Automating Course Articulation

A Deep Metric Learning Framework Using Public Data

A Thesis submitted to the faculty of

San Francisco State University

In partial satisfaction of the

requirements for

the Degree

Master of Science

in

Data Science and Artificial Intelligence

by

Mark S. Kim

San Francisco, California

May 2025

Certification of Approval

I certify that I have read Automating Course Articulation: A Deep Metric Learning Framework Using Public Data by Mark S. Kim and that in my opinion this work meets the criteria for approving a thesis submitted in partial fulfillment of the requirement for the degree Master of Science at San Francisco State University.

Hui Yang, Ph.D

Professor

Thesis Committee Chair

Arno Puder, Ph.D

Professor

Anagha Kulkarni, Ph.D

Professor

Abstract

The manual process of determining course equivalency is a significant barrier to student mobility and educational equity, leading to substantial credit loss and delayed graduation that disproportionately harms underrepresented students. Previous automated approaches have been limited by a reliance on sensitive student enrollment records or the operational intractability and opacity of using large language models for direct classification.

This thesis introduces and validates a novel framework that overcomes these challenges by decoupling semantic representation from classification, using only publicly available course catalog data. The methodology leverages deep metric learning to fine-tune contextual embedding models on public course text. These specialized embeddings are then used to construct a novel composite distance vector, which serves as a rich feature set for training traditional machine learning classifiers.

Evaluated on a real-world dataset, the proposed framework achieves state-of-the-art accuracy, with F_1-scores exceeding 0.99. The result is a computationally efficient, scalable, and privacy-preserving solution that provides institutions with a practical tool to automate course articulation, reduce administrative burden, and foster a more equitable educational ecosystem.

Acknowledgments

I would like to express my deepest appreciation to my advisors, Professors Hui Yang, Arno Puder, and Anagha Kulkarni. Their patience, support, and confidence in me were invaluable throughout my graduate journey at SFSU. I also owe a special debt of gratitude to Professor Tao He, whose crucial suggestion to incorporate a global similarity metric with the embedding vectors significantly improved the model's classification performance.

This research was made possible through the contributions of many individuals. I'd like to thank the Program Pathways Mapper (PPM) team, which includes representatives from the Kern Community College District, the Foundation for California Community Colleges, and the California Community Colleges Chancellor's Office, for their partnership in providing the foundational data for this work. I'd like to recognize Natalie Yam, Parth Panchal, and Joanne Park for their assistance with data collection, compilation, and preliminary analysis.

This work was supported in part through the Platform for Open Learning, Academic Research, & Innovative Scientific computing (POLARIS) High-Performance Computing (HPC) cluster at San Francisco State University. We acknowledge these resources, services, and the support provided by the Academic Technology Systems Team.

On a personal note, I am immensely thankful for my dear friends who have been there for me through thick and thin, cheering me on whenever I felt I could not continue. To my good friend, Phil, I offer my deepest appreciation for his incredible generosity and support, which made navigating the financial challenges of graduate school possible. To Julie and

Bita, I am profoundly thankful for their invaluable guidance and perspective, which were instrumental to my personal growth and well-being throughout this journey. To my brother Nick, I am eternally grateful for his quiet strength and unwavering support during times when distance and understanding meant everything.

To all who stood by me with patience, kindness, and belief—I carry your support with deep gratitude.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

California's public higher education system is a titan of American academia, a complex, three-tiered structure comprising the University of California (UC), California State University (CSU), and the California Community Colleges (CCC) [13]. Collectively, these 149 colleges and universities serve nearly 2.9 million students, forming the largest public higher education system in the United States [13, 32, 7, 6]. A foundational principle of this system is the promise of student mobility, particularly the pathway from a two-year community college to a four-year university. However, the mechanism designed to facilitate this movement, the process of determining course equivalency, or articulation, is a formidable, largely manual process that creates significant barriers for students.

At the heart of this process is the Articulation System Stimulating Interinstitutional Student Transfer (ASSIST), the state's official public repository for articulation agreements [4]. While ASSIST provides a centralized platform for students and advisors to view established

equivalencies, it is fundamentally a display for agreements that are negotiated and updated manually by articulation officers at each individual campus [3]. Given the sheer number of institutions, the process of defining and maintaining these agreements is a task of bleak combinatorics, rendering it inefficient, slow, and inherently intractable [27]. This manual paradigm places a considerable burden on academic advisors and administrative staff, who must meticulously review course descriptions and syllabi to compare content, rigor, and learning outcomes [27]. The result is a system that struggles to keep pace with the needs of a vast and mobile student body, making California a critical case study for a problem that extends far beyond its borders.

The challenges exemplified by California's system are a microcosm of a systemic crisis in American higher education. The act of transferring between institutions has become a normative component of the modern student's academic journey. Data from the National Student Clearinghouse Research Center reveals that in the fall of 2023, transfer enrollment constituted 13.2% of all continuing and returning undergraduates [10]. This trend is not static; it represents a post-pandemic resurgence in student mobility, with transfer enrollment growing by 5.3% from Fall 2022 to Fall 2023 and an additional 4.4% in Fall 2024 [10, 11]. This mobile population is increasingly diverse, comprising not only students following traditional two-year to four-year pathways but also a substantial number of returning learners who have previously paused their education. Over half of these returning students opt to re-enroll at a new institution, underscoring the critical role of the transfer system in providing flexible pathways to degree completion [12].

The consequences of this systemic inefficiency are borne almost entirely by the students, manifesting in significant academic and financial setbacks. The most direct and damaging outcome is the loss of earned academic credit. A comprehensive 2017 report by the U.S. Government Accountability Office (GAO) estimated that students who transferred between 2004 and 2009 lost an average of 43% of their credits in the process [39]. This finding is echoed across numerous studies, with reports indicating that more than half of all transfer students lose at least some credits, and approximately one-fifth are forced to repeat courses for which they have already received a passing grade at a previous institution [1].

This loss of credit creates a cascade of negative consequences. It invariably leads to an increased time-to-degree, delaying graduation and entry into the workforce. Each repeated course also carries a financial cost, increasing the total tuition burden and potentially exhausting a student's eligibility for federal financial aid programs like Pell Grants and Direct Loans [39]. A process that is often undertaken to save money (for example, by starting at a less expensive community college) can paradoxically result in a greater overall financial commitment, trapping students in a cycle of additional coursework and debt [5].

The friction and frustration inherent in the transfer process also have a measurable impact on student persistence and graduation. Studies have shown that transfer students, as a group, tend to have lower retention and graduation rates than their peers who begin and end their studies at the same institution [31]. This issue transcends mere administrative inefficiency and becomes a critical matter of educational equity. Low-income students and students from historically underrepresented racial and ethnic groups are more likely to

begin their postsecondary journey at community colleges and rely on transfer pathways to attain a bachelor's degree [38]. The recent growth in transfer enrollment has been driven disproportionately by Black and Hispanic students [10]. Therefore, the barriers imposed by an inefficient articulation system such as credit loss, increased cost, and delayed graduation, disproportionately harm the very student populations that institutions are striving to support.

A clear and troubling feedback loop emerges from this analysis. The fundamentally manual and inefficient nature of course articulation is a direct cause of credit loss. This credit loss imposes a tangible academic and financial burden on students which falls most heavily on underrepresented and low-income students, who are a large and growing segment of the transfer population. This disproportionate impact, in turn, undermines institutional goals of improving student retention and closing persistent equity gaps in degree attainment. Thus, the seemingly low-level administrative task of determining course equivalency is revealed to be a significant driver of systemic inequity in higher education. Addressing this challenge through robust automation is not merely an operational optimization; it is a necessary intervention to foster a more equitable, efficient, and supportive educational ecosystem for all students.

1.1 Contributions

This research makes several key contributions to the fields of educational data mining and natural language processing, offering a practical and powerful solution to the long-standing challenge of course articulation.

- **A High-Accuracy, Automated Framework**: This thesis develops and validates a novel framework for determining course equivalency that achieves state-of-the-art accuracy, with F_1-scores exceeding 0.99 on a challenging real-world dataset. Crucially, it accomplishes this using only publicly available course catalog text, making it broadly applicable.

- **An Innovative Feature Engineering Technique**: It introduces a composite distance vector, Δ_c, that uniquely combines element-wise embedding differences with cosine similarity. This technique provides a richer input signal for classification and is shown to demonstrably improve the performance of downstream machine learning models, particularly linear classifiers.

- **A Computationally Efficient and Scalable Approach**: The research demonstrates that by decoupling semantic representation from classification, it is possible to harness the power of deep contextual embeddings without the high computational costs, API dependencies, and opaque nature of direct LLM-based classification. This makes the proposed solution more efficient, scalable, and practical for institutional

deployment.

- **A Privacy-Preserving Methodology**: By relying exclusively on public course descriptions, the proposed method circumvents the significant privacy, security, and data access challenges associated with techniques that require sensitive student enrollment records. This makes the framework more ethically sound and generalizable across any pair of institutions, regardless of their data-sharing agreements.

1.2 Thesis Roadmap

The remainder of this thesis is structured to provide a comprehensive account of this research.

- **Chapter 2: Background and Related Work** will provide a detailed in-depth survey of the landscape of student transfer automation and the evolution of technological interventions.

- **Chapter 3: Methodology** will offer a deep dive into the data collection and preparation processes, the specific embedding models evaluated, the construction of the feature vectors, and the theoretical underpinnings of the machine learning classifiers employed.

- **Chapter 4: Experimental Setup and Results** will detail the experimental design, the datasets used for training and validation, and a comprehensive analysis of the classification performance, including ablation studies and model comparisons.

- **Chapter 5: Discussion and Future Work** will interpret the results in a broader context, discuss the limitations of the current study, and outline promising avenues for future research, including the development of a full-scale course recommendation system and the exploration of fine-tuning techniques.

- **Chapter 6: Conclusion** will summarize the key findings of the thesis and reiterate the significance of its contributions to both academic research and the practical administration of higher education.

