Course-Job Fit: Understanding the Contextual Relationship Between Computing Courses and Employment Opportunities

Christopher Lukas Kverne, Florida International University

Christopher Lukas Kverne is an undergraduate researcher pursuing a B.S in Computer Science, plus a minor in Mathematics, with a cumulative GPA of 3.86. His research interests lie Deep learning, Optimization and Quantum Machine learning where his goal is to optimize training processes and reduce the computing power needed to develop powerful models. Christopher has led multiple projects in ML, Systems and Quantum research and will continue his studies in graduate school.

Federico Monteverdi, Florida International University

Federico Monteverdi is an IT Applications Programmer Associate at Progressive Insurance, where he focuses on backend technologies and large-scale system integration. He earned his B.A. in Computer Science from Florida International University, graduating Magna Cum Laude with a 3.89 GPA, and completed consecutive internships in software engineering and testing before starting full-time in 2025. During his undergraduate studies, Federico received a full-ride research scholarship at the EPSI Lab, where he developed full-stack prototypes to support Ph.D. research in wireless power transfer. He later joined the DaMRL Lab, contributing to the core applications behind the paper Course-Job Fit: Understanding the Contextual Relationship Between Computing Courses and Employment Opportunities. He also served as a teaching assistant for Discrete Structures, with academic interests spanning systems design, applied machine learning, and translating research into scalable industry solutions.

Agoritsa Polyzou, Florida International University

Agoritsa Polyzou is an Assistant Professor at the Knight Foundation School of Computing and Information Sciences in Florida International University (FIU), Miami. Agoritsa received the bachelor's degree in computer engineering and informatics from the University of Patras, Greece, and her Ph.D. degree in computer science and engineering from the University of Minnesota. Next, she was a Postdoctoral Fritz Family Fellow with the Massive Data Institute of McCourt School of Public Policy at Georgetown University, Washington, DC. She is involved in projects in the intersection of education, data mining, machine learning, ethics, and fairness. Her research interests include data mining, recommender systems, predictive models within educational contexts, and the fairness concerns that arise from their use. Her goal is to help students succeed using data and machine learning models.

Dr. Christine Lisetti, Florida International University

Christine Lisetti is an Associate Professor at Florida International University (FIU) in the Knight Foundation School of Computing and Information Sciences , and the Director of the Virtual Intelligent Social AGEnts (VISAGE) Laboratory. Her long-term research goal is to create engaging virtual social agents (VISAGEs) that can help humans in a variety of contexts by interacting with them in innovative ways, through natural expressive multimodal interaction (e.g. in digital health interventions, cybertherapy, health counseling, educational serious games, cyberlearning, simulation-based social skill training systems). She conducts basic research at the intersection of human-computer interaction (HCI), affective computing (I was on the founding Editorial Board of the IEEE Transactions on Affective Computing), human-centered artificial intelligence (AI), and virtual reality (VR) in order to discover design principles for VISAGESs. She also conducts applied research by applying these principles to different application domains (e.g. healthcare, medicine, education, social skill training).

Janki Bhimani, Florida International University

Dr. Janki Bhimani is the Director of the Data Management Research Lab and a Rising Scholar Professor in Computer Science at Florida International University (FIU), as well as the founding CEO of LegalPro+. She holds a Ph.D. and M.S. in Computer Engineering from Northeastern University. Her research spans system design, storage systems, computer architecture, EDA, ML, cloud and quantum computing, HPC, and computer education. She is a distinguished scholar and recipient of several prestigious honors, including the NSF CAREER Award, FIU Top Scholar Award, and the In the Company of Women Award.

Course-Job Fit: Understanding the Contextual Relationship Between Computing Courses and Employment Opportunities

Abstract

In today's world, where higher education is increasingly vital, aligning curricula with industry demands is essential. This paper explores the contextual relationship between computing courses and technical jobs using various transformer models to encode course syllabi and job descriptions into high-quality fixed-sized vector spaces (embeddings), enabling efficient and nuanced comparisons that reveal deeper contextual relationships. Our research makes multiple unique contributions that address gaps in existing work. First, we gather a large, recent data set of 197,296 jobs in five technical fields. Secondly, we perform in-depth analysis between courses and job postings using advanced transformer models, offering clear and deeper insights into how well academic content aligns with the industry. Third, we investigate salary trends to identify courses and skills linked to high-paying jobs. Fourth, we examine core and elective courses separately to provide insights for curriculum development and assist students in choosing elective courses considering industry demands.

Our findings show that top-ranking courses emphasize a combination of technical skills and professional skills like communication and teamwork. Moreover, skills like cloud technologies and databases are prevalent across multiple fields, highlighting their importance as essential skills for success in various technical domains. We found that the core courses mandated by the curriculum for all students to complete generally align better with job market demands compared to elective courses. Interestingly, undergraduate courses exhibit stronger alignment with job postings overall, while graduate courses improve their alignment specifically with higher-paying jobs. This highlights the importance of considering career goals when deciding whether to pursue graduate education. Overall, this paper introduces a replicable methodology for analyzing curricula and demonstrates its application through a case study of one institution's computing programs.

1 Introduction

The rapid evolution of the job market, driven by artificial intelligence (AI) and automation, alongside shifting economic demands, underscores the need for an adaptable education system. Although educational institutions strive to equip students with the necessary knowledge for successful careers, many graduates struggle to land jobs that match their qualifications, even with the high demand for tech talent. A 2024 study conducted by Hanson et al. [21] found that approximately 37% of students in fields such as computer science (CS) are underemployed. A common factor behind this phenomenon is the significant gap between academic curricula and employer expectations [22]. Graduates may feel they are prepared to enter the workforce, but there is often a mismatch between the skills and knowledge they have acquired and the requirements of employers [13, 22, 29, 41].

This research aims to bridge the gap between educational content and employment needs by leveraging advanced analytics techniques of machine learning (ML) and transformer models to analyze and correlate course descriptions with job postings. By identifying the alignment or misalign-

ment between educational content (such as course syllabi) and employment needs (such as job descriptions), this study aims to provide valuable insights for student career advising, curriculum analysis, and policy-making. Ultimately, the goal is to enhance graduate employability and ensure that educational institutions remain responsive to the evolving needs of the job market. This study addresses the following research questions (RQs):

RQ1: How can we transform course syllabi and job descriptions into fixed-sized vector embeddings to compare and reveal how well academic content aligns with industry needs?

RQ2: How can we identify the courses and skills related to high-paying job opportunities?

RQ3: How do core and elective classes differ in their alignment to the job market?

Anonymized experimental data and source code will be made publicly available on GitHub (https://github.com/Damrl-lab/Course-Job-Fit) to facilitate further research and extension into other educational and job domains. While in this paper our focus is on technical jobs and computing course syllabi from the computing department of an R1 Hispanic Serving Institution (HSI), this approach could be extensively applied to compare other job fields and course contents such as catalogs, syllabus, and lecture materials. Our findings are contextual to a single institution but provide a framework for curriculum analysis that can be applied across diverse educational settings.

In the remainder of this paper, Section 2 elaborates on the related work in the field and our contributions. Section 3 introduces the details of our data collection and pre-processing. Section 4 elaborates on our methodology to analyze and compare course syllabi with job descriptions. Section 5 evaluates our implementation by ranking courses and analyzing them to identify trends. Section 6 provides a clear guide for re-using our study. Section 7 discusses potential limitations of this study, while Section 8 summarizes our work and concludes the paper.

