Independent runs of LDA with distinct parameterization are segregated in the database into "Result Sets", allowing sets of inferred topics to sit side by side without interfering. This also allows the presentation of different sets of inferred topics to the end user.

The initialization parameters of MALLET's LDA implementation are summarized in Table 3.3. Experiments were run varying parameters throughout the experimental range, including the MALLET default values. However, as the number of topics increased greatly beyond 750, and as β decreased greatly below 0.0001, the MALLET framework began to encounter instability and errors. Any runs which encountered infinite or "not-a-number" values were immediately discarded. Ultimately 77 result sets were retained for analysis.

3.4 Visualization

An interactive, dynamic user visualization of course descriptions and topics has been prototyped. The visualization is a web-based application built upon the Python Flask library. The tool interfaces with the same database layer used by the rest of the application framework to provide an aesthetically pleasing user-facing interface with several primary modules. By default, the web application presents the user with a high level "dashboard" overview of the dataset and available result sets, where a result set is the set of topics inferred after a single run of the LDA module with a distinct initialization set. After selecting a result set, the user can browse the data by course, department, university, or inferred topics. The remainder of this section describes implementation details of the tool's primary features.

3.4.1 Explore Courses

When presented with the complete university dataset, the user may interactively search for a university. Once a university is selected, the user may search through its registered departments and select a course of interest. On selecting a course, the text of its description is displayed in both its original format and its cleaned, stemmed format. Additionally, the course's inferred topics are listed in order of their proportion within the course description. These topics are expandable, and upon interaction present the user with the list of other known courses with that inferred topic. Any topics appearing in a document with proportion under 15% are automatically hidden from the user. This design choice was implemented to reduce visual clutter from inferred topics with low relevance to the class.

3.4.2 Prerequisite Chain Analysis

In addition to the course-specific data, an interactive, collapsible tree visualization of the course' prerequisite chain is displayed. The recursively generated tree visualization provides a high level view of the course's position within a department. Figure 3.1 details the prerequisite tree visualization. Above this visualization is the prerequisite chain conceptual analysis tool. This tool automatically provides a view of any registered prerequisite courses along with their inferred topics. Any shared topics between the prerequisites and the selected course are highlighted to indicate conceptual overlap.

Observe the course CS 310 "Data Structures" in the upper middle of the prerequisite tree. Table 3.4 details its topics, along with the topics of its prerequisite courses CS 105 "Computer Ethics and Society" and CS 211 "Object-Oriented Programming". Clearly there is a significant area of overlap between a data structures course and an introductory course in object oriented programming, as illustrated by the two italicized overlapping topics. Specifically, the overlap is in the areas of algorithms and data structures as well as object oriented programming. The ethics course, however, does not share any conceptual overlap with CS 310. Inspecting the context of these courses, however, reveals that CS 105 is a common freshman requirement at George Mason, and many upper level classes depend on its completion. It is intended as a baseline of student maturity rather than a prerequisite because of the concepts it introduces.

3.4.3 Compare Departments

The user may also compare two university departments. Topics inferred from every course in the department are collected and displayed side by side. Topics unique to each department are displayed separately from the intersection set of common topics. Similarity metrics describing the relationship between the two departments are defined as well — Jaccard index, cosine similarity, and Euclidean distance. Defined as $\frac{|A \cap B|}{|A \cup B|}$, the Jaccard index is based on the number of items unique to and shared between each set [11]. The remaining metrics are based on a "topic-vector" representation of each department. The topic-vector is a binary vector where each bit indicates whether a particular topic was inferred for the given department. Features are unweighted and the topic-vector indicates only whether a topic is present in a department's topic set, and not its frequency of occurrence. Interpreting this vector representation geometrically, the Euclidean and cosine distances are calculated.