Chapter 2

Background and Related Work

The manifest inefficiencies and inequities of manual course articulation, exemplified by the challenges within California's vast system, have prompted a range of research efforts aimed at automating the process [25, 30, 26, 21, 40]. These technological interventions have evolved in sophistication, mirroring the broader advancements in natural language processing (NLP) and machine learning. A critical review of this literature reveals a clear trajectory from simple statistical methods to complex deep learning models, with each stage introducing new capabilities while also exposing new limitations. This evolution illuminates the path toward a more robust and scalable solution.

2.1 Keyword and Statistical Methods

The earliest attempts at automating course comparison relied on foundational text analysis techniques that, while computationally simple, lack semantic depth. The most basic systems are essentially search engines or databases that depend on exact keyword matching or pre-populated tables of known equivalencies [23]. These systems are inherently brittle; they cannot recognize semantic variations (e.g., equating "Introduction to Programming" with "Fundamentals of Computer Science I") and require continuous manual updates to remain relevant [36].

A more advanced statistical method, Term Frequency-Inverse Document Frequency (TF-IDF), improves upon keyword matching by vectorizing documents and weighting terms based on their importance. A term's frequency within a single document (TF) is balanced against its rarity across a collection of documents, or corpus (IDF) [2]. This allows the model to assign higher importance to distinctive terms (e.g., "calculus") and lower importance to common words (e.g., "the," "a," "is") [2]. TF-IDF has been a workhorse for information retrieval and has been applied to course similarity tasks [2]. However, the fundamental limitation of TF-IDF and other bag-of-words models is their complete lack of semantic understanding. They treat words as discrete, unrelated tokens and cannot grasp that "calculus" and "differentiation" are related concepts, nor can they distinguish between different meanings of the same word.

2.2 Static Semantic Representations

The development of word embeddings represented the first major leap toward a true semantic understanding of course content. Models like Word2Vec and GloVe, trained on vast text corpora, learn to represent words as dense vectors in a high-dimensional space, where words with similar meanings are positioned closer to one another. For example, the vectors for "car" and "automobile" would be near each other, while being distant from the vector for "planet." This innovation enabled a more nuanced comparison of texts than was possible with TF-IDF. In the context of educational data mining, these techniques have been applied to content-based course recommendation systems, typically by creating a single vector representation for a course description by averaging the vectors of all its constituent words [28].

Despite this advancement, these models produce static embeddings. Each word is assigned a single, fixed vector regardless of its context. This is a significant drawback, as it fails to account for polysemy: words with multiple meanings. For instance, the word "bank" would have the same vector in the phrases "river bank" and "bank account," despite their disparate meanings. Furthermore, the common practice of averaging all word vectors to create a document-level representation is a crude heuristic that can dilute or lose critical semantic information, especially in complex or lengthy descriptions.

2.3 Contextual Semantic Representations

The introduction of the transformer architecture, and specifically models like Bidirectional Encoder Representations from Transformers (BERT), revolutionized NLP by enabling the generation of contextual embeddings. In these models, the vector representation of a word is dynamically influenced by the words surrounding it in a sentence [14]. This allows the model to disambiguate word meanings and capture a much richer, more accurate semantic representation of the text. Architectures such as Sentence-BERT (SBERT) were subsequently developed to fine-tune these models specifically for the task of producing semantically meaningful embeddings for entire sentences or short paragraphs, which can then be efficiently compared using metrics like cosine similarity [33].

2.4 Direct LLM Classification

A more recent evolution in this domain involves the direct application of large-scale generative models, or Large Language Models (LLMs) like GPT-4 and Gemini, for classification tasks. The preliminary work for this thesis explored this very approach. Through sophisticated prompt engineering and in-context learning, these models can be instructed to perform pairwise comparisons of course descriptions and render a judgment on their equivalency.

While these experiments yielded promising accuracy, they also uncovered significant practical and theoretical limitations. The direct use of LLMs for this task is computationally expensive and inefficient, as it requires repeatedly sending full text descriptions to a model

API for every comparison. Performance is acutely sensitive to the exact phrasing of the prompt, necessitating a costly and time-consuming iterative tuning process. Most critically, the decision-making process of an LLM is a "black box," providing a categorical output (e.g. "equivalent") without a quantifiable similarity score or confidence level. This opacity makes it difficult to rank potential matches, set decision thresholds, or provide transparent justifications for the model's conclusions. This approach is also ill-suited for handling more complex articulation scenarios, such as one-to-many or many-to-many course mappings. A detailed review of our efforts are outlined in Chapter 3.1.

2.5 Enrollment-Based Approaches

It is essential to situate this research within the context of parallel efforts that leverage different data sources. A notable body of work has demonstrated that course similarity and prerequisite relationships can be predicted with high accuracy by analyzing student enrollment data. Models such as *course2vec* learn course embeddings not from their textual descriptions, but from the patterns of which courses students tend to take together [29]. The underlying principle is that courses frequently taken in the same semester or in sequence likely share a functional or topical relationship [29].

While this behavioral approach is powerful, its reliance on large-scale, proprietary institutional datasets of student records presents two major obstacles. First, it raises significant data privacy and security concerns, as it involves the analysis of sensitive student informa-

tion [37]. Second, it limits the scalability and generalizability of the solution. The model is only applicable at institutions that can provide access to such data, and it cannot be used to compare courses between two institutions that have no history of student transfer between them. This approach is therefore not a universal solution for the broader course articulation problem.

The evolution of these varied approaches reveals a fundamental trade-off: as models gain greater semantic power, they tend to become more computationally intensive, less interpretable, and more demanding of specialized or private data. The limitations of direct LLM classification (cost, opacity) and enrollment-based methods (data privacy, limited access) point toward the need for a new paradigm. An ideal solution should harness the semantic power of large pre-trained models without inheriting their operational burdens. This suggests that the next logical step is not simply a larger, more complex end-to-end model, but rather a more intelligent, hybrid framework. Such a framework would decouple the task of deep semantic representation from the task of final classification, allowing each component to be optimized for what it does best. This conceptual shift forms the central motivation for the methodology proposed in this thesis. Table 2.1 summarizes the primary methods for determining course transferability.

Table 2.1: Comparative Taxonomy of Course Equivalency Determination Methods

Approach	Key Characteristics	Data Source(s)	Semantic Capability	Strengths	Limitations
Manual Review	Human experts (advisors, faculty) compare syllabi descriptions.	Course Catalogs, Syllabi	High (Human-level)	Nuanced, context-aware, trusted by faculty.	Extremely slow, not scalable, subjective, prone to inconsistency.
Keyword/ TF-IDF	Bag-of-words representation, statistical term weighting.	Course Catalogs	None to Low	Simple, computationally cheap, easy to implement.	Fails to capture synonyms, context, or true semantic meaning
Static Embeddings (Word2Vec/ GloVe)	Pre-trained word vectors, often averaged for document representation.	Course Catalogs	Medium	Captures word-level semantics, better than TF-IDF.	Context-insensitive, averaging vectors loses information.
Enrollment-Based (e.g., course2vec)	Embeddings learned from student co-enrollment patterns.	Proprietary Student Records	High (Behavioral)	Captures functional relationships between courses, highly predictive.	Requires access to sensitive private data, not generalizable, privacy concerns.
Direct LLM Classification	End-to-end classification using prompt engineering.	Course Catalogs	Very High	High accuracy potential, understands complex language.	Computationally expensive, "black box" opacity, prompt sensitive, no quantifiable similarity score, risk of hallucinations.
Proposed Method (Embeddings + ML)	Deep contextual embeddings as features for traditional classifiers.	Course Catalogs	Very High	State-of-the-art accuracy, computationally efficient, quantifiable, uses public data only.	Relies on the quality of the pre-trained embedding model.

Chapter 3

Methodology

This chapter delineates the comprehensive and systematic methodology employed to develop and evaluate a sophisticated classification pipeline capable of determining course equivalency from domain-specific textual data. The methodological framework presented here is the result of an evolutionary research process that progressed through two distinct phases. The investigation commenced with an initial, exploratory phase to assess the feasibility of using Large Language Models (LLMs) as end-to-end classifiers for the task. This direct approach was conceived as an initial benchmark, leveraging a small, manually curated dataset while a larger, more comprehensive dataset was anticipated but not yet available.

In this first phase, the LLM was treated as a holistic reasoning engine, responsible for the entire classification process from raw text input to a final equivalency judgment. The findings from this stage were critical; they demonstrated the potential of modern LLMs but also highlighted significant limitations, including high sensitivity to prompt engineering

and the absence of fine-grained similarity metrics for nuanced comparisons. These challenges fundamentally shaped the subsequent research, motivating the development of a more robust and scalable framework.

Consequently, the primary focus of this thesis is a more advanced, decoupled methodology designed specifically to overcome the drawbacks of the direct classification approach. The foundational principle of this final pipeline is to utilize state-of-the-art deep embedding models not as end-to-end classifiers, but as highly sophisticated feature extraction engines. In this paradigm, each course description is converted into a high-dimensional numerical vector that encapsulates its rich semantic content. By pre-processing and storing these embeddings, the framework eliminates the need to repeatedly process raw text through costly APIs, drastically reducing computational overhead and making the system inherently more scalable and suitable for real-world applications.

The narrative of this chapter follows this logical and chronological evolution, detailing the data acquisition, model architectures, training protocols, and evaluation frameworks for each stage of the research. We begin by examining the initial direct LLM classification approach, its implementation, and the key findings that paved the way for the more advanced methods that followed.

3.1 Initial Approach: Direct LLM Classification

The research commenced with an exploratory phase designed to determine the feasibility of leveraging Large Language Models (LLMs) as end-to-end classifiers for the course equivalency task. This direct approach was a pragmatic first step, conceived as an initial benchmark to establish a baseline for performance using a small, manually curated dataset. At this stage of the research, a larger, more comprehensive dataset from the Program Pathways Mapper (PPM) was anticipated but not yet available, making a focused, smaller-scale investigation the most logical starting point. This methodology treated the LLM as a holistic reasoning engine, tasked with performing the entire classification from raw text input to a final equivalency judgment without intermediate feature engineering. The findings from this initial stage were critical, as they highlighted both the potential and the inherent limitations of direct LLM classification, thereby fundamentally shaping the development of the more complex, decoupled pipeline that followed.