2 Related Work

In this section, we discuss existing works on identifying employability skill gaps, efforts to standardize and understand curriculum changes, and job-to-course comparison studies, emphasizing the gap in exiting works, unique context and need for our research in this paper.

Carnevale et al. [9] highlighted the growing importance for higher education in preparing students for the job market emphasizing the need for cognitive and soft skills. Similarly, Markes [32] reviewed employability skills needed in engineering, finding a disconnect between academic training and industry requirements. Weligamage et al. [54] reinforced this concern, stressing the need of keeping curriculum up-to-date with current demands. In recent years many comprehensive reviews identified teamwork, communication, problem-solving, self-management, and technical skills as crucial for workforce success, advocating for curriculum updates to incorporate these skills [11, 14, 19, 25, 37, 45, 44, 46]. However, while these studies establish the importance of aligning curricula with industry needs, they often rely on discussions or feedback rather than direct comparisons between course syllabi and job postings making it difficult to identify how effectively educational content covers these topics.

Previous research has also leveraged natural language processing (NLP) techniques to analyze university curricula. Li et al. [27] used text mining and clustering to compare data science curricula

across institutions finding a lack of standardization in course content. Bhaduri and Roy [5] used Word2Vec [33] embeddings to analyze college mission statements, finding similarities in public and private institutions. West [55] used Term Frequency-Inverse Document Frequency (TF-IDF) to identify key skills taught in data science curricula, discovering that many institutions tend to mimic each other's curricula rather than developing programs tailored to the industry. Although these studies present trends across academic institutions, they don't provide concrete comparisons between the job market and academic content.

Other studies have applied basic NLP techniques for job-to-course comparisons. Kawintiranon et al. [23] and Xun et al. [58] compared course content with specified skill lists from sources such as the Association for Computing Machinery (ACM), by using latent semantic analysis (LSI) and TF-IDF finding a gap between skills taught and listed. Alibasic et al. [1] and Fortino et al. [17] took a more direct approach by comparing course and major descriptions with job postings, identifying critical skill gaps by using TF-IDF and LSI. However, these methods (LSI and TF-IDF) often fall short in capturing the nuanced context of job descriptions and course content, a limitation that can be addressed using recent advancements in ML and NLP, particularly transformer models [24, 31, 50, 59].

Our research stands out from previous studies in several ways:

1. **Data Set**: We collected a large dataset of 197,296 jobs posted between October-December 2024 from popular recruiting sites such as Glassdoor, Indeed, and ZipRecruiter to compare courses with recent and relevant employment opportunities.
2. **Semantic Depth:** Unlike traditional NLP methods such as TF-IDF and LSI, our use of transformer models capture the nuanced context of job descriptions and course content, providing a deeper semantic analysis.
3. **Direct Comparisons:** We perform direct 1:1 comparisons between course syllabi and job postings, rather than relying on predefined skills.
4. **Incorporation of Salary Data:** We examine salaries to identify high-paying skills, offering insights into the economic value of specific courses.
5. **Core and Elective Analysis:** We separately analyze core and elective courses to assist students in class selection and assist universities in reevaluating their curriculum.
6. **Computing Field:** By focusing on computing and tech, some of the fastest-growing fields [7, 18], our research helps an important knowledge group in today's society.

In summary, while previous studies have laid the groundwork for understanding the alignment between education and the industry, our research advances the field by leveraging advanced techniques like transformer models, while also using a larger more recent dataset, and adding additional layers to the analysis of the courses. To the best of our knowledge, none of the existing works explore the aspects of salaries and core vs. elective classes in their analysis.

3 Data Collection

In this section we present our methodology for gathering job postings and processing the course syllabi and job descriptions.

3.1 Course Syllabi

We gathered course syllabi from the course catalog of a computing department at a public university in the United States, where a total of `210 courses` are listed. These courses come from various computing programs such as undergraduate and graduate programs in IT, CS, Data Science (DS), Cybersecurity (CYS), and more. The course syllabus is a PDF document typically between 1-3 pages that contain a detailed description of what each course covers. This includes fields such as prerequisite requirements, learning outcomes, grading policies, and topics taught. While formats and lengths would vary, they were generally similar in their structure. The average length of these documents are 717.2 words making them sufficiently descriptive.

It's important to note that different professors might adapt these syllabi based on their preferred topics or teaching approaches. However, we used a standardized version designed to capture each course's content as accurately as possible regardless of the instructor. These standardized syllabi have been reviewed to ensure they effectively represent the course curriculum. We believe this approach provides a reliable representation of course content while acknowledging that some variation may exist in actual course delivery, a limitation further discussed in Section 7.

3.2 Job Fetching

Job postings were collected using `JobSpy`, a Python repository designed for scraping job listings from various websites [53]. JobSpy can scrape data from recruiting websites, such as Glassdoor, Indeed, LinkedIn, and ZipRecruiter [53]. We excluded LinkedIn for our job collection because it tended to block requests after scraping 10-50 jobs, and could potentially lock us out from their services after extensive usage. The scraper uses APIs from each recruiting website and sends a request looking for jobs based on queries passed. The API returns multiple pages of jobs, where pages are sorted by relevance to our queries, and fetches them page by page until the desired number of jobs has been retrieved [53].

Computing degrees enable graduates to pursue multiple career options. To reflect this, we have collected jobs from five different fields common for computing graduates to work in. These fields were selected based on public data released by universities such as Carnegie Mellon and University of Southern California about where their CS graduates go after graduation[8, 49]. These five fields are:

- Cyber Security (CYS)
- Data Science (DS)
- Software Engineering (SWE)
- Technical Product Management (TPM)
- Information Technology (IT)

We acknowledge that a computing degree can lead to jobs outside these fields, however the selected fields will represent the vast majority of graduates career outcomes and goals [8, 49]. Moreover, the jobs within each field are very diverse, offering a broad range of technical roles. An example of this were the Software engineering jobs where roles included topics like Fullstack development, Game development, Systems engineering, DevOps engineering among others. Overall, we believe this dataset is of high quality and covers recent, relevant, and diverse roles most computing graduates pursue after completing their studies.

For our software engineering jobs we used these parameters: `"Location: USA"`, `"Search Term: Software Engineer"`, `"Results Wanted: 2,000"`, and `"Hours Old: 24"`. Using these parameters we would receive up to 2,000 software engineering job postings that were 24 hours old or less and located in the USA. The number of jobs would fluctuate based on the day and field, where we would typically receive around 500-1,000 jobs for each category per day. We used similar queries for the other fields, however we changed the search term to `Cyber security`, `Data scientist`, `Technical product manager` or `IT` respectively.

We executed the job fetching script starting on 10/10/2024 and repeated the data fetching every 1-3 days until the 12/30/2024, ensuring the jobs were recent, until we had gathered a total of 197,296 total job postings. The distribution is: 38,514 Cyber security jobs, 28,896 Data science jobs, 61,159 Software engineering jobs, 29,199 Product management jobs, and 39,528 IT jobs. The returned job postings required further filtering and pre-processing because they included numerous fields not relevant to our analysis, such as CEO name, job URL, and number of employees. Our primary focus is on the job title, salary, and description. Additionally, job descriptions varied significantly: some postings were longer due to information about the company, diversity policies, and contact details, while others focused primarily on the job and its requirements. The average length of the job descriptions is 715.8 words with a standard deviation of 386.6 words.

3.3 Data Processing

To analyze the job postings, we filtered and pre-processed the data to ensure it was clean, consistent, unique and ready for input into the transformer models.