3.4.4 Evaluate Department

The tool also allows users to easily evaluate university departments against third party benchmarks. The Association of Computing Machinery (ACM) maintains an annual writeup of guidelines for undergraduate computer science education [2]. These guidelines include two important sections, the ACM Exemplar Courses (EC) and Knowledge Areas. The Knowledge Areas are 18 broad topics within Computer Science as put forth by the ACM. EC include real course descriptions from disparate sources compiled by the ACM and

Parameter	Description	Experimental Range	Default
α	Dirichlet concentration parameter.	[1, Iterations]	Iterations
β	Dirichlet concentration parameter.	[0.0001, 0.5]	0.01
Iterations	The number of LDA iterations.	—	3000
Topics	The number of topics to infer.	[100, 1000]	—

Table 3.3: LDA Initialization Parameters

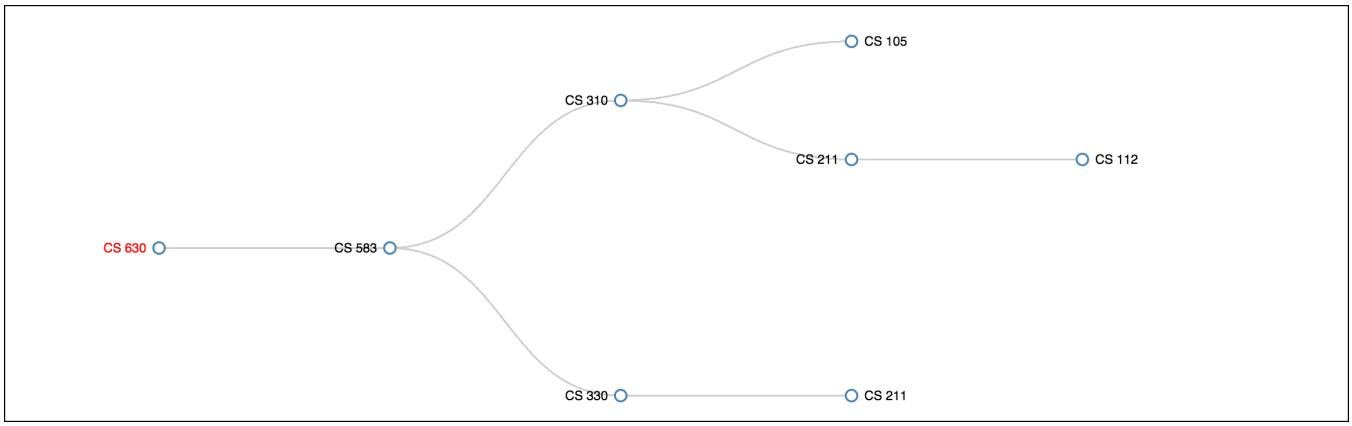

Figure 3.1: Interactive prerequisite tree visualization tool

GMU CS 310 Topics	Proportion
languag, object, program, orient, includ, type, abstract, design, implement, concept	29.946%
program, problem, solv, algorithm, data, structur, comput, introduct, languag, techniqu	21.461%
includ, design, system, topic, comput, introduct, cover, applic, algorithm, techniqu	26.117%
GMU CS 211 Topics [Prerequisite]	**Proportion**
languag, object, program, orient, includ, type, abstract, design, implement, concept	29.433%
program, problem, solv, algorithm, data, structur, comput, introduct, languag, techniqu	26.772%
code, compil, pars, analysi, optim, gener, languag, lexic, techniqu, construct	18.685%
comput, method, theori, basic, princip, includ, topic, model, cover, scientif	16.076%
GMU CS 105 Topics [Prerequisite]	**Proportion**
ethic, comput, issu, profession, social, technolog, privaci, legal, relat, digit	75.964%

Table 3.4: Topics of GMU CS 310 and its prerequisite courses, CS 105 and CS 211. Overlapping topics are italicized.

manually annotated with the Knowledge Areas they cover. These data sets are used to perform primary evaluation. The benchmarks used in this study are the ACM Knowledge Areas.

The web tool automatically evaluates the performance of a university department. The tool checks for conceptual overlap between courses and ACM Knowledge Areas and predicts Knowledge Area labels where overlap exists. Additionally, if the department has been manually annotated with Knowledge Areas, these "ground truth" labels are compared against the predicted labels, and similarity coefficients between the two sets are computed. The label set similarity coefficient is computed in two ways. First, the Jaccard index of the predicted and ground truth labels is calculated. Then, the percent of the ground truth labels included in the prediction set is calculated. The web visualization provides an automatic interface for performing this evaluation process.