Initial Data Corpus and Pre-processing

The dataset for this initial evaluation was constructed to represent a challenging, real-world scenario using publicly available data. The process began by identifying five required lower-division courses for the Computer Science major at San Francisco State University (SFSU). Using ASSIST, articulation agreements were found for these courses across 63 different California public colleges and universities. The raw course data was then manually

collected from the online course catalogs of each respective college. This data was consisted of the full, unmodified text including department codes, course numbers, titles, descriptions, and all associated metadata such as prerequisites, unit counts, and grading options. This approach was deliberately chosen to ensure that the analysis could compare the effectiveness of classification using the complete raw text versus more structured, extracted information.

The initial dataset consisted of 228 equivalent course pairs based on the articulation agreements. To create a more robust dataset for binary classification, this set was expanded by assuming symmetry and transitivity for course equivalency, which generated a total of 5,660 equivalent pairs. An equivalent number of non-equivalent pairs was then generated by randomly pairing courses from different subjects. From this expanded corpus of over 11,000 pairs, a final stratified random sample of 400 pairs (200 equivalent and 200 non-equivalent) was created to serve as the evaluation set for the models.

Model Selection and Prompt Engineering

An initial review of various LLMs was conducted to assess their ability to reliably generate structured data from the raw course descriptions. While many open-source and proprietary models were tested, Google's PaLM2 and its successor, Gemini Pro v1.0, were ultimately selected for this phase of the research. This decision was based on their accessibility via a free-tier API and, most importantly, their consistent ability to produce well-formatted, structured data from the unprocessed text.

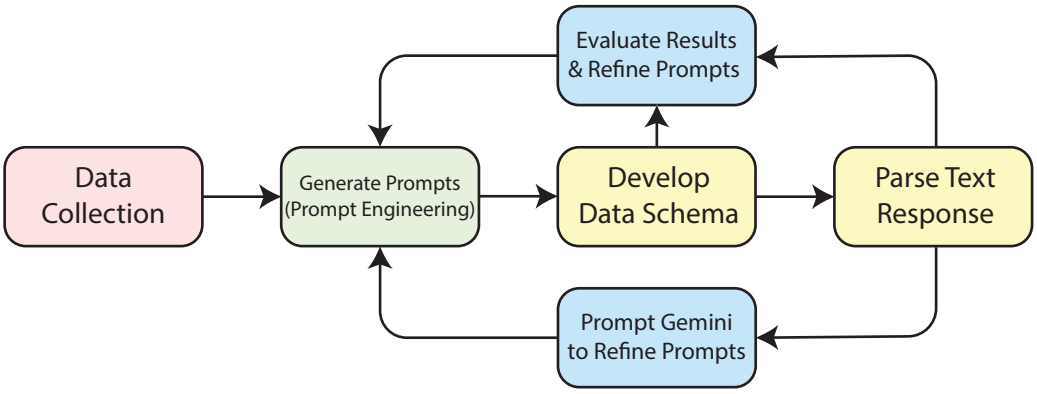


Figure 3.1: Prompt Engineering Process

A systematic, iterative prompt engineering process, illustrated by Figure 3.1, was employed to develop effective prompts for both the extraction of structured data (like course topics) and the final equivalency classification. This process involved starting with simple prompts and gradually refining them based on established design principles from natural language processing research and community guides [41, 22, 34]. The final prompts for data extraction were highly structured, consisting of five parts: a preamble summarizing the task, the raw course data, specific formatting instructions, a JSON model schema defining the desired output, and a postamble with additional clarifying instructions. Despite this careful refinement, the structured data extraction process, particularly for deducing course topics, remained a significant challenge and was prone to occasional contextual errors.

Classification and Evaluation Findings

The core of this initial research phase involved pairwise course assessments using Gemini Pro with the previously developed prompts. The evaluation was structured to explore

different classification scenarios to gain a comprehensive understanding of the model's capabilities:

- **Input Data**: Evaluations were conducted using both the full, raw course description text and a structured format using only the course discipline and the topics extracted by the LLM.
- **Classification Tasks**: Three distinct classification tasks were designed to simulate the complexities of human decision-making: a standard binary classification ("equivalent" vs. "non-equivalent"), a three-class model that added an "unsure" category for ambiguous cases, and a four-class model that also included an "insufficient data" category.

The evaluation yielded promising results, with the model achieving a notable 90.5% accuracy on the binary classification task when using the full raw text. This result confirmed that the unprocessed text was a superior input format, as performance degraded significantly when using only the structured topics, possibly due to the minor but persistent errors introduced during the LLM's extraction process. Furthermore, the experiments that introduced "unsure" and "insufficient data" categories also proved valuable. While these classifications cannot be measured against a ground truth, they successfully simulated the ambiguity a human advisor might face and demonstrated a method for isolating cases that would require further manual review.

Crucially, this initial investigation revealed several fundamental weaknesses in the direct LLM approach, making it unsuitable for a scalable, reliable production system. The method provided no confidence scores for its decisions, leaving the end-user with a "black box" judgment without any measure of certainty. It also lacked a granular similarity metric, preventing the system from ranking different course pairs or identifying courses that are "close but not equivalent". The approach was also computationally expensive due to the significant token input required for every single pairwise comparison. Finally, the prompts were found to be highly model-specific, which would lead to significant rework if the underlying model were ever updated or changed. These limitations directly motivated a pivot in methodology towards the decoupled, embedding-based pipeline. This new approach was first developed and prototyped using the initial, manually-curated dataset. The following sections will now detail this final, more robust methodology, beginning with the larger and more comprehensive data corpus upon which it was ultimately trained and validated.

3.2 The PPM Corpus and Data Preparation

The limitations identified in the initial study necessitated not only a more sophisticated methodology but also a larger, more comprehensive dataset to robustly train and evaluate it. This required data corpus was procured at a later stage of the research in partnership with the Program Pathways Mapper (PPM). The acquisition of this dataset was a critical step that enabled the full-scale implementation and validation of the decoupled pipeline. This

section characterizes this final PPM data corpus and details the preparation steps undertaken before its use in the model fine-tuning and evaluation stages described later in this chapter.

Dataset Characterization and Partitioning

The corpus provided by the PPM initially contained 2,217 courses, each labeled with a Course Identification Numbering System (C-ID) code that serves as the ground truth for course equivalency. To prepare this data for a robust, stratified partitioning, a critical filtering step was applied first. All C-ID classes with fewer than four associated courses were removed from the dataset. This step was essential to guarantee that after splitting the data, both the training and subsequent test sets would contain enough examples to form at least one equivalent course pair for every class, a necessary condition for the fine-tuning process. This filtering resulted in a final, clean corpus of 2,157 courses distributed across 157 distinct C-ID classes

To ensure a rigorous and unbiased evaluation of the final models, this final dataset was partitioned into two distinct, non-overlapping subsets: a training set and a test set. A stratified 50/50 split was employed, using the C-ID code as the stratification key. This resulted in a training set of 1078 courses and a test set of 1079 courses. The stratification ensures that each of the 157 classes is represented in both subsets, with each class containing between 2 and 33 equivalent courses per set. The test set is held in reserve and used only once for the final, conclusive evaluation of the optimized classification pipeline, providing an

honest estimate of the model's generalization performance on unseen data.

The training set serves a dual purpose in this methodology, a strategy made possible by the different data formats required by the training and validation procedures. For the fine-tuning process itself, the training set is used to generate (anchor, positive, negative) triplets as required by the Triplet Loss objective. For the validation within the fine-tuning process, this same training set is utilized to generate binary pairs of "equivalent" and "non-equivalent" course pairs. This validation set is then used with the `BinaryClassificationEvaluator` to monitor model performance during training and select the best model checkpoint. This approach allows for robust validation without data leakage, as the model is evaluated on our primary task (binary pair classification) rather than the one it is directly optimizing for (triplet distance minimization).

Input Document Normalization

In a departure from conventional NLP pipelines, this research deliberately eschewed standard text pre-processing techniques such as lowercasing, stop-word removal, or the stripping of special characters. This decision was made to more accurately simulate a real-world use case where input data, such as course information copied directly from a university website, may be imperfectly formatted or contain extraneous characters. The methodology, therefore, relies on the inherent semantic power and robustness of modern transformer-based embedding models to interpret and handle this "raw" text.

Instead of cleaning the text, a normalization step was performed to create a consistent, structured input document for the embedding models. A new field, "Formatted Course Info," was generated for each course by concatenating four key pieces of information: the department name, the department course number, the course title, and the full course description. For example, a course might be transformed into the following single string:

This concatenated string serves as the single document representation for each course and is the direct input for the document embedding process. This approach ensures that all relevant textual context is preserved in a standardized format before being converted into a numerical vector.

Data Structuring for Fine-Tuning

While the previous subsection defined the content of each input document, this section details how those documents are structured and batched to meet the specific requirements of the chosen fine-tuning objective. The family of triplet loss functions used in this research operates on a conceptual structure of (anchor, positive, negative) triplets [33, 18]. A key advantage of the batch-based triplet loss functions is that they do not require the manual, pre-generation of these triplets; instead, the loss function intelligently forms the necessary triplets on-the-fly from within each mini-batch of (text, label) pairs. [18].

A crucial constraint imposed by triplet loss functions, particularly those available in the sentence-transformers library, is that the training data must contain a minimum of two

examples for each class label [19]. This is not a superficial requirement but a fundamental necessity for the loss computation to be valid. The formation of an (anchor, positive) pair requires selecting two distinct samples from the same class [19]. If a class is represented by only one sample, it can never form a positive pair and can only contribute to (anchor, negative) pairings.