Filtering: First, we filtered the data to exclude job entries that did not include a job description and removed duplicate job postings from each field, which reduced the dataset to 95,376. Many job postings listed their salaries in ranges where a minimum and maximum salary are given. For our analysis, we used the average of these values. Some jobs listed salaries on an hourly basis, while others used a yearly format. We converted hourly salaries to yearly amounts by multiplying by 52 (weeks per year) and then by 40 (hours per week). We understand that job salary may depend on location and additional benefits, but this is outside the scope of this study, as we are focusing only on the average earnings.

Pre-processing: After filtering, we pre-processed the job descriptions to make them easier for the models to process. This involved converting all text to lowercase and removing extra white space. To pre-process the course syllabi we apply the same text cleaning process as we did for the job descriptions, which involves converting all characters to lowercase and removing extra whitespace.

4 Methodology

Our methodology for comparing courses with jobs is summarized in Figure 1. It involves gathering and processing the data, inputting it into the models, which tokenizes the input and generates an embedding for each token, passing the embeddings through multiple transformer layers, combining these embeddings into a single sentence embedding, and finally calculating the similarities between each course and job embedding to rank and analyze the courses. Each model was run separately on a single NVIDIA A100-PCIe-40GB GPU with a total execution time between 78-140 minutes based on the model used.

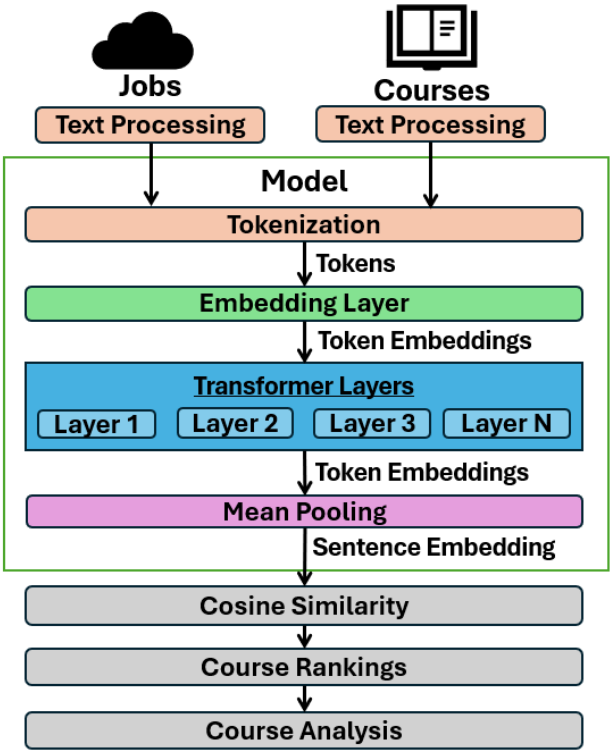

Figure 1: Process Overview

4.1 Model Selection

Extensive research has shown the effectiveness of transformer models in tasks such as creating sentence embeddings and calculating semantic similarities, where they consistently outperform classical NLP methods like LSTM networks, algorithms such as TF-IDF and LSI, and word-level embedding models like Word2Vec [10, 12, 43, 50]. Their advantage lies in the self-attention mechanism that enables them to process entire sequences simultaneously and capture relationships and contexts between words that traditional approaches might miss [12, 50].

For example, when analyzing a course syllabus describing ”Python applications in data analysis,” traditional methods like TF-IDF might treat these as separate concepts, while transformers can understand how they relate to job requirements. Additionally, transformers can recognize semantic similarities even when different terminology is used, such as understanding that ”software

development" and "programming" describe related concepts, or that "cloud computing" relates to specific platforms like AWS or Azure. This capability is particularly valuable for our study, as job descriptions and course syllabi often use different terminology to describe similar skills and concepts [50]. In this study we use encoder-only transformer models which have been specifically designed for converting text to embeddings [6, 26, 42]. When selecting which models to use we look at the following criteria.

1) Open source: We want our study to be easily replicable and free, which is why we decided to use open source models.

2) Accuracy: We want to use models designed specifically for creating text embeddings that achieve top performance for tasks such as semantic similarity comparisons.

3) Performance: We want to use models which can generate embeddings efficiently without the need of extensive computing power, further enhancing the replicability of this study.

Looking at these three criteria, we decided to use these five models.

- Sentence-BERT (SBERT) [43]
- BAAI General Embedding (BGE) [57]
- General Text Embeddings (GTE) [28]
- MPNet [47]
- E5 [51].

SBERT and MPNet were chosen due to their speed, popularity and compact size [43, 47], while E5, BGE and GTE were chosen as they are some of the most recent and powerful embedding models to date [28, 51, 57]. The decision to use multiple models stems from our effort to alleviate varying limitations of individual transformers. Transformers have been documented to struggle with technical or domain-specific terminology if they are not trained on this type of data [2, 30]. Furthermore, their embeddings can be sensitive to changes in text structure or wording, which can result in inconsistent similarity comparisons [52]. Lastly, their fixed token lengths require creative approaches for handling longer documents [3, 39] which we discuss further in Section 4.2. By using multiple models we can mitigate these limitations as it offers several advantages such as:

1. Training Data: Each model has been trained on different data. For example, MPNet was primarily trained on BookCorpus and English Wikipedia [47], while E5 included technical documentation and academic papers [51]. These differences can therefore lead to strengths in different domains which is crucial when working with the large diverse dataset employed in this study.
2. Architectural Variations: The models use different architectures (e.g., SBERT's siamese architecture [43] or BGE's contrastive learning approach [57]) which affect how they capture semantic relationships. Prior work has shown that these differences can make the models capture and focus on different aspects of the text when encoding it, complementing each other when used together [10, 28, 35].

3. Domain Specialization: Individual models may show stronger performance in certain domains or writing styles. For example, some models might better capture technical terminology while others excel at understanding broader conceptual relationships. This phenomenon has been documented in [2, 30].

By combining multiple models, we create a more robust evaluation system where individual models' strengths and limitations balance each other out. Table 1 compares the five models based on the number of parameters, Massive Text Embedding Benchmark (MTEB) score [35], release year, token limit, and embedding dimension. The MTEB score is the standard evaluation metric for embedding models, evaluating their performance for tasks such as semantic similarity, classification, and clustering, using various datasets [35]. All the models used have respectable MTEB scores reflecting their capability at generating high-quality embeddings. While SBERT and MPNet have slightly lower scores compared to the newer, larger models like GTE and BGE, their smaller size and faster processing times make them valuable for scenarios requiring computational efficiency.

Model	**Params**	**MTEB**	**Year**	**Token lim**	**Dim.**
SBERT	22M	56.09	2019	256	384
BGE	335M	64.23	2023	512	1024
GTE	335M	63.13	2023	512	1024
MPNet	110M	57.17	2021	384	768
E5	335M	62.20	2023	512	1024

Table 1: Comparison of Embedding Models.

4.2 Generating embeddings

All five models are based on the transformer architecture, although they differ in their specific implementations. The general process of converting text to embeddings follows these steps:

1. Tokenization: Text is broken into tokens where each token represents a word or a subword and their numerical representation which the models can process.

2. Input Encoding: Each token is converted into an embedding which combines token embeddings (numerical and contextual representation of each token), position embeddings (information about where each token appears in the sequence), and special tokens in certain models (tokens like [CLS] which represents the entire sequence for classification purposes or [SEP] which marks boundaries between different text segments) that help the model understand the structure of the input.

3. Attention Processing: The encoded input is then passed through multiple transformer layers which apply self-attention mechanisms. This helps the model evaluate the relationships between different tokens and determine which tokens are the most relevant for each other based on their context. The output is then refined further by feed-forward networks, where each layer progressively builds a deeper understanding of the text.