4 Case Studies

In the sequel, three case studies are described in depth to exemplify the main features of the tool. First, Section 4.1 discusses the prerequisite analysis features of the tool, and presents a summary of statistics about the universities included in this study. Second, Section 4.2 closely considers two areas within Computer Science and discusses the inferred topics for courses within those areas. Finally, Section 4.3 details the similarities and differences between pairs of university departments and also presents a summary of comparisons between all the universities in this study.

4.1 Case Study: Prerequisite Analysis

To understand how a course fits into a department, its position in the prerequisite chain must be analyzed. Take, for example, a course in mobile application development. We predict that any course of this nature will most likely be an upper level elective with a moderate number of prerequisite courses. Indeed, we see this is the case with George Mason's CS 477, Portland State's CS 410, and Stanford's 231M. However, many of the topics covered in a mobile development course are niche topics, and specific to that field. Therefore, it might not be the case that any particular lower level courses will cover specific, overlapping topics.

Let us consider George Mason University's CS 477 "Mobile Application Development". The course description discusses mobile platforms and the various software design issues specific to mobile platforms. There are two inferred topics for this course. The first topic, with 32% proportion, includes the terms "develop", "platform", and "mobile" at high frequency. The second topic, with only 15% proportion, is more generic and includes terms like "system", "computer", and "topic". The course also has two registered prerequisites: CS 310 "Data Structures" and CS 367 "Computer Systems and Programming". Within the context of the department, these two courses are major prerequisites for any upper level course. As expected, neither of the two prerequisite courses share the mobile-specific topic inferred for CS 477. CS 310, however, does overlap with the generic computer systems topic. We therefore conclude that the CS 477 course registers its prerequisites primarily to ensure a baseline level of maturity and skill among its students, rather than because some necessary concepts are introduced at a lower level and expanded upon at the higher level.

To quantify the typical level of conceptual overlap between prerequisites within a department, we introduce a vector representation of a course, the "weighted topic-vector". Like the unweighted topic-vector representation of a department, the weighted topic-vector is a vector of uniform length corresponding to the total number of inferred topics among all known courses. However, the features of a weighted topic-vector represent the proportion with which the particular topic is represented in the course's description. In this way, the weighted topic-vector takes into account the importance of a topic to a course, rather than simply binary topic membership as in the unweighted topic-vector. By computing the average distance between the weighted topic-vectors of a course and its prerequisites, the level of conceptual overlap for that course can be quantified. Averaging these distance measures over every course in the department that has registered prerequisites results in a measure of average prerequisite conceptual overlap within the department. Table 4.1 summarizes the levels of conceptual overlap for the five universities in the dataset with registered prerequisite trees, where a higher average value of conceptual overlap indicates a closer relationship between courses and their prerequisites. The five universities not included did not have prerequisite relationships in their collected data, and thus this analysis could not be performed.

	Prereq_μ	Prereq_σ
GMU	0.324	0.211
AU	0.278	0.134
KSU	0.273	0.213
Utah	0.257	0.249
UTK	0.201	0.256

Table 4.1: Level of conceptual overlap between courses and their prerequisites in five universities. Prereq_μ is the average level of conceptual overlap between prerequisites, Prereq_σ is the standard deviation.

4.2 Case Study: Topics In Computer Science

4.2.1 Ethics In Computer Science

Computer Science is a wide ranging field, with a number of disparate subfields — according to the ACM, there are 18 distinct Knowledge Areas [2]. At any given university, the Computer Science department will, ideally, cover all or most of these areas. One particular area covered by most universities in this study is "Social Issues and Professional Practice". A clear example of a course within this Knowledge Area is any course in computing and ethics. Take, for example, Kansas State's CIS 415 "Ethics and Computing Technology." The description is brief and to the point, focusing on computing ethics within a professional context. The only topic inferred for this course, at 64% proportion, includes the terms "ethics", "computer", "profession", "issue", and "social" at high frequency. Searching for other courses that teach to the same topic yields 30 courses across nine universities (every university in the study except for Rensselaer Polytechnic). These courses have titles like "Computer Ethics and Society" (GMU CS 105) and "Ethics in Computing" (LSU CSC 1200). This demonstrates the ability of our tool to not only automatically infer relevant concepts from a course description, but to match related courses across universities.