This constraint has a direct and significant implication for the data loading and batching strategy. A simple random sampling approach could easily result in mini-batches that lack the necessary class diversity, containing only one or even zero instances of certain classes. In such a scenario, no valid triplets could be formed for the anchors of those underrepresented classes within that batch, leading to an inefficient and potentially biased training process. To mitigate this, a more sophisticated data loading mechanism was required. Rather than implementing a fully custom solution, this research leveraged a specialized, built-in feature of the sentence-transformers library: the `GroupByLabelBatchSampler`. This sampler, activated by setting the `batch_sampler` argument to `BatchSamplers.GROUP_BY_LABEL` during trainer configuration, is specifically designed for use with triplet loss functions [20]. It addresses the sampling challenge by ensuring that each mini-batch is constructed by grouping samples with the same label, thereby guaranteeing that every batch contains the necessary class diversity to form valid and informative triplets for all anchors. This demonstrates a key principle of the methodology: the data preparation and loading stage is not an independent preliminary step but is intrinsically linked to and constrained by the fine-tuning objective.

3.3 Embedding Model Selection and Fine-Tuning

A central hypothesis of this research is that a generic, pre-trained deep embedding model can be adapted to produce highly specialized and semantically rich embeddings for the specific domain of the data corpus. These bespoke embeddings are expected to provide a more discriminative feature representation for downstream classification tasks compared to off-the-shelf embeddings. This section details the architecture, learning objective, and training protocol used to achieve this adaptation through a process of deep metric learning.

Model Selection

The selection of an appropriate embedding model is a critical first step that influences the entire downstream pipeline. Rather than committing to a single model, this research began with a broad preliminary analysis of a variety of open-source embedding models to identify strong candidates for a more in-depth, comparative study. The initial models reviewed, summarized in Table 3.1, spanned a wide range of parameter sizes and characteristics.

To screen these models efficiently, the simple but effective cosine similarity accuracy metric was used: for a given anchor course, a model was considered correct if the cosine similarity to an equivalent course was greater than the similarity to a non-equivalent course. Sufficient to narrow down the field without the computational expense of a full classification pipeline for every single model, this method allowed for the rapid elimination of poorly performing models. This initial screening revealed that while performance varied, many

Table 3.1: Initial Embedding Model Review

Model Name	Rank*	Params†	Dims	Acc
GIST-small-Embedding-v0	41	33	384	0.9759
bge-small-en-v1.5	47	33	384	0.9670
GIST-Embedding-v0	33	109	768	0.9768
bge-base-en-v1.5	35	109	768	0.9732
gte-base-en-v1.5	31	137	768	0.9732
mxbai-embed-large-v1	24	335	1024	0.9759
gte-large-en-v1.5	21	434	1024	0.9777
multilingual-e5-large-inst	34	560	514	0.9670
stella_en_1.5B_v5	3	1543	8192	0.9857
SFR-Embedding-2_R	4	7111	4096	**0.9839**
Agte-Qwen2-7B-instruct	5	7613	3584	0.9804
nvidia/NV-Embed-v2	1	7851	4096	0.9831

* Huggingface Overall Leaderboard Rank

† in Millions

models achieved high accuracy, with a sample mean of 0.9767 and a standard deviation of 0.00605. Based on these results and a desire to evaluate a representative spectrum of model sizes, three models were selected for the primary analysis:

- `BAAI/bge-small-en-v1.5` (BGE): Representing a high-performing small model;
- `avsolatorio/GIST-Embedding-v0` (GIST): Representing a medium-sized model; and
- `nvidia/NV-Embed-v2` (NVE): Representing a large-scale model.

At a later stage of the research, `Salesforce/SFR-Embedding-2_R` (SFR) was also included for additional comparison due to its strong performance on public leaderboards. The foundational architecture for all these models is the transformer, which allows them to generate rich, contextual embeddings. By employing a pooling layer (typically mean-pooling),

these models produce a single, fixed-size vector for each input document, making them highly suitable for feature extraction.

Metric Learning with Batch Triplet Loss Functions

To determine if performance could be further improved, BGE was subjected to a fine-tuning process using a metric learning approach. Metric learning aims to learn an embedding space where the geometric distance between samples corresponds to their semantic similarity. One of the central hypotheses of our research is that a generic, pre-trained model can be adapted to produce highly specialized and semantically rich embeddings specifically for the domain of course descriptions, which could perform better than strong off-the-shelf models.

Theoretical Foundation of Triplet Loss

The concept of Triplet Loss was first introduced by Yu, et al. in the context of face recognition and has since been widely applied to supervised similarity learning [42]. It operates on the (Anchor, Positive, Negative) triplets described in Section 3.2. The fundamental goal of Triplet Loss is to train the embedding function, $f(x)$, to map inputs into a vector space where the distance between an anchor sample (A) and a positive sample (P) from the same class is smaller than the distance between the anchor and a negative sample (N) from a different class. To prevent the model from collapsing all embeddings to a single point and to ensure a meaningful separation, the loss function enforces a margin, α. The distance

between the anchor-positive pair must be smaller than the distance to the anchor-negative pair by at least this margin.

The mathematical formulation of the Triplet Loss is given by:

$$L(A, P, N) = \max(d(f(A), f(P)) - d(f(A), f(N)) + \alpha, 0).$$

Here, d represents a distance metric, which in this study is the Euclidean distance. The $\max(d(f(A), f(P)) - d(f(A), f(N)) + \alpha, 0)$ component ensures that the loss is only incurred for triplets that violate the margin constraint. If the negative sample is already further from the anchor than the positive sample by at least the margin, the loss for that triplet is zero, and no weight update is performed.

In the context of this research, metric learning, and more specifically Triplet Loss, provides a more powerful and nuanced objective than simple classification. This directly addresses one of the key limitations of the direct LLM classification approach, which expressly lacked this metric for comparing different course pairs.

Online Triplet Mining and Batch-based Losses

A naive implementation of Triplet Loss would form all possible triplets from the training data, an approach known as "offline" mining. This is computationally infeasible for large datasets and highly inefficient, as the vast majority of triplets are "easy" and provide no useful learning signal. A more effective approach is "online" mining, where informative triplets are selected on-the-fly from within each mini-batch of training data. Within this

paradigm, triplets can be categorized by their difficulty :

- **Easy Triplets**: These triplets already satisfy the margin constraint, i.e., $d(A, P)+\alpha < d(A, N)$. They result in zero loss and do not contribute to learning.
- **Semi-Hard Triplets**: In these triplets, the negative sample is more distant than the positive sample but still lies within the margin, i.e., $d(A, P) < d(A, N) < d(A, P) + \alpha$. These triplets have a positive loss and provide a useful, stable gradient for training.
- **Hard Triplets**: These are the most challenging cases, where the negative sample is closer to the anchor than the positive sample, i.e., $d(A, N) < d(A, P)$. These triplets produce the largest loss values and provide the strongest learning signal, forcing the model to learn fine-grained distinctions.

Empirical Evaluation of Batch Triplet Loss Functions

Recognizing that different triplet mining strategies can significantly impact model performance, this research empirically evaluated all four primary batch-based triplet loss implementations available in the sentence-transformers library to identify the optimal strategy for this specific dataset and task. This approach ensures that the chosen loss function is best suited to the data's characteristics, rather than relying on a single, pre-selected method. The evaluated loss functions were:

- **BatchAllTripletLoss**: This function computes the loss for all valid triplets that can be formed within a given batch. While this approach is comprehensive, it can be

computationally intensive, and the learning signal can be diluted by the high number of "easy" triplets that contribute zero to the loss.

- **BatchSemiHardTripletLoss**: This function focuses the training effort by considering only semi-hard triplets for each anchor. This is a common strategy that provides stable training but can sometimes lead to slower convergence as it ignores the most challenging "hard" examples that often provide the strongest learning signal.

- **BatchHardTripletLoss**: This function implements a more aggressive mining strategy. For each anchor sample in a mini-batch, it identifies the hardest positive (the positive sample that is furthest away) and the hardest negative (the negative sample that is closest). The loss is then computed on these most challenging pairs. This can accelerate convergence but is also known to be "temperamental," potentially leading to a noisy and difficult optimization landscape.

- **BatchHardSoftMarginTripletLoss**: This function is a variation of `BatchHardTripletLoss` that also focuses on the hardest positive and negative samples but does not require a manually specified margin parameter. This simplifies hyperparameter tuning while still leveraging an aggressive mining strategy.

Because different mining strategies can significantly impact final model performance, all four implementations were empirically tested to identify the optimal strategy for this specific dataset and task. The final model used for feature generation was trained with the loss

function that demonstrated the best performance on the validation set during this empirical evaluation.

Optimization and Training Protocol

The successful fine-tuning of a model with a challenging objective like `BatchHardTripletLoss` is critically dependent on the choice of optimizer and learning rate schedule. The components selected for this research, AdamW and Cosine Annealing with Warm Restarts, were not chosen in isolation but as parts of a cohesive, synergistic framework designed to ensure stable and effective learning.

Optimizer Selection: AdamW

The optimization of the model's weights was performed using the AdamW optimizer. The foundation of AdamW is the Adam (Adaptive Moment Estimation) optimizer, which has become a de facto standard in deep learning due to its computational efficiency and effectiveness. Adam combines the benefits of two other optimization algorithms: the momentum method, which helps accelerate gradient descent in the relevant direction, and RMSprop, which adapts the learning rate for each parameter based on the magnitude of past gradients. This per-parameter adaptive learning rate makes Adam particularly well-suited for the noisy gradients that can arise from `BatchHardTripletLoss`.

However, the standard implementation of Adam has a subtle flaw in how it handles L2 regularization, also known as weight decay. In standard Adam, the weight decay term

is coupled with the gradient update itself. This means that the regularization effect is influenced by the magnitude of the gradient, leading to an inconsistent application of weight decay. For parameters with large gradients, the effective weight decay is reduced, which can hinder proper regularization and lead to poorer generalization [24].