4. Pooling: Finally, to get a single embedding for the entire text we need to merge the embedding of each token together. SBERT and MPNet use mean pooling over their last layer [43, 47], while for GTE, BGE and E5, this must be implemented manually [28, 51, 57].

The resulting embeddings are high-dimensional vectors (between 384-1024 dimensions) that capture the semantic meaning of the input text, allowing for similarity comparisons between different texts. When tokenizing the course syllabi and job descriptions, a possible issue that can arise is the token limit. If our text exceeds the token limit, the tokenizers will by default truncate our text to fit the limit, meaning that the embeddings created will not represent the entire course syllabi or job description.

To address this issue, we developed a weighted embedding strategy to effectively represent all the text in the job descriptions and course syllabi represented in Algorithm 1. When a description exceeds the token limit, we split the tokens into chunks with a maximum size of *max_seq_len* which represents the largest number of tokens each model can process (line 6-8). Then we pass the chunks through the models which return an embedding for each of them (line 10). We make sure to keep track of the number of tokens each embedding represents (line 11-13). Finally, we combine these embeddings, where we weigh them based on how many tokens they represent (line 14-17). This strategy ensures that all of the text is included, while also giving fairness to embeddings representing more text.

No method for solving this problem is perfect, and drawbacks of this approach include additional computational overhead [3, 39]. Moreover, this approach will give more importance to embeddings representing more text although more text doesn't necessarily equal more importance. As most of the course syllabi and job descriptions contained substantial content, around 80-90% of them exceeded the token limit, based on which models was used, making this step crucial for keeping all information.

Algorithm 1: Generate Embeddings

```
   Input: text, max_len, tokenizer, model
1  tokens = tokenizer(text);
2  if tokens.length ¡= max_len then
3      embedding = model(tokens);
4      return embedding ;
5  else
6      chunk = [ ], embeddings = [ ], token_lens = [ ];
7      foreach (token, i) in tokens do
8          chunk.append(token);
9          if chunk.length == max_len or i == tokens.length - 1 then
10             chunked_embedding = model(chunk);
11             embeddings.append(chunked_embedding);
12             token_lens.append(chunk.length);
13             chunk = [ ];
14     final_embedding = [ ];
15     foreach (emb, t_len) in (embeddings, token_lens) do
16         final_embedding += emb * (t_len / tokens.length);
17     return final_embedding;
```

4.3 Course Rankings

After transforming every job and course into an embedding, we calculated the cosine similarity between them, which gave us a normalized value between 0-1, where 0 means no similarity, and 1 means that they are identical. With 210 courses and 95,376 jobs we made $210 * 95,376 = 20,028,960$ different comparisons. Cosine similarity was selected as our comparison metric, as it gives us a normalized range (0-1) while also being the recommended metric for similarity comparison of text embeddings [43, 47]. We then find the average similarity each course has to every field by using Equation 1,

$$AS_i = \frac{\sum_{j=1}^{N} S_{ij}}{N}, \tag{1}$$

where AS_i is the average similarity of course i, N is the total number of jobs, and S_{ij} is the similarity between course i and job j. As seen in Table 2, the range and mean values for the average similarity differ based on the model used. This behavior comes from the differences in the models' structure, training, and purpose. To normalize these values, we rank each course from 1-210 (1 being the best) and calculate their combined rank given by:

$$R_i = \frac{R_i(SBERT) + R_i(BGE) + R_i(GTE) + R_i(MPNet) + R_i(E5)}{5}, \tag{2}$$

where the rank of a course R_i is calculated by summing its rank from the five models and averaging them out.

Model	Mean	Min	Max
SBERT	0.29	0	0.81
BGE	0.66	0.39	0.89
GTE	0.81	0.66	0.95
MPNet	0.25	0	0.82
E5	0.79	0.67	0.89

Table 2: Average Similarity Model

4.4 Course Analysis

To analyze the courses, we perform a topic analysis, a salary analysis, and a core vs. elective comparison. The results of the topic analysis can be found in Section 5.2, the results of the salary analysis can be found in Section 5.3, and the results of the core vs. elective comparison can be found in Section 5.4.

Topic Analysis: While the course rankings provide valuable information and can be used to identify courses with little or high alignment to the industry, they fail to explain why courses perform better or worse in each field. To gain a better understanding of what drives these rankings, we performed a topic analysis of the course content by examining the specific skills covered in each course. To achieve this, we extracted keywords from each course using KeyBERT [20]. The process works in three steps: First, KeyBERT generates two types of embeddings, one for the entire course syllabus (document embedding) and individual embeddings for each word in the syllabus.

Second, it calculates the cosine similarity between each word embedding and the document embedding to determine how semantically important each word is to the overall document. Finally, we select the 15 words with the highest similarity scores and save them as keywords [20].

For this analysis, we excluded generic keywords such as ”prerequisite,” ”textbook,” or ”course” as they are present in a majority of courses (ranked high and low) and offer little insight into the course's subject. After filtering, the remaining keywords provide a detailed summary of the core topics covered in each course, helping us identify what skills the industry values. This method allows us to connect the rankings to specific themes or trends, offering actionable insights for curriculum development.

Salary Analysis: Understanding salary trends and their association with courses is critical for aligning educational offerings with student aspirations. While students choose majors and courses based on multiple factors such as personal interest, career goals, and academic strengths, salary potential remains a significant consideration. Studies show that students increasingly select college majors and courses based on their perceived potential to secure high-paying jobs, reflecting a growing trend of prioritizing economic returns on educational investment, especially in fields like computing and engineering [4, 34, 36, 56].

Recognizing that salary is just one of many factors in course selection, but an important one for many students, we analyze the relationship between courses and high-paying jobs to help students who prioritize salary potential in their course selection. To address RQ2 and identify the courses most closely aligned with high-paying jobs, we follow a similar approach to the course rankings. However, instead of comparing every course with every job, we first identify the top 1% highest paying jobs and then calculate the similarities between the courses and these high paying jobs. Using Equations 1 and 2, we re-rank the courses based on their alignment with these high-paying job postings and analyze the new course rankings.

Core vs. Elective Comparison: Courses can be divided into core and elective courses, where core courses are required to complete your major, and electives can be chosen. While universities design curricula considering multiple factors including fundamental knowledge, theoretical foundations, and broader educational and social goals, understanding industry alignment can provide valuable insights. Core courses serve dual purposes as they cover essential foundations while preparing students for professional success. Analyzing this alignment can reveal whether courses we assume to be industry-relevant might be missing certain skills or topics highly valued in the industry, helping universities better understand potential gaps in their curriculum. It is therefore important to compare core and elective courses separately for two main reasons:

1. Students can decide which electives to take based on their relevance and alignment with their career goals.
2. Universities can identify potential gaps between core course content and industry requirements, while balancing this with fundamental knowledge.

To compare core and elective courses (RQ3), we calculate the average ranking of each group. First, we determine each course's ranking using Equation 2. Then we separately compute the mean ranking of core and elective courses allowing us to compare their overall alignment with the industry across all five fields. The core and elective courses compared in Section 5.4 come from

the B.S in CS program.

5 Results

In this section, we conducted a comprehensive analysis of the results based on our methodology. Section 5.1 shows the results regarding the top course rankings and model agreement, Section 5.2 identifies the keywords of the top courses and analyzes them, Section 5.3 shows the results of the salary analysis and Section 5.4 compares core and elective courses. The results in this section come from analyzing one institution's computer science curriculum and therefore serve as a case study demonstrating our methodology's application.