4.2.2 Artificial Intelligence

Another major topic within computer science is the study of Artificial Intelligence (AI). AI itself contains a great number of subfields and areas of specialization, but it is defined in ACM Knowledge Area "Intelligent Systems" as "the study of solutions for problems that are difficult or impractical to solve with traditional methods" [2]. Consider Portland State's CS 441 "Artificial Intelligence." Its course description is highly typical of AI courses included in this study: brief and to the point, it lists a number of subfields within AI that will be touched upon in the course. The only topic inferred for this course, at 86% proportion, includes the terms "intelligence", "knowledge", "artificial", and "agent" at high frequencies. Every other university in this study also contains at least one course which teaches to this topic in some proportion. George Mason's CS 480 "Introduction to

Artificial Intelligence", Louisiana State's CSC 4444 "Artificial Intelligence" and University of Utah's CS 6380 "Multi-Agent Systems" to name a few. Indeed, many of these courses share very similar descriptions.

However, one of the most unique examples of a course with this topic is the University of Tennessee's COSC 420 "Biologically-Inspired Computation." The title alone might not suggest that this is an AI based class, and indeed it is not entirely AI. The course description talks about swarm intelligences, multi-agent systems, and other biomimetic computational systems. While the common AI topic discussed prior is inferred at 42%, two other topics, one with terms from biology and one with terms from neural networks, a subfield of AI, appear at proportions 24% and 17% respectively. UTK's COSC 420 is an excellent example of LDA's ability to infer many possibly unrelated topics in a mixture within a document in order to represent the full nature of the text.

4.3 Case Study: Comparing Departments

4.3.1 George Mason vs Stanford

While institutions generally cover a wide range of topics, there are always certain topics that cannot be or are not taught. Take for example George Mason University and Stanford University. The two computer science departments share a lot in common. Of 68 total topics covered between the two, 41 are common to both while only 14 are unique to Stanford and 13 unique to George Mason. The shared topics contain terms that directly relate to similar courses at each school. For example "secure" and "network" are terms in a topic taught in network security classes at both schools. However, a topic containing "linux", "unix", "lab" appears unique to Stanford, covered by CS 1U "Practical Unix", a course that does not exist at George Mason. Similarly, a topic containing "parallel" and "algorithm" appears unique to George Mason. The course GMU CS 683 "Parallel Algorithms" teaches to this topic, with no analogous course at Stanford.

By analyzing the set of unique and shared topics within two departments, simple coefficients can be computed to quantify the degree of similarity. The Jaccard coefficient is computed as the most basic representation of departmental similarity, along with Euclidean and cosine similarity metrics discussed in Section 3.4.3. A pairwise comparison of the universities in this study is presented in Figure 4.1. In this figure, darker shades of blue represent a higher degree of similarity. Also see Table 4.2 for the specific computed similarity values.

4.3.2 George Mason vs ACM EC

The EC provide a wide view of different courses from institutions around the world. If we assume the EC to compose a single, hypothetical university's CS department, we can compare existing departments against it in the same fashion as was used in Section 4.3.1. Take George Mason University, for example, to compare against the EC. Both departments share a great deal in common, with 38 overlapping topics. There are additionally not that many topics unique to the two departments, with 14 unique to the EC and 18 unique to GMU. A number of interesting artifacts appear, however, upon closer inspection.

Two topics unique to the EC include the terms "moral" and "religious", one each. Inspecting these topics reveals two members of the EC with the titles "Ethics & the Information Age" and "Technology, Ethics, and Global Society". Both of these courses teach ethics in computation, but from a vastly different perspective than the ethics course at George Mason. These two EC ethics courses include professional ethics, but also moral, religious, and social philosophies of ethical behavior, while the George Mason ethics courses (e.g., CS 105 "Computer Ethics and Society") only discuss professional ethics. This is an important yet subtle distinction to make, which might be overlooked by simply considering course titles.