AdamW, proposed by Loshchilov and Hutter, corrects this flaw by decoupling the weight decay from the gradient update step. The adaptive moment updates are performed as in standard Adam, but the weight decay is applied as a separate, final step, directly modifying the weights. This decoupled approach ensures that weight decay acts as a true regularizer, penalizing large weights uniformly without interfering with the optimizer's adaptive learning mechanism. For a complex and potentially unstable training process like the one employed here, the enhanced generalization and training stability offered by AdamW make it a superior choice over standard Adam [24]. It provides the necessary regularization to prevent overfitting on the "hard" examples mined by the loss function, while preserving the adaptive learning that helps navigate the complex loss surface.

Learning Rate Schedule: Cosine Annealing with Warm Restarts

Static or linear learning rates are often suboptimal for training deep neural networks. A global learning rate schedule, which dynamically adjusts the learning rate during training, can significantly improve convergence and final model performance, even when using an adaptive gradient algorithm such as AdamW [24]. For this research, the `CosineAnnealingWarmRestarts` schedule was employed. This schedule combines two

powerful concepts: cosine annealing and warm restarts. Cosine annealing smoothly decays the learning rate from an initial maximum value, $\nu_{\max}$, to a minimum value, $\nu_{\min}$, following the shape of a cosine curve. The learning rate ν_t at a given epoch T_{cur} within a cycle of length T_i is calculated as:

$$\nu_t = \nu_{\min} + \frac{1}{2}(\nu_{\max} - \nu_{\min})\left(1 + \cos\left(\frac{T_{\text{cur}}}{T_{\max}}\pi\right)\right)$$

This smooth, gradual decay is often more effective than abrupt step-wise decays, allowing the model to fine-tune its parameters more carefully as it approaches a minimum in the loss landscape [16]. The warm restarts technique introduces periodic "restarts" into the training process. After a specified number of epochs, T_0, the learning rate is abruptly reset to its initial maximum value, $\nu_{\max}$, and the cosine decay cycle begins anew. The length of subsequent cycles can be progressively increased by a multiplicative factor, T_{mult}. For example, if $T_0 = 10$ and $T_{\text{mult}} = 2$, the restarts would occur after 10, 30 (10+20), 70 (10+20+40), and so on, epochs.

The combination of these two techniques forms a powerful optimization strategy. The aggressive nature of `BatchHardTripletLoss` creates a complex loss surface with many sharp, suboptimal local minima. A standard decay schedule might cause the optimizer to converge into one of these minima and become stuck. The "warm restarts" provide a crucial escape mechanism. When the model's performance on the validation set begins to plateau, suggesting convergence into a local minimum, the sudden increase in the learning rate effectively "kicks" the optimizer out of that basin, allowing it to explore other regions of the loss land-

scape. The subsequent smooth cosine decay then enables the optimizer to carefully descend into any new, potentially broader and deeper, minimum it discovers.

This entire fine-tuning framework (`BatchHardTripletLoss`, `AdamW`, and `CosineAnnealingWarmRestarts`) represents a carefully orchestrated system. The loss function provides a powerful, albeit challenging, learning signal. The optimizer provides a stable, well-regularized mechanism for applying weight updates based on that signal. The learning rate schedule provides a global exploration strategy to guide the optimizer across the complex landscape, preventing premature convergence and increasing the likelihood of finding a high-quality solution. This synergistic design is critical for successfully training a highly discriminative embedding model under these demanding conditions.

3.4 Generating Course Embeddings

Upon completion of the model selection and fine-tuning processes, the next stage of the methodology involves using the selected models to transform the entire text corpus into structured numerical formats suitable for a comparative analysis of downstream machine learning classifiers.

This process was executed by loading each selected embedding model and operating it in inference mode. The "Formatted Course Info" text for each course in the training and test sets was individually fed into each model to generate a fixed length embedding vector. This procedure was performed for the models summarized in Table 3.2.

Table 3.2: Embedding Models & PCA Explained Variance

Model Name	Type Used*	# of Parameters (in Millions)	Embedding Dimensions	Explained Variance (# of PCs) 70%	80%	90%
BAAI/bge-small-en-v1.5	OTS & FT	33	384	28	45	76
avsolatorio/GIST-Embedding-v0	OTS & FT	109	768	23	40	73
nvidia/NV-Embed-v2	OTS	7851	4096	20	37	73
Salesforce/SFR-Embedding-2_R	OTS	7111	4096	20	37	73

* OTS: Off-The-Shelf; FT: Fine-Tuned

The outcome of this procedure is the creation of multiple distinct sets of numerical matrices (one for the training set and one for the test set per embedding model). In each matrix, every row corresponds to a specific text sample from the original corpus, and the columns represent the dimensions of that model's embedding space. These matrices, along with their corresponding class labels, constitute the final feature sets that serve as the direct input for the suite of classification models evaluated in the next section. This step marks the critical transition from the unstructured domain of natural language processing to the structured domain of tabular data analysis, upon which the final classification experiments are built.

3.5 Feature Vector Construction

The high-dimensional embedding vectors generated in the previous step, while semantically rich, are not the final features used for classification. To prepare the data for the downstream classifiers, a two-stage feature engineering pipeline was executed. This pipeline first applies various dimensionality reduction techniques to the embeddings and then con-

structs pairwise difference vectors from these reduced (and original) embeddings to explicitly represent the relationship between two courses for the equivalency classification task.

Dimensionality Reduction

The high-dimensional vectors produced by modern embedding models, while semantically rich, can present challenges for downstream machine learning algorithms, a phenomenon often referred to as the "curse of dimensionality." In this high-dimensional space, the data becomes extremely sparse, which can make it more difficult for some algorithms to discern meaningful patterns and can increase the risk of the model overfitting to noise in the training data. To address these potential issues, a systematic process of dimensionality reduction was applied to the embeddings generated by each model.

This investigation was motivated by several potential benefits. First, by reducing the number of features, we can sometimes improve a model's ability to generalize to unseen data. This is achieved by removing redundant or noisy dimensions, forcing the classifier to learn from the most salient and robust features in the data. Second, a reduction in dimensionality leads to a direct reduction in computational complexity. Fewer dimensions mean that the downstream classifiers require less memory and can be trained and evaluated more quickly, which is a significant practical advantage for both experimentation and potential production deployment. Together, these benefits serve to mitigate the curse of dimensionality, making the classification task more tractable.

To ensure the integrity of the evaluation and prevent any form of data leakage, the reduction models were governed by a strict protocol. Using a dedicated EmbeddingReducer class, each reduction technique was fit exclusively on the training data. The same fitted model was then used to transform both the training and the held-out test sets. This methodology guarantees that no information from the test set influences the parameters of the reduction models, ensuring that the final evaluation remains unbiased.

A comprehensive exploration was conducted to determine the optimal dimensionality for the task. This included using Principal Component Analysis (PCA) to reduce the vectors to the number of components required to explain 70%, 80%, and 90% of the original variance. Additionally, to test performance in very low-dimensional spaces, the embeddings were reduced to a static 4 and 7 dimensions using multiple techniques, including PCA, t-SNE, and PaCMAP. The outcome of this stage was a suite of new training and test matrices for each base embedding model, each corresponding to a unique combination of a reduction technique and a target dimensionality, ready for the next step of feature vector construction.

Global and Local Distance Vector

The ultimate goal of this research is to classify pairs of courses as either "equivalent" or "not equivalent." This requires input features that represent the relationship between two courses, not just the characteristics of a single one. Our preliminary analysis showed that relying on a single, holistic metric like cosine similarity was insufficient for establishing a clear

and reliable decision boundary. To overcome this, we designed a composite feature vector that provides a richer, more discriminative representation of a course pair's relationship.

The feature vector, denoted as Δ_c, is constructed by concatenating the element-wise difference of the two course embedding vectors ($\mathbf{A}$ and $\mathbf{B}$) with their cosine similarity. The formal definition is as follows:

$$\Delta_c = \left(a_1 - b_1, \ldots, a_k - b_k, \frac{\mathbf{A} \cdot \mathbf{B}}{\| \mathbf{A} \| \| \mathbf{B} \|} \right)$$

where $\mathbf{A} = (a_1, \ldots, a_k)$ and $\mathbf{B} = (b_1, \ldots, b_k)$ are the k-dimensional embedding vectors for the two courses. This design is powerful because it provides the subsequent classifier with two distinct types of information simultaneously. The element-wise difference captures granular, dimension-specific (local) disparities between the two semantic representations, while the cosine similarity provides a single, normalized measure of their overall (global) alignment in the vector space. This composite vector creates a much more informative and discriminative feature set than a single similarity score alone.

To generate the data for our models, a `CoursePairGenerator` class was implemented. This utility systematically created the training and test datasets by first assembling positive pairs (courses with the same C-ID label) and negative pairs (courses with different C-ID labels). For each of these pairs, it then computed the composite feature vector Δ_c, resulting in a highly informative and discriminative feature set ready for classification.

This procedure was applied to every set of embeddings (e.g., BGE-original, BGE-PCA4-reduced, GIST-original, etc.), resulting in a comprehensive suite of training and testing

datasets. Each dataset corresponds to a unique combination of an embedding model and a dimensionality reduction technique, ready for evaluation by the downstream classifiers detailed in the next section.

3.6 Classification Models

To determine the most effective method for classifying the generated pairwise feature vectors, a broad suite of machine learning algorithms was evaluated. This process began with a comprehensive initial evaluation of eight different models to understand which algorithmic families were best suited to the data. Based on those preliminary results, a smaller, more focused set of high-performing models was selected for the final, in-depth analysis.

Initial Evaluation

The initial evaluation included a diverse set of classifiers, each chosen to test a different hypothesis about the structure of the feature space:

- **Linear Models (Logistic Regression, Ridge, Lasso)**: These models were used to establish a baseline and determine the degree of linear separability of the data. Their simplicity and interpretability make them excellent for understanding the foundational difficulty of the classification task.

- **Instance-Based Model (k-Nearest Neighbors)**: KNN was included to probe the

local structure of the feature space. Its performance indicates whether courses with the same equivalency status form dense, localized clusters.

- **Kernel-Based Model (Support Vector Machine)**: An SVM with a non-linear kernel was used to test for complex, non-linear decision boundaries that linear models cannot capture.