Our analysis reveals three distinct types of insights:

1. Ranking Methodology Validation (Section 5.1): This section evaluates the consistency and reliability of our course ranking approach across different models.
2. Industry-Level Findings (Section 5.2): This section includes findings of more general trends in the job market and what skills or topics they favor in curricula.
3. Institution-Specific Findings (Sections 5.3 and 5.4): Our analysis of salary potentials and relationship between core and elective courses is specific to our institution's curriculum and might vary at different universities. In these sections we highlight results to look for and how to interpret them to analyze and improve curricula.

5.1 Evaluating Course Ranking Metrics

Figure 2 shows the Spearman correlation [48] of the course rankings in all fields. The Spearman correlation measures the strength and direction of association between two ranked variables which in our case means how similarly each pair of models ranks the same set of courses. The Spearman correlation is calculated as:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{3}$$

where d_i is the difference between the two ranks of each course and n is the number of courses. This coefficient ranges from -1 to +1, with values closer to +1 indicating stronger positive correlation. The correlations in our study range from 0.58 - 0.90, showing moderate to high agreement in the course rankings. Although the exact similarities produced by the models differ as seen in Table 2, the rankings of the courses are very similar. The top 10 courses in each field are summarized in Table 3. Courses marked with $^{\text{C}}$ and $^{\text{E}}$ represent core and elective courses respectively. For a complete list, all our code and data is available on GitHub and can be used to identify courses with little or high alignment to the industry. The course rankings for the core courses are particularly important as every computing student must take them, while the course rankings for the elective courses can help students in course selection based on what fields they're interested in.

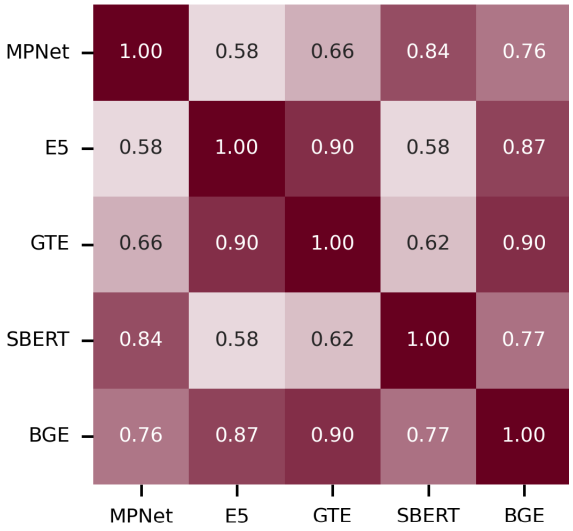

Figure 2: Model Rank Spearman Correlation

Field	Top Ranked Courses
CYS	Empowering Emerging Tech Talent [E], Enterprise IT Troubleshooting [E], Emerging Topics in Digital Life [C], Cloud Essentials [E], Secure Application Programming [C], Enterprise Cybersecurity Policies and Practices [C], Software Engineering I [C], Internship Ready Software Development [E], IT Automation [E], Software Engineering II [C]
DS	Advanced Data Science [C], Empowering Emerging Tech Talent [E], Introduction to Data Science [C], Internship Ready Software Development [E], Introduction to Data Mining [E], Fundamentals of Data Science [C], Introduction to Deep Learning [C], Artificial Intelligence for All [E], Senior Project [C]m Artificial Intelligence [E]
SWE	Empowering Emerging Tech Talent [E] , Internship Ready Software Development [E] , Software Engineering I [C] , Software Engineering II [C] , Enterprise IT Troubleshooting [E] , Senior Project [C] , Cloud Essentials [E] , Components-Based Software Development [C] , Software Design and Development Project [E] , Advanced Data Science [C]
TPM	Empowering Emerging Tech Talent [E] , Internship Ready Software Development [E] , Software Engineering I [C] , Introduction to Data Science [C] , Senior Project [C] , Software Engineering II [C] , Enterprise IT Troubleshooting [E] , Advanced Data Science [C] , IT Automation [E] , Cloud Essentials [E]
IT	Empowering Emerging Tech Talent [E] , Enterprise IT Troubleshooting [E] , Internship Ready Software Development [E] , Software Engineering I [C] , Senior Project [C] , Cloud Essentials [E] , IT Automation [E] , Software Engineering II [C] , IT Fundamentals [C] , Introduction to Data Science [C]

Note: [C] and [E] markers represent core and elective courses respectively.

Table 3: Top Ranked Courses based on Current Course Syllabi

5.2 Findings - Key Terms

Using KeyBert [20], Table 4 shows a list of keywords for the 10 highest ranked courses in each field. The keywords extracted from the top courses can generally be grouped into three categories:

1. **Teaching or Course Related:** Any words or topics related to teaching. These include words like: *Course, Prerequisite, Textbook* etc. **These words are filtered out**.
2. **Technical Skills**: Words that highlight specific technical skills taught in a course like: *Python, AWS, Data mining*, etc.
3. **Professional Skills:** Words related to professional skills taught in a course like: *project, career, communication* etc.

Our analysis of the technical terms revealed both field-specific and cross-disciplinary skills. While each field maintains its core technical focus (e.g., security specific terms in cyber security and AI related terms in data science), several terms related to Cloud technologies, database skills, or programming languages like Python appeared consistently across all fields. These skills are therefore essential for the industry as they are applicable for various technical roles. Product management had the most diverse technical skills, combining topics from IT, software engineering, and data science.

Importantly, terms like ”team”, ”project”, and ”internship” appeared consistently across all fields, highlighting the industry's focus on practical experience and collaborative skills. This focus on professional skills suggests that the industry seeks well-rounded students who can combine technical expertise with strong communication, management, and real-world experience. Additionally, our analysis found that broader knowledge areas like ”Data Mining” or ”Data Analysis” appeared more frequently than specific technologies or programming languages related to them. Based on these findings, we recommend universities to:

1. Integrate more professional skills related to careers, internships, and communication into their curriculum.
2. Ensure comprehensive coverage of cloud and database technologies across core and elective courses.
3. Focus on teaching broader technical concepts rather than specific programming languages or frameworks.

5.3 Salary Results

This section evaluates the salary potential of courses by comparing how well graduate and undergraduate courses align with jobs in general versus high-paying jobs specifically. Course alignment is measured using the average similarity score and rank described in Section 4.3, where we calculate the average rank of graduate and undergraduate courses across all jobs and compare it to their average rank when considering only the top 1% highest-paying jobs. The course ranks compares each course's alignment to the industry with the other courses, where a lower rank indicates a stronger alignment. This is shown in Figure 3 where:

- Light blue bars show graduate course alignment with all jobs.

- Light orange bars show undergraduate course alignment with all jobs.
- Dark blue bars show graduate course alignment with top 1% highest-paying jobs.
- Dark orange bars show undergraduate course alignment with top 1% highest-paying jobs.