Another unique topic to the EC includes the terms "design", "circuit", and "digital". The courses which teach to this topic, across institutions, are primarily digital design and logic courses. Inspecting the George Mason CS curriculum reveals that an analogous course is required within the CS degree, ECE 301 "Digital Electronics", but is not contained within the CS department, unlike some other universities. This particular artifact appears because the scope of our data sets include only CS departments. We make the assumption that necessary courses to the CS program at an institution will fall under the heading of Computer Science, which in this case is not true.

Note that included in Figure 4.1 are the ACM EC. Each course from the EC was entered into the dataset along with every other university course, and had topics inferred by LDA in the same manner. In this way the EC act as control courses with a ground truth label set. High similarity to the EC indicates a high degree of compliance with ACM standards.

5 Discussion

The results of this study are promising. Exploratory K-Means clustering resulted in clusters with a high level of completeness. Even superficial manual analysis of clusters indicated that the collected data were being appropriately grouped. Similar findings were encountered after the application of LDA topic modeling.

Inferred topics generally fall into one of two categories for each course. The first category of topic includes relevant terms and keywords found in the course description. A highly specialized course in a particular subfield might contain a set of these topics that relate to keywords from within the specific domain of the course. The second category of topic includes more generic words common to a large number of courses. Topics in this category often relate to concepts common across courses, e.g., student research

	ACM EC	AU	GMU	KSU	LSU	PDX	RPI	SC	Stanford	Utah	UTK
ACM EC	1.000	0.420	0.433	0.426	0.407	0.386	0.433	0.443	0.366	0.427	0.373
AU	0.420	1.000	0.339	0.480	0.522	0.342	0.431	0.500	0.377	0.382	0.462
GMU	0.433	0.339	1.000	0.500	0.532	0.460	0.463	0.471	0.603	0.545	0.439
KSU	0.426	0.480	0.500	1.000	0.623	0.494	0.561	0.655	0.493	0.528	0.587
LSU	0.407	0.522	0.532	0.623	1.000	0.462	0.491	0.554	0.524	0.514	0.500
PDX	0.386	0.342	0.460	0.494	0.462	1.000	0.427	0.469	0.488	0.586	0.347
RPI	0.433	0.431	0.463	0.561	0.491	0.427	1.000	0.525	0.547	0.535	0.532
SC	0.443	0.500	0.471	0.655	0.554	0.469	0.525	1.000	0.443	0.521	0.480
Stanford	0.366	0.377	0.603	0.493	0.524	0.488	0.547	0.443	1.000	0.600	0.407
Utah	0.427	0.382	0.545	0.528	0.514	0.586	0.535	0.521	0.600	1.000	0.409
UTK	0.373	0.462	0.439	0.587	0.500	0.347	0.532	0.480	0.407	0.409	1.000

Table 4.2: Pairwise similarity of CS departments computed as Jaccard index of department topic sets.

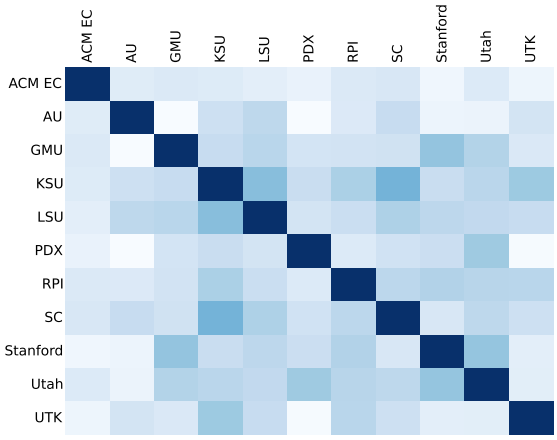

Figure 4.1: Pairwise similarity of CS departments using the Jaccard index. Darker shades indicate higher similarity. ACM EC refers to the ACM Exemplar Courses (EC). See Table 4.2 for the specific recorded values.

or exam and project information.