- **Ensemble Model (Random Forest)**: Random Forest was chosen for its robustness to overfitting and its ability to capture complex feature interactions by combining the predictions of many decision trees.

- **Probabilistic Models (LDA and QDA)**: Linear and Quadratic Discriminant Analysis were used to test assumptions about the geometric distribution of the data, specifically whether the classes share a common covariance (LDA) or have unique ones (QDA).

Final Classifier Selection

Based on the preliminary results from the comprehensive evaluation, four models were selected for the final analysis due to their consistently strong performance: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Random Forest (RF), and XGBoost. XGBoost, a powerful gradient boosting implementation, was added at this stage to include a state-of-the-art boosting algorithm known for its high performance on structured data.

These models represent the most promising approaches for this classification task, covering instance-based, maximal-margin, and advanced ensemble methods.

3.7 Multi-Stage Evaluation

A rigorous, multi-stage evaluation framework was designed to systematically narrow down the optimal combination of embedding models, dimensionality reduction techniques, and classifiers.

Stage 1: Preliminary Embedding Model Selection

The process began with a broad analysis of twelve open-source embedding models to identify strong candidates for more intensive study. This initial screening was performed on a small, manually curated dataset. The evaluation metric was a simple accuracy score based on cosine similarity: for a given course, a model was scored as correct if an equivalent course had a higher cosine similarity than a non-equivalent course. This efficient, low-cost evaluation allowed for the rapid elimination of poorly performing models, resulting in the selection of three top candidates representing a range of sizes: BAAI/bge-small-en-v1.5 (BGE), avsolatorio/GIST-Embedding-v0 (GIST), and nvidia/NV-Embed-v2 (NVE).

Stage 2: Comprehensive Classifier and Reducer Evaluation

Using the three selected embedding models and the initial small dataset, a comprehensive GridSearchCV was conducted. This stage systematically tested each embedding model in combination with multiple dimensionality reduction techniques (PCA, t-SNE, and PaCMAP at 4 and 7 dimensions, as well as with 70%, 80%, and 90% explained variance for PCA) and the full suite of eight initial classifiers (Logistic Regression, Ridge, Lasso, KNN, SVM, Random Forest, LDA, and QDA). The goal of this exhaustive search was to identify the most effective downstream classifiers and to understand the impact of dimensionality reduction on performance.

Stage 3: Evaluation During Fine-Tuning

With the larger, more robust PPM dataset, the two most promising non-proprietary models (BAAI/bge-small-en-v1.5 and avsolatorio/GIST-Embedding-v0) were fine-tuned to specialize them for the course equivalency task. To monitor the quality of the learned embeddings during this process, the `BinaryClassificationEvaluator` from the sentence-transformers library was employed. At the end of each training epoch, the evaluator was run on binary-labeled course pairs generated from the training set. The primary metric monitored was Average Precision based on cosine similarity, and the model checkpoint that achieved the highest score on these validation pairs was saved as the best model for that fine-tuning run.

Stage 4: Final Downstream Classifier Evaluation

The final and definitive evaluation was conducted on the held-out test portion of the PPM dataset. This stage used the feature vectors generated from the best-performing embedding models (both the original off-the-shelf versions and the newly fine-tuned versions). These feature sets were then used to train and evaluate the final selection of high-performing classifiers: KNN, Random Forest, SVM, and XGBoost. This ensures an unbiased assessment of the complete pipeline's ability to generalize to new, unseen data. To provide a comprehensive view of model performance, the standard suite of classification metrics was calculated from the confusion matrix: accuracy, precision, recall, and F_1-Score.

Beyond classification accuracy, the computational efficiency of each model pipeline was assessed by measuring both training and inference times from the hyperparameter cross-validation grid search. While training time provides insight into the resources required to develop a model, inference time is the more critical metric for this research's use case. Inference time directly affects the system's responsiveness and scalability in a production environment, making it a key factor in determining the practical viability of a given solution.

The performance of each of the final classifiers on each of the final feature sets was measured using these metrics. The results of this final evaluation, which form the core findings of this thesis, will be presented and analyzed in the subsequent chapter.

Chapter 4

Experimental Setup and Results

This chapter presents the empirical results that validate the decoupled, deep metric learning framework proposed in Chapter 3. Having detailed the systematic methodology, we now turn to the execution and outcomes of the multi-stage evaluation plan. The central objective of this chapter is to systematically assess the performance of each component of the pipeline—from the choice of embedding model and the impact of fine-tuning to the effects of dimensionality reduction and the selection of a downstream classifier—to identify the optimal configuration for both classification accuracy and computational efficiency.

The analysis is structured to first establish the validity of the core feature engineering approach before proceeding through the comparative stages of the evaluation. The chapter begins by detailing the computational environment, datasets, and formal evaluation metrics used throughout the experiments. It then immediately presents a critical ablation study to validate the efficacy of the novel composite distance vector (Δ_c), the cornerstone of the

feature representation. With the feature vector's design validated, the chapter then follows the primary evaluation sequence: first, establishing a baseline by assessing the performance of various off-the-shelf embedding models and classifiers; second, quantifying the performance gains achieved through fine-tuning; and third, analyzing the trade-offs between accuracy and efficiency introduced by dimensionality reduction. The chapter culminates in a definitive comparative analysis on the held-out test data, synthesizing all prior findings to identify the single best-performing pipeline configuration. We begin by outlining the experimental setup that forms the foundation for all subsequent results.

4.1 Experimental Environment & Datasets

Computational Environment

The research presented in this thesis was conducted using a hybrid computational environment, leveraging both cloud-based services for initial language model evaluations and a powerful on-premise high-performance computing (HPC) cluster for the primary, computationally intensive experiments. The initial direct classification tasks described in Section 3.1 were performed using Google's Gemini v1.0, a proprietary cloud-based Large Language Model. All subsequent stages of the research, including model fine-tuning, embedding generation, dimensionality reduction, and the comprehensive downstream classifier evaluations, were executed on San Francisco State University's "POLARIS" High Performance Compute

Cluster.

The POLARIS cluster provided a robust and flexible environment, running on Rocky Linux 8.9 with the Slurm Workload Manager for job scheduling. The GPU-intensive deep learning tasks, particularly the fine-tuning of the embedding models, were performed on the `gpucluster node`. This node is equipped with two AMD EPYC 9334 CPUs (providing 64 cores and 128 threads), 1 TB of RAM, and is accelerated by four NVIDIA A100 GPUs, each with 80 GB of VRAM and supported by CUDA v12.4. The extensive hyperparameter grid searches for the traditional machine learning classifiers were primarily run on the `cpucluster`, which consists of three nodes, each powered by two AMD EPYC 9534 CPUs (128 cores, 256 threads) and containing 576 GB of RAM. The cluster nodes are interconnected with a high-speed InfiniBand 200 Gb/s network, ensuring efficient data transfer during distributed tasks.

The entire experimental pipeline was implemented in Python, with Jupyter Notebooks serving as the primary environment for rapid prototyping, initial data exploration, and results analysis. The core deep learning components were built using PyTorch, with extensive use of the Hugging Face ecosystem, including the `Transformers`, `Sentence-Transformers`, and `Hub` libraries for model access and training. The classical machine learning experiments were conducted using `scikit-learn` and `XGBoost`. Data manipulation and analysis were handled with `numpy` and `pandas`, while `plotly` was used for visualization and `dill` for object serialization.

Table 4.1: Summary of Datasets Used in Evaluation

Characteristic	Initial Dataset	PPM Corpus
Source	Manually curated via ASSIST	Program Pathways Mapper (PPM)
Purpose	Preliminary screening, prototyping, and initial classifier evaluation	Definitive fine-tuning and final pipeline evaluation
Ground Truth	Established articulation agreements	Course Identification Number (C-ID)
Final Size	400 course pairs (for evaluation set)	2,157 courses (across 157 classes)
Partitioning	Stratified random sample	Stratified 50/50 train/test split

Datasets

The evolutionary methodology described in Chapter 3 was developed and validated using two distinct datasets, each serving a critical purpose at different stages of the research. The use of a smaller, initial dataset for prototyping followed by a larger, more comprehensive corpus for final evaluation is a core component of the experimental design. Table 4.1 provides a comparative summary of the key characteristics of both datasets used in this research.

Initial Dataset

The first dataset, hereafter referred to as the Initial Dataset, was a manually curated corpus used for the preliminary investigations in Stages 1 and 2 of the evaluation framework. This dataset was constructed from five lower-division courses at San Francisco State University and their established articulation agreements from 63 other California public colleges, sourced via the ASSIST repository. It was used to establish the initial performance baseline with the direct LLM classification approach and to prototype the decoupled pipeline. The final evaluation set for these preliminary stages consisted of a balanced, stratified random

sample of 400 course pairs drawn from a larger set of over 11,000 generated pairs.

PPM Corpus

The primary and definitive experiments for this thesis were conducted on the PPM Corpus, a larger and more robust dataset provided by the Program Pathways Mapper (PPM). This corpus served as the foundation for the full-scale implementation, fine-tuning, and final validation of the decoupled pipeline (Stages 3 and 4). After a filtering process detailed in Section 3.2, the final corpus consists of 2,157 courses, each labeled with a Course Identification Number (C-ID) that serves as the ground truth for equivalency. This corpus was partitioned via a stratified 50/50 split into a training set of 1,078 courses and a held-out test set of 1,079 courses. This held-out test set is crucial for methodological rigor and was used only once for the final, conclusive evaluation of the optimized pipelines to provide an honest and unbiased estimate of their generalization performance.

Evaluation Metrics

To facilitate a comprehensive and multi-faceted analysis, a suite of standard evaluation metrics was used to assess the various pipeline configurations. These metrics were chosen to measure performance across two critical dimensions: classification efficacy and computational efficiency, ensuring that the final recommended pipeline is not only accurate but also practical for real-world deployment.