Field	Keywords in Top Ranked Courses
CYS	patterns enterprise, systems assessment, integration cloud, ipt1 architectures, team, troubleshooting coverage, cybersecurity, sharing, cloud certifications, azure, vulnerabilities security, stackguard, enterprise cybersecurity, countermeasures, risk analysis, model assessment, analysis, internship, automation mgmt, systems basics, software planning, project, iot risk, ip, mobile analysis, security subject, tcp, java, scripting basics, systems shell, computing network, administrating computing, python, internet, computer design, computer ethics
DS	data science, data analysis, internship, data mining, robotics, visualizations, introduction ai, ai problems, force computing, programming, planning multiagent, models assessment, learning, algorithm runtime, certifications cloud, azure object, computational thinking,queries course, enterprise database, excel, patterns enterprise, systems assessment, integration cloud, troubleshooting coverage, team project, mastery
SWE	database, software components, creating web, integration cloud, project team, software testing, software planning, systems assessment, internship, assessment, patterns enterprise, troubleshooting coverage, implement requirements, certifications cloud, analysis experiences, azure object, software engineering, data science, data, automation mgmt, systems basics, information science, jsp sending, queries, application, administration computing, python
TPM	software testing, management, technical writing, communications, analysis experiences, internship, models assessment, data, software planning, patterns enterprise, systems assessment, integration cloud, troubleshooting coverage, data science, automation, systems, project team, master problems, knowledge focus, information science, cybersecurity, software engineering, queries, enterprise database', computational thinking, algorithms, sharing digital, telecommunications, software components
IT	patterns enterprise, systems assessment, integration cloud, ipt1 architectures, troubleshooting coverage, cybersecurity, team, analysis experiences, internship, models assessment, certifications cloud, azure object, automation mgmt, systems basic, software planning, information science, data course, project, technical writing, communications, administration computing, downloading internet, telecommunications embedded, cybersecurity beginners, sharing digital, vulnerabilities security, stackguard, queries course, enterprise database, introduction linux',software engineering

Note: Keywords are selected based on semantic relevance using KeyBERT, with ordering not indicating relative importance.

Table 4: Keywords in Top Ranked Courses

We noticed three following trends:

1. **Graduate Course Improvement:** Graduate level courses improved their alignment with higher-paying jobs (comparing light vs. dark blue bars), specifically noticeable in the fields of Cyber Security, IT and Software Engineering.
2. **Undergraduate Course Performance:** Undergraduate courses showed strong alignment in all fields with a slight decrease in performance for higher-paying jobs (comparing light vs. dark orange bars).
3. **Consistent Top Courses:** The course rankings for the top 10-20 courses remained consistent with our findings in Section 5.1 (comparison of courses with every job), suggesting that higher paying positions similarly value a combination of technical knowledge and professional skills.

Based on these findings, several insights emerge. First, the consistently stronger performance of undergraduate courses across all fields suggests that our undergraduate curriculum is more strongly aligned with industry needs. This could indicate that undergraduate courses more successfully balance fundamental concepts with practical skills that employers value.

The lower alignment of graduate courses shouldn't necessarily be viewed as a weakness. Graduate education serves multiple purposes beyond industry alignment, as it often focuses on research methodology, theoretical foundations, and specialized knowledge that may not be immediately reflected in many of our job postings [15, 40, 16, 38]. However, the improvement in graduate course alignment with high-paying jobs in fields like Cybersecurity, IT, and Software Engineering suggests that advanced knowledge becomes more valuable at senior level roles in these domains.

This pattern raises important questions about the role of graduate education. For students primarily focused on maximizing job opportunities, it is important to determine whether their target careers explicitly require an advanced degree or if work experience alone can lead to similar or better career progression. Graduate school should be pursued strategically, particularly for those aiming for research-intensive roles, academic careers, or positions that explicitly demand advanced education.

5.4 Core vs. Elective Results

Figure 4 shows the average rank for the core and elective courses for the B.S in CS program in each field. The core courses outperform the elective courses in all fields, as their rankings are lower across the board (1 being the top course and 210 being the worst). The field with the smallest difference was Data Science, while Cyber Security exhibited the greatest difference between core and elective rankings. The B.S in CS elective courses at this university were divided into three groups: Systems, Mathematics, and Applications, making the topics quite diverse. Many electives focused on more theoretical concepts, which likely contributed to their lower rankings, as theoretically and mathematically oriented courses tended to score poorly in terms of similarity measurements.

The trend of core courses outperforming electives is advantageous, as students can choose electives with higher rankings, while core courses are mandatory. Therefore, our university has made a strong selection of core courses. We encourage other universities to add this aspect to their curriculum analysis and if core courses are being outperformed by electives, changes based on our

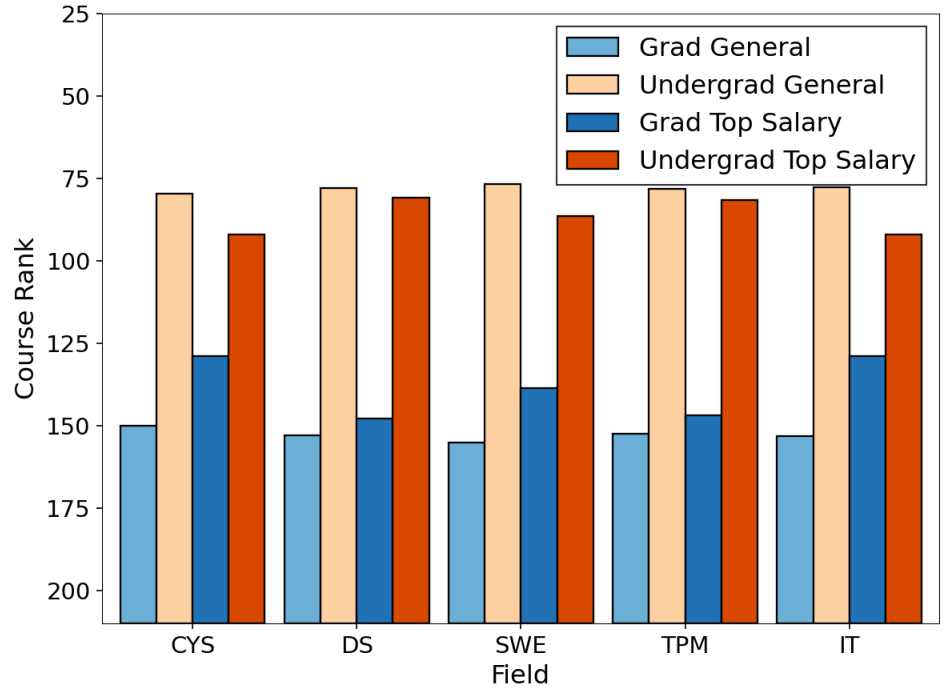

Figure 3: Salary Trends: Graduate vs. Undergraduate

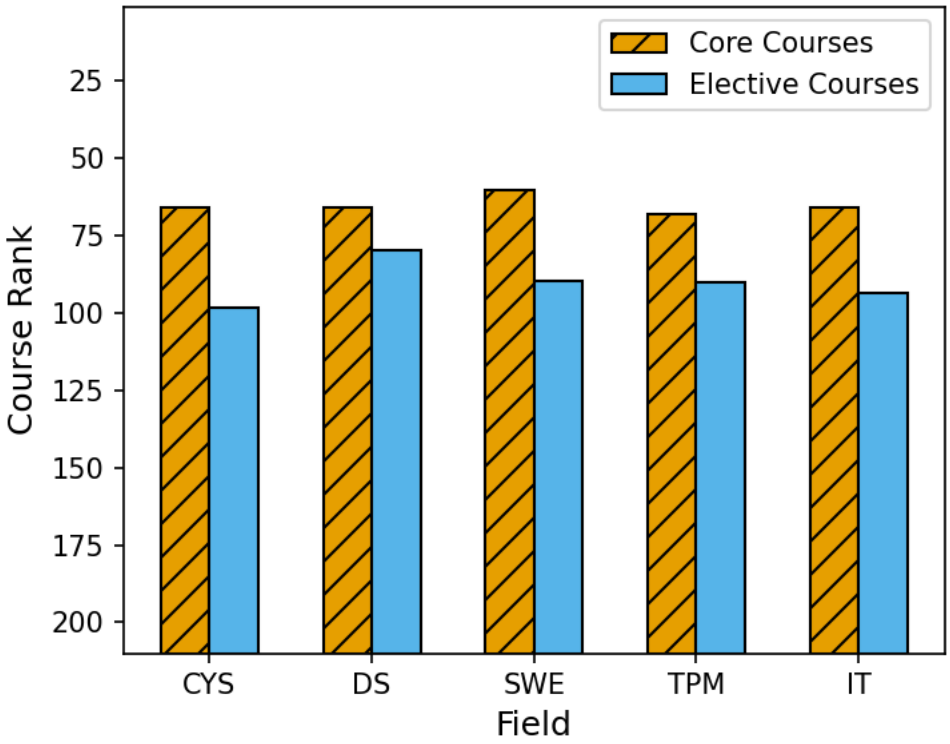

Figure 4: Core vs. Elective Course Rankings

findings and recommendations in Section 5.2 could be made to make them more relevant to the industry.