Take, for example, a computer science course in ethics. At George Mason University the senior level Computer Science ethics course is CS 306. The primary topic inferred for this course in a given run of LDA might look like "ethic, comput, issu, profession, social, impact, privaci, digit, context, technolog." Additionally, the prerequisite course to CS 306, CS 105, shares this same topic. While the topic itself is a frequency distribution over vocabulary, and does not quantifiably evaluate to an "ethics" topic, manual inspection clearly shows that this topic involves ethocs and social issues as they relate to technology professionals.

Our confidence in the applicability of LDA as a course content inference system stems from the appearance of the same or related topics within the same or related courses. Inspecting the same course at different institutions results in the same topics being inferred at each institution. Inspecting prerequisite courses within the same institution illustrates the relationship between the courses by highlighting conceptual overlap, i.e., the appearance of the same topic in both course and prerequisite. As mentioned in Section 3.4.4, third party course descriptions act as an additional evaluative metric for this approach. A high level of consistency is indicated by the same topics being inferred for the same course at multiple institutions, related courses within the same institution, and the third party benchmark course.

5.1 Limitations

We recognize that one of the limitations of our work is that we are focused solely on the formal learning of students [6]. This is an artifact of the data used in the study. We believe that the techniques we use can also be applied to better understand learning in other settings if useful data exists. Additionally, our methodology operates under the assumption that course description data is accurate, up to date, and descriptive. In truth, this is not always the case. Oftentimes course descriptions do not fully describe the actual content of what is taught in a course, or the course description might only apply to some sections of a given course. It has even been observed that some course descriptions are merely

held as filler text until a later date, and do not provide any useful information about the course. However, while these circumstances lie outside of our control, we take measures to prevent invalid course descriptions from entering the dataset. The cleaning process described in Section 3.1 removes descriptions that are unlikely to be valid based on length and vocabulary size. It should be noted that this is a general purpose technique which can also be applied to intake other course related information, such as syllabi and assignment data instead of course descriptions.

Another limitation of our approach is an inability to meaningfully summarize or categorize inferred topics. While the raw topics are used internally for comparison, they do not provide an ideal interface for the end user. This limitation stems from our use of LDA as the primary topic inference tool. We propose a solution in the following Section 6.

6 Future Work

One of the major benefits and weaknesses of LDA is its unsupervised nature. Beneficially, it allows for the extraction of information from an entirely unknown dataset. In this context, this flexibility allows its application to any number of diverse academic departments. However, the main drawback of this characteristic is the lack of categorical information for the inferred topics. While a topic can be understood via manual inspection of its terms, LDA offers no single comprehensive label to summarize it. A possible solution to this problem might involve a meta-analysis of inferred topics. Each topic could be classified as one of a number of learning outcomes based on its composition and weighting in a description. Mapping the learned topics onto a standardized framework of learning outcomes [15] would allow for immediate integration of the extracted course concepts into existing academic evaluative frameworks based on learning outcome literature.

As educators who often have deficient resources to improve their pedagogy [7] look towards online or virtual mechanisms to support them, the system we have designed can be very useful. Based on ideas discussed by Brown and Kölling [7], one potential we see for future work is the integration of our system with an existing virtual community of CS educators, or the creation of community features around the system we have designed. For instance, we can make it easier for educators to share or request resources from others or to learn more about why certain content is or is not covered in specific courses. As more educators provide data to the system, the quality of results will benefit as well.

We also foresee that in the future we will be able to combine course related data with specific course assignments (through learning-management system (LMS) data), thereby providing a better picture of the kinds of experiences students can hope to receive in any given course. This combination of data will also allow a better examination of the effects of different teaching strategies on students' learning [18, 13, 3]. For instance, we will be able to better understand the role of pedagogical techniques, such as problem-based and service-learning, on student outcomes. By aligning with LMS data, we will also be able to learn more about how students perform on different assignments related to a specific competency or course content. In the future we also plan to combine this data with data about student demographics and that can help provide a better picture of performance across gender and race [12, 17]. This can be useful in designing more supportive pedagogical elements.