Classification Efficacy

The core assessment of classification performance is based on a standard suite of metrics derived from the confusion matrix, which tabulates the counts of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). While Accuracy, defined as $\frac{TP+TN}{TP+TN+FP+FN}$, provides a general overview of correctness, it can be insufficient for capturing the nuances of a classifier's behavior. Therefore, this research also evaluates:

- **Precision**: Calculated as $\frac{TP}{TP+FP}$, this metric measures a model's exactness. High precision indicates that when the model predicts an equivalency, it is likely to be correct.
- **Recall**: Calculated as $\frac{TP}{TP+FN}$, this metric measures a model's completeness. High recall indicates that the model is effective at identifying the full set of all true equivalencies.

The F_1-Score, the harmonic mean of Precision and Recall ($2 \cdot \frac{Precision \cdot Recall}{Precision+Recall}$), is used as the primary metric for reporting and comparing the final classification performance. The F_1-Score provides a single, robust value that balances the trade-off between precision and recall, making it an ideal metric for this task where both avoiding false equivalencies and identifying all true ones are important.

Efficiency Metric

Beyond classification efficacy, the practical viability of each pipeline was assessed by measuring its computational efficiency. Both Training Time and Inference Time were systematically captured during the hyperparameter tuning process using the detailed statistics provided by `scikit-learn`'s `GridSearchCV` utility. Training time reflects the resources required to fit a model configuration, offering insight into the cost of experimentation. Inference time, reported by `GridSearchCV` as `score_time`, measures the time required for a trained model to make predictions on new data. For the use case of this research, inference time is considered the more critical efficiency metric, as it directly impacts the system's real-world responsiveness and scalability in a production environment. Finally, for the specific task of monitoring model improvement during the fine-tuning process (Stage 3), a specialized metric, Average Precision based on cosine similarity, was employed by the `BinaryClassificationEvaluator` to select the best-performing model checkpoint.

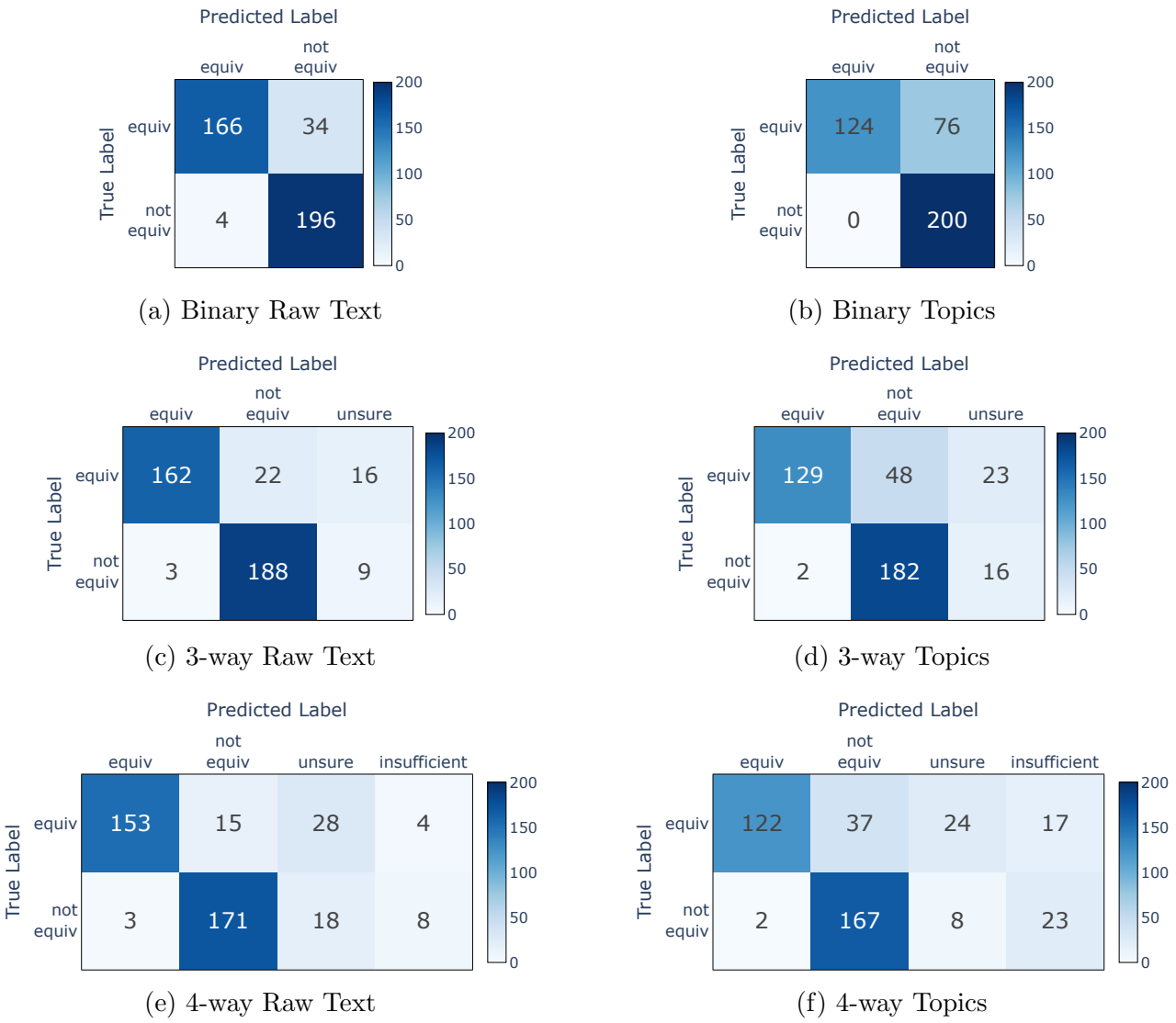
4.2 Direct LLM Classification

The initial phase of this research sought to establish a performance baseline by evaluating the feasibility of using a Large Language Model for end-to-end course equivalency classification. The experiments were conducted using the Initial Dataset, a balanced 400-pair sample, and leveraged Google's Gemini Pro v1.0 as the reasoning engine. The evaluation was designed to test two key conditions: the impact of input data type (full raw text vs. extracted

topics) and the effect of different classification schemes (binary, 3-way, and 4-way) on the model's performance and behavior.

The classification results were encouraging, with the model achieving a peak accuracy of 90.5% in the binary task using raw text. A primary finding from this phase was that using the full, unprocessed raw text consistently yielded superior results compared to using only the extracted structured topics. This performance gap is likely attributable to the challenges inherent in the data extraction step itself. The process of deducing structured topics from unstructured text proved to be the most time-consuming and difficult aspect of this initial approach, requiring two to four times more time than the classification task and exhibiting a high sensitivity to prompt wording—a finding that aligns with previous studies on prompt engineering by Sclar et al., Errica et al., Chen et al., Cheung, and Gerych et al. [35, 15, 8, 9, 17]. Although Gemini demonstrated a general capability to understand the context of the course descriptions, its limitations in flawlessly deducing course topics appear to have introduced enough noise to degrade the performance of the topics-based classification.

An analysis of the confusion matrices, presented in Figure 4.1 reveals a consistent and telling behavioral pattern: the model exhibits a strong conservative bias. Across all scenarios, the "not equivalent" class showed higher recall than precision, while the "equivalent" class showed higher precision than recall. For example, in the binary raw text evaluation, the model produced 34 false negatives but only 4 false positives. This indicates that the model is far more likely to misclassify a truly equivalent course as "not equivalent" than it is to incorrectly approve a non-equivalent pair. This conservative tendency suggests the LLM



(a) Binary Raw Text

(b) Binary Topics

(c) 3-way Raw Text

(d) 3-way Topics

(e) 4-way Raw Text

(f) 4-way Topics

Figure 4.1: LLM Classification Confusion Matrices

operates with a high internal threshold for declaring equivalency, preferring to err on the side of caution.

The introduction of the "unsure" and "insufficient data" categories was designed to simulate how a human advisor might handle ambiguity. While there is no ground truth for these labels, their value lies in the model's ability to isolate cases that require further manual

Table 4.2: Performance Summary of Direct LLM Classification

Input Data	Classification Task	Accuracy	F1-Score*	Precision*	Recall*
Raw Text	Binary	0.9050	0.8973	0.9765	0.8300
	3-way	0.8750	0.8877	0.9818	0.8100
	4-way	0.8100	0.8596	0.9808	0.7650
Topics	Binary	0.8100	0.7654	1.0000	0.6200
	3-way	0.7775	0.7795	0.9847	0.6450
	4-way	0.7225	0.7531	0.9839	0.6100

* F1-Score, Precision, and Recall are reported for the positive class ("Equivalent").

review. This is particularly relevant for courses that may have significant topical overlap but different learning outcomes, or for descriptions that are too sparse to allow for a definitive decision. For instance, in the 4-way raw text evaluation, the model flagged 46 pairs as "unsure" and 12 as having "insufficient" information, effectively separating these ambiguous cases from the main classification flow. Table 4.2 provides a comprehensive summary of the performance metrics for all six experimental conditions.

To contextualize the performance of Gemini Pro, a comparative analysis was conducted against a suite of other prominent open-source LLMs (see Table 4.3 for a summary of their specifications and performance). The results of this benchmark revealed that Gemini Pro v1.0 outperformed all other models tested. While most models were able to complete the task, their performance varied significantly. The strongest competitor was Anthracite Magnum v1 72B, which achieved a respectable accuracy of 85.25%, but still falling short of Gemini's 90.5%.