6 Reproduce and Use

As this study aims to enhance employability by analyzing curricula, our ultimate goal is for institutions to adopt and build on our methodology. Although this study focuses specifically on computing majors and five technical fields, it can be replicated to compare any major with any industry field. To reproduce our study, the following six steps should be taken:

1. **Data Collection:** Select the fields and majors you wish to compare, and collect syllabi and job postings from those fields. While our dataset of 197,296 job postings is available online, you can gather your own dataset by using our data-fetching code and adjusting the search parameters. Although our dataset is large and significantly expands on previous works [1, 17, 23, 58], a smaller dataset is also acceptable. Our codebase includes logic for reading course syllabi PDF files and cleaning both course and job data. Alternatively, course descriptions can be used, which provides less information about a course but gives a general idea of what it covers.

2. **Generate Embeddings:** While this study uses five different models, using fewer (or even just one) can also produce good results. Our codebase includes logic for generating sentence embeddings with five different transformer models while handling text exceeding the token limit. The code will automatically calculate the similarities between courses and jobs and save the results in a CSV file for easy analysis.

3. **Course Ranking:** After generating sentence embeddings, our codebase calculates the average similarity for each course and ranks them to normalize the values across all five models. While average similarity is a logical choice for comparing courses, other metrics could also be explored. Our codebase includes logic for comparing model rankings.

4. **Course Analysis:** After ranking the courses, we analyze the top courses to understand what makes them highly rated. Our codebase includes logic for using KeyBERT to extract keywords, however other approaches could also be explored.

5. **Core vs. Elective:** We encourage researchers to analyze core and elective courses separately. As core courses are mandatory, the results should ideally show that core courses perform better across industry fields. Moreover, examining elective courses independently can highlight which electives are best suited for each field, aiding students in course selection. Our codebase can be copied directly for this step.

6. **Salary Analysis:** To identify the courses with the greatest salary potential, we compare the courses to the highest-paying jobs. We also analyze graduate and undergraduate courses separately to observe how trends might differ. This step is important for students driven by salary potentials but not strictly necessary for a thorough curriculum analysis.

7 Discussion

A current limitation of this study is the misleading evaluation of prerequisite courses. As prerequisite courses often cover fundamental topics necessary for other courses, they can potentially have a lower similarity with the industry. An example of this is the Data Structure course, which had an average rank of 79.6 out of 210. While it covers topics like recursion, searching, sorting and

data structures (lists, stacks, queues, trees etc.), concepts fundamental to programming and computer science, these terms rarely appear explicitly in job descriptions. While transformer models are able to capture the similarity between a specific course and job very effectively, our current methodology does not take a specific course's importance for other courses into account.

While this limitation might lead to some unfair rankings, our findings show that this limitation only affects a minority of the prerequisite courses. In the core vs. elective analysis, we found that core courses (which include many prerequisites) generally outperform elective courses. Additionally, many prerequisite courses scored highly in their rankings as they included professional skills or general knowledge applicable to multiple fields. The prerequisite courses most affected were typically more theoretical, like the Data Structures course. We encourage other researchers to study this problem and build on our methodology to evaluate them differently.

Another potential limitation concerns the course syllabi. Our study assumes that the course syllabi accurately reflect the learning outcomes of a course. If these documents are outdated or unrepresentative of the topics taught, the course rankings would be flawed. Moreover, even when syllabi are current, the actual course content and learning outcomes can vary between different instructors. For example, two professors teaching the same "Database" course might focus on different aspects, such as theoretical or practical concepts. This difference in course delivery and learning outcomes is not captured in our current methodology, which only relies on syllabus content. However, conducting a comprehensive analysis of course delivery and learning outcomes for different instructors requires a separate study beyond the scope of this paper, which we plan to explore in the future.

8 Conclusion

In conclusion, our research offers a comprehensive analysis of computing curricula by comparing them to the technical demands of five diverse fields. By ranking courses based on their similarity to industry needs, we identified those that best align with the evolving job market. Our findings emphasize the importance of curricula that balance both technical proficiency and professional skills, with top-performing courses incorporating these elements. Through a detailed analysis of core and elective courses, we provide valuable insights into elective prioritization, core course sequencing, and potential curriculum updates. This data-driven approach holds the potential to significantly improve student employability by helping universities tailor their offerings to better meet industry demands, addressing a crucial gap in higher education.

Our study also highlights the salary potential of courses, revealing that while undergraduate courses generally align better across all fields, graduate courses show notable improvement when it comes to higher-paying jobs. This insight is particularly important for students who are focused on maximizing their career earnings. Although this study focuses primarily on computing majors and technical fields like software engineering, our methodology is versatile and can be adapted to compare any academic discipline with its corresponding industry field. Although limitations such as biases in transformer models and the challenge of evaluating prerequisite courses exist, we have taken steps to mitigate these issues by employing multiple models. Additionally, we encourage future researchers to explore alternative methods for assessing prerequisite courses. Despite these limitations, we believe that our study sets a new standard for curriculum analysis, offering a valu-

able framework for future research, and providing actionable insights for both institutions and students.

9 Acknowledgments

This work was supported by the following NSF grants: CSR 2402328, CAREER 2338457, CSR 2406069, CSR 2323100, and HRD 2225201.

References

[1] Armin Alibasic, Himanshu Upadhyay, Mecit Can Emre Simsekler, Thomas Kurfess, Wei Lee Woon, and Mohammed Atif Omar. Evaluation of the trends in jobs and skill-sets using data analytics: A case study. *Journal of Big Data*, 9(1):32, 2022.

[2] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[3] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[4] Mark C. Berger. Predicted future earnings and choice of college major. *ILR Review*, 41(3): 418–429, 1988. ISSN 00197939, 2162271X. URL `http://www.jstor.org/stable/2523907`.

[5] Sreyoshi Bhaduri and Tamoghna Roy. A word-space visualization approach to study college of engineering mission statements. In *2017 IEEE Frontiers in Education Conference (FIE)*, pages 1–5, 2017. doi: 10.1109/FIE.2017.8190704.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] U.S. Department of Labor Bureau of Labor Statistics. Computer and information technology occupations, 2023. URL `https://www.bls.gov/ooh/computer-and-information-technology/home.htm`. Accessed: 2024-07-06.

[8] Carnegie Mellon University. First destination outcomes - employment, 2024. URL `https://www.cmu.edu/career/outcomes/post-grad-dashboard.html`. Last updated: January 12, 2024.