7 Conclusion

Programs of study in higher education differ widely between departments and universities. Because of these discrepancies, program evaluation methodologies are employed with the goal of understanding the contents of a program of study within a standardized framework. However, this process generally requires manual inspection by a domain expert to extract information from large quantities of course descriptions. Automating the digestion and processing of these descriptions will greatly reduce the time and effort required. Probabilistic topic modeling presents a statistical machine learning method to automatically extract the core concepts covered by a course description. Latent Dirichlet allocation (LDA) results in a feasible breakdown of textual descriptions into component concepts. This work has presented a software framework for the ingestion and processing of large volumes of textual course descriptions. Additionally, a web-based visualization tool has been developed to present inferred topics for university programs of study in an accessible format.

References

[1] Accreditation Board for Engineering and Technology. Criteria for accrediting computing programs. Website, 2015.

[2] ACM/IEEE-CS Joint Task Force on Computing Curricula. Computer science curricula 2013. Technical report, ACM Press and IEEE Computer Society Press, December 2013.

[3] Lecia J Barker and Kathy Garvin-Doxas. Making visible the behaviors that influence learning environment: A qualitative exploration of computer science classrooms. *Computer Science Education*, 14(2):119–145, 2004.

[4] David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April 2012.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[6] Jonas Boustedt, Anna Eckerdal, Robert McCartney, Kate Sanders, Lynda Thomas, and Carol Zander. Students'

perceptions of the differences between formal and informal learning. In *Proceedings of the Seventh International Workshop on Computing Education Research*, ICER '11, pages 61–68, New York, NY, USA, 2011. ACM.

[7] Neil Christopher Charles Brown and Michael Kölling. A tale of three sites: Resource and knowledge sharing amongst computer science educators. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, pages 27–34, New York, NY, USA, 2013. ACM.

[8] Thomas D. Effland. Focused mining of university course descriptions from highly variable sources. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 716–716, New York, NY, USA, 2015. ACM.

[9] Michael Hewner. How cs undergraduates make course choices. In *Proceedings of the tenth annual conference on International computing education research*, pages 115–122. ACM, 2014.

[10] Peter Hubwieser, Johannes Magenheim, Andreas Mühling, and Alexander Ruf. Towards a conceptualization of pedagogical content knowledge for computer science. In *Proceedings of the ninth annual international ACM conference on International computing education research*, pages 1–8. ACM, 2013.

[11] Paul Jaccard. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50, 1912.

[12] Sandra Katz, David Allbritton, John Aronis, Christine Wilson, and Mary Lou Soffa. Gender, achievement, and persistence in an undergraduate computer science program. *ACM SIGMIS Database*, 37(4):42–57, 2006.

[13] Judy Kay, Michael Barg, Alan Fekete, Tony Greening, Owen Hollands, Jeffrey H Kingston, and Kate Crawford. Problem-based learning for foundation computer science courses. *Computer Science Education*, 10(2):109–128, 2000.

[14] Maria Klawe and Ben Shneiderman. Crisis and opportunity in computer science. *Communications of the ACM*, 48(11):27–28, 2005.

[15] David R Krathwohl. A revision of bloom's taxonomy: An overview. *Theory into practice*, 41(4):212–218, 2002.

[16] S. Lloyd. Least squares quantization in PCM. *Information Theory, IEEE Transactions on*, 28(2):129–137, Mar 1982.

[17] Mehran Sahami, Alex Aiken, and Julie Zelenski. Expanding the frontiers of computer science: designing a curriculum to reflect a diverse field. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 47–51. ACM, 2010.

[18] Pete Sanderson and Ken Vollmar. A primer for applying service learning to computer science. In *ACM SIGCSE Bulletin*, volume 32, pages 222–226. ACM, 2000.

[19] Dermot Shinners-Kennedy and Sally A. Fincher. Identifying threshold concepts: From dead end to a new direction. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, pages 9–18, New York, NY, USA, 2013. ACM.

[20] Yiming Yang, Hanxiao Liu, Jaime Carbonell, and Wanli Ma. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 159–168, New York, NY, USA, 2015. ACM.

[21] Stuart Zweben. Computing degree and enrollment trends. Technical report, Computing Research Association, 2011.