[9] Anthony P Carnevale, Nicole Smith, and Jeff Strohl. Recovery: job growth and education requirements through 2020. georgetown public policy institute, 2020.

[10] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[11] David C Curtis and Phillip McKenzie. Employability skills for australian industry: Literature review and framework development. 2002.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] Catherine A. DiBenedetto and Victoria C. Willis. Post-secondary students' perceptions of career readiness skills. *Journal of Agricultural Education*, 61(1):44–59, Mar. 2020. doi: 10.5032/jae.2020.01044. URL https://jae-online.org/index.php/jae/article/view/2047.

[14] Nuryake Fajaryati, Budiyono, Muhammad Akhyar, and Wiranto. The employability skills needed to face the demands of work in the future: Systematic literature reviews. *Open Engineering*, 10(1):595–603, 2020.

[15] David J Feola, Esther P Black, Patrick J McNamara, and Frank Romanelli. Development of guiding principles for a new era in graduate education. *American Journal of Pharmaceutical Education*, 83(2):7422, 2019. doi: 10.5688/ajpe7422.

[16] Denise Fleith. The role of creativity in graduate education according to students and professors. *Estudos de Psicologia (Campinas)*, 36, 01 2019. doi: 10.1590/1982-0275201936e180045.

[17] Andres Fortino, Qitong Zhong, Wei Chieh Huang, and Roy Lowrance. Application of text data mining to stem curriculum selection and development. In *2019 IEEE Integrated STEM Education Conference (ISEC)*, pages 354–361, 2019. doi: 10.1109/ISECon.2019.8882067.

[18] World Economic Forum. The future of jobs report 2023, 2023. URL https://www.weforum.org/reports/the-future-of-jobs-report-2023/. Accessed: 2024-07-06.

[19] G Gowsalya and M Kumar. Employability skill: A literature review. *International Journal of Advance Research in Computer Science and Management Studies*, 3(3), 2015.

[20] Maarten Grootendorst. Keybert: Minimal keyword extraction with bert., 2020. URL https://doi.org/10.5281/zenodo.4461265.

[21] Andrew Hanson, Carlo Salerno, Matt Sigelman, Mels de Zeeuw, and Stephen Moret. Talent disrupted: Underemployment, college graduates, and the way forward, February 2024. URL https://stradaeducation.org/wp-content/uploads/2024/02/Talent-Disrupted.pdf.

[22] John Jerrim. Do uk higher education students overestimate their starting salary? *Fiscal Studies*, 32(4):483–509, 2011. ISSN 01435671, 14755890. URL http://www.jstor.org/stable/24440182.

[23] Kornraphop Kawintiranon, Peerapon Vateekul, Atiwong Suchato, and Proadpran Punyabukkana. Understanding knowledge areas in curriculum through text mining from course materials. In *2016 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pages 161–168. IEEE, 2016.

[24] Henrik Kortum, Jonas Rebstadt, and Oliver Thomas. Dissection of ai job advertisements: A text mining-based analysis of employee skills in the disciplines computer vision and natural language processing. 2022.

[25] Dawn Lees. Graduate employability-literature review. 2002.

[26] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, 2020.

[27] Duo Li, Elizabeth Milonas, and Qiping Zhang. Content analysis of data science graduate programs in the u.s. *CUNY Academic Works*, 2023. URL `https://academicworks.cuny.edu/ny_pubs/798`.

[28] Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.

[29] Elena Lisá, Katarína Hennelová, and Denisa Newman. Comparison between employers' and students' expectations in respect of employability skills of university graduates. 05 2019.

[30] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.

[31] Mantas Lukauskas, Viktorija Šarkauskaitė, Vaida Pilinkienė, Alina Stundžienė, Andrius Grybauskas, and Jurgita Bruneckienė. Enhancing skills demand understanding through job ad segmentation using nlp and clustering techniques. *Applied sciences*, 13(10):6119, 2023.

[32] Imren Markes. A review of literature on employability skill needs in engineering. *European Journal of Engineering Education*, 31(6):637–650, 2006.

[33] Tomas Mikolov. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 3781, 2013.

[34] Amy Milsom and Julie Coughlin. Satisfaction with college major: A grounded theory study. *NACADA Journal*, 35:5–14, 11 2015. doi: 10.12930/NACADA-14-026.

[35] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

[36] William Muse and Iryna Muse. College selectivity, choice of major, and post-college earnings. *Journal of Economic Analysis*, 3:33–51, 08 2023. doi: 10.58567/jea03020003.

[37] Subbu M Nisha and V Rajasekaran. Employability skills: A review. *IUP Journal of Soft Skills*, 12(1):29–37, 2018.

[38] Council of Graduate Schools. Graduate education and the public good, April 2008.

[39] Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. Hierarchical transformers for long document classification. In *2019 IEEE automatic speech recognition and understanding workshop (ASRU)*, pages 838–844. IEEE, 2019.

[40] Julie R. Posselt and Eric Grodsky. Graduate education and social stratification. *Annual Review of Sociology*, 43(Volume 43, 2017):353–378, 2017. ISSN 1545-2115. doi: https://doi.org/10.1146/annurev-soc-081715-074324. URL https://www.annualreviews.org/content/journals/10.1146/annurev-soc-081715-074324.

[41] J. Puckett, Leila Hoteit, Sergei Perapechka, Ekaterina Loshkareva, and Gulnara Bikkulova. Fixing the Global Skills Mismatch, July 2020. URL https://www.bcg.com/publications/2020/fixing-global-skills-mismatch.

[42] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[43] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[44] Cort W Rudolph, Kristi N Lavigne, and Hannes Zacher. Career adaptability: A meta-analysis of relationships with measures of adaptivity, adapting responses, and adaptation results. *Journal of Vocational Behavior*, 98:17–34, 2017.

[45] Ceylan Sen, Zeynep Sonay Ay, and Seyit Ahmet Kiray. Stem skills in the 21st century education. *Research highlights in STEM education*, pages 81–101, 2018.

[46] Gitta Siekmann. What is stem? the need for unpacking its definitions and applications. *National Centre for Vocational Education Research (NCVER)*, 2016.

[47] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867, 2020.

[48] C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. ISSN 00029556. URL http://www.jstor.org/stable/1412159.

[49] University of Southern California, Viterbi School of Engineering. Employment outcomes – computer science, data science & cyber security, 2024. URL https://viterbigradadmission.usc.edu/employment-outcomes-csds/.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[51] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[52] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788, 2020.

[53] Cullen Watson. Jobspy. `https://github.com/Bunsly/JobSpy`, 2023. GitHub repository.

[54] Susima Samudrika Weligamage et al. Graduates' employability skills: Evidence from literature review. *Sri Lanka: University of Kelaniya*, 2009.

[55] Jason West. Validating curriculum development using text mining. *The Curriculum Journal*, 28(3):389–402, 2017.

[56] Matthew Wiswall and Basit Zafar. New approaches to understanding choice of major. *NBER Reporter*, (2):18–21, 2021. ISSN 0276-119X. URL `https://hdl.handle.net/10419/265465`.

[57] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

[58] Law Sheng Xun, Swapna Gottipati, and Venky Shankararaman. Text-mining approach for verifying alignment of information systems curriculum with industry skills. In *2015 International Conference on Information Technology Based Higher Education and Training (ITHET)*, pages 1–6. IEEE, 2015. doi: 10.1109/ITHET.2015.7217959.

[59] Mike Zhang. Computational job market analysis with natural language processing. *arXiv preprint arXiv:2404.18977*, 2024.