This benchmark also highlighted significant operational challenges. Several models strug-

Table 4.3: Model Specifications and Performance

Model Name	Parameters*	Context Length	Support	Accuracy	Precision	Recall	F_1-score
Google Gemini Pro 1.0	Unknown	32,768	400	**0.9050**	0.9765	**0.8300**	**0.8973**
Meta Llama 3.1 8B Instruct	8	128,000	208 (52%)	0.6250	1.0000	0.3500	0.5185
Microsoft Phi 3 Medium Instruct	14	128,000	400	0.7100	1.0000	0.4200	0.5915
Google Gemma 2 27b	27.2	8,000	N/A	N/A	N/A	N/A	N/A
Meta Llama 3.1 70B Instruct	70	128,000	400	0.7350	1.0000	0.4700	0.6395
Qwen 2 72B Instruct	72.7	131,072	400	0.7650	1.0000	0.5300	0.6928
Anthracite Magnum v1 72B	72.7	32,768	400	0.8525	1.0000	0.7050	0.8270
CalmeRys 78B Orpo v0.1	78	32,768	400	0.7175	1.0000	0.4350	0.6063
Mixtral 8x22B Instruct v0.1	141	65,536	400	0.6450	1.0000	0.2900	0.4496
Meta Llama 3.1 405B Instruct**	405	128,000	400	0.7775	1.0000	0.5550	0.7138

*in Billions

**INT4 Quantized Model

gled to reliably produce valid output. For example, Google's Gemma 2 27B consistently generated unintelligible or irrelevant responses, preventing the collection of any meaningful performance data. Similarly, Meta's Llama 3.1 8B Instruct model was only able to respond correctly to approximately 52% of the prompts, making it too unreliable for practical use. A surprising insight from this analysis was the lack of a strong correlation between model size and performance on this task. It is important to note, however, that the prompts used in this evaluation were originally designed and optimized for Gemini. This lack of prompt specificity for the other models may have contributed to their lower accuracy and, in some cases, their inability to respond appropriately.

This initial study confirmed that direct LLM classification can achieve high accuracy but is sensitive to input quality and exhibits a conservative bias. More importantly, it validated the fundamental limitations of the approach, its computational cost, lack of a quantifiable similarity score, and "black box" nature, that motivated the development of the decoupled pipeline. The following sections will now present the results of this more advanced

framework, beginning with a critical validation of its most novel component: the composite distance vector.

4.3 Composite Distance Vector Validation

A foundational contribution of this research is the design of the composite distance vector, Δ_c, which serves as the input for all downstream classifiers. As detailed in the methodology, this vector was designed to provide a richer feature set than a single similarity score alone by combining granular, dimension-specific (local) information with a holistic, global similarity metric . This section presents the results of a critical ablation study designed to validate this hypothesis and quantify the impact of the global cosine similarity term on classifier performance.

The experimental setup for this validation involved training the full suite of classifiers on two different versions of the feature set. The first used the complete composite distance vector (Δ_c), which includes both the element-wise differences and the cosine similarity score. The second used an ablated vector (Δ_l) containing only the element-wise differences. By comparing the performance across these two conditions, it is possible to isolate and measure the contribution of the global similarity feature.

The results of this ablation study revealed a stark and significant dichotomy between the behavior of linear and non-linear models. For all linear models evaluated—Logistic Regression, Ridge Regression, Lasso, and Linear Discriminant Analysis (LDA), the inclusion

of the cosine similarity term resulted in a dramatic and statistically significant improvement in classification performance. For example, using the off-the-shelf BGE embeddings with no dimensionality reduction, the F1-score for the Logistic Regression classifier surged from 0.686 to 0.901 with the addition of the single cosine similarity feature. This massive performance lift demonstrates that for these simpler models, which are constrained to learning linear decision boundaries, the explicit global similarity feature provides critical information that they are unable to derive from the local element-wise differences alone.

In stark contrast, the more complex non-linear models, k-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Random Forest (RF), were not significantly influenced by the inclusion or omission of the cosine similarity term. Their performance remained exceptionally high in both conditions. For instance, the F_1-score for the SVM classifier was virtually unchanged, registering 0.9988 without the term and 0.9983 with it. This suggests that these more powerful models, which can learn complex, non-linear interactions between features, are capable of effectively reconstructing the global similarity information directly from the high-dimensional local difference vector.

The conclusion from this ablation study is clear and decisive. The composite distance vector, Δ_c, is demonstrably the superior feature representation. It provides a critical performance boost to simpler, baseline models while causing no harm to the performance of more sophisticated, non-linear models. Therefore, based on this empirical validation, the full composite distance vector was adopted as the standard feature set for all subsequent experiments presented in this thesis.

4.4 Fine-Tuning a Model

This section details the empirical investigation into enhancing the performance of pre-trained embedding models by fine-tuning them on the domain-specific PPM Corpus. The central hypothesis, as stated in Section 3.3, is that adapting a general-purpose model to the specific linguistic nuances of course catalog text will yield a more discriminative feature representation for the downstream course equivalency task. This process, a form of deep metric learning, aims to restructure the embedding space such that the geometric distance between vectors directly corresponds to semantic similarity. Such an approach is particularly relevant for scenarios with high intra-class variance and low inter-class variance, a common characteristic of specialized domains where fine distinctions are paramount. This section outlines the rigorous experimental design, presents the empirical results, and provides a detailed analysis that culminates in the selection of a single, optimized model for subsequent evaluation.

Evaluation Framework

To identify the optimal fine-tuning configuration, a systematic evaluation was designed and executed, exploring a matrix of key hyperparameters. The experimental runs were orchestrated via a series of shell scripts that automated the submission of training jobs with different configurations. This framework was designed to methodically test the impact of different loss functions and learning rate schedules on model performance.

- **Base Models**: The experiments focused on fine-tuning the most promising small model identified in the initial screening (Section 3.3): `BAAI/bge-small-en-v1.5` (BGE). This model was selected to represent high-performing small architecture.

- **Loss Functions**: A critical component of metric learning is the choice of the loss function, which defines the optimization objective. This study evaluated four distinct online triplet mining strategies, each implemented as a loss function within the sentence-transformers library. The selection of a loss function directly influences which training examples are considered most informative, impacting both convergence speed and final model performance. The evaluated functions were:

  - **BatchAllTripletLoss (atl)**: Computes the loss over all valid triplets within a mini-batch, offering a comprehensive but potentially diluted learning signal.

  - **BatchSemiHardTripletLoss (shtl)**: Considers only semi-hard triplets, where the negative sample is more distant than the positive but still violates the margin. This is a common strategy that balances stability and learning effectiveness.

  - **BatchHardTripletLoss (htl)**: Focuses on the hardest triplets in each batch, providing a strong but potentially unstable learning signal by using the most challenging examples.

  - **BatchHardSoftMarginTripletLoss (hsmtl)**: A variation of `BatchHardTripletLoss` that does not require a fixed margin parameter, simplifying hyperparameter tuning.

- **Learning Rate Schedulers**: The learning rate schedule governs how the learning rate is adjusted during training, a critical factor for navigating complex loss landscapes and avoiding premature convergence. Three distinct schedules, with parameters defined in the training script, were evaluated.
  - **linear (lr_linear)**: A standard linear decay of the learning rate from its initial value to zero over the course of training.
  - **cosine_with_restarts (cr_v1)**: A cosine annealing schedule with 6 restart cycles and an early stopping patience of 21 epochs.
  - **cosine_with_restarts (cr_v2)**: A more aggressive cosine annealing schedule with 10 restart cycles and a reduced early stopping patience of 15 epochs.

The dynamic behavior of these schedulers, particularly the periodic resets of the cosine annealing schedules, is visualized in the training logs, confirming their implementation as designed (Image 1).

Implementation Details

The fine-tuning experiments were conducted with a high degree of methodological rigor, leveraging a sophisticated distributed training environment and custom tools to ensure robustness and reproducibility. The computationally intensive nature of fine-tuning transformer models necessitated the use of a high-performance computing (HPC) environment. All experiments were executed on the POLARIS HPC cluster at San Francisco State Uni-

versity. The SLURM submission script specifies a distributed data-parallel (DDP) strategy using `torchrun` with a configuration of `nproc_per_node=4`, distributing the workload across four NVIDIA A100 GPUs on a single node. This setup significantly accelerated the training process, making the comprehensive grid search feasible.

The choice of a data batching strategy is intrinsically linked to the mechanics of the loss function. The batch-based triplet loss functions perform "online" mining, meaning they construct informative (anchor, positive, negative) triplets on-the-fly from the samples present within each training mini-batch. For a valid triplet to be formed, the batch must contain at least two examples from the same class, enabling the selection of a distinct anchor and positive sample. Standard random sampling does not guarantee this condition, which can lead to inefficient training where many anchors in a batch cannot form valid triplets. To address this, the `GroupByLabelBatchSampler` class was employed, as specified in the training script. This sampler, a feature of the `sentence-transformers` library, ensures that each mini-batch is constructed by grouping multiple samples with the same label. This guarantees that every batch contains the necessary class diversity to form valid and informative triplets for all anchors, maximizing the utility of each training step.

To manage the long-running, distributed training jobs, a set of custom `TrainerCallback` classes were implemented. The `DistributedStoppingCallback` is particularly significant as it demonstrates a sophisticated approach to process control in a distributed setting. This callback integrates two crucial features: early stopping based on a patience parameter (i.e., halting training if the validation metric does not improve for a set number of evaluations)

and a hard time limit (`max_duration`) to ensure jobs do not exceed their allocated time on the HPC cluster. The callback is designed to perform these checks on the main process (rank 0) during the evaluation phase and then broadcast a stop signal to all other worker processes. This architecture prevents the common issue of hanging processes in distributed training and ensures a graceful, synchronized termination of the job, underscoring the methodological robustness of the experimental setup.

The performance of the model during the fine-tuning process was monitored at the end of each epoch using the `BinaryClassificationEvaluator` from the `sentence-transformers` library. The metric used to identify and save the best-performing model checkpoint was the F_1-score on a binary classification task, specifically `eval_binary classification evaluator_cosine_f1`. This represents a deliberate and important design choice. By training the model to optimize a triplet loss objective while evaluating its performance on a binary pair classification task, the framework directly measures the model's aptitude for the ultimate downstream task. This prevents overfitting to the specific mechanics of the triplet loss and provides a more honest assessment of the model's ability to generalize.

Empirical Results of Fine-Tuning Configurations

The systematic evaluation of the twelve fine-tuning configurations (four loss functions crossed with three learning rate schedules) on the BGE base model produced a rich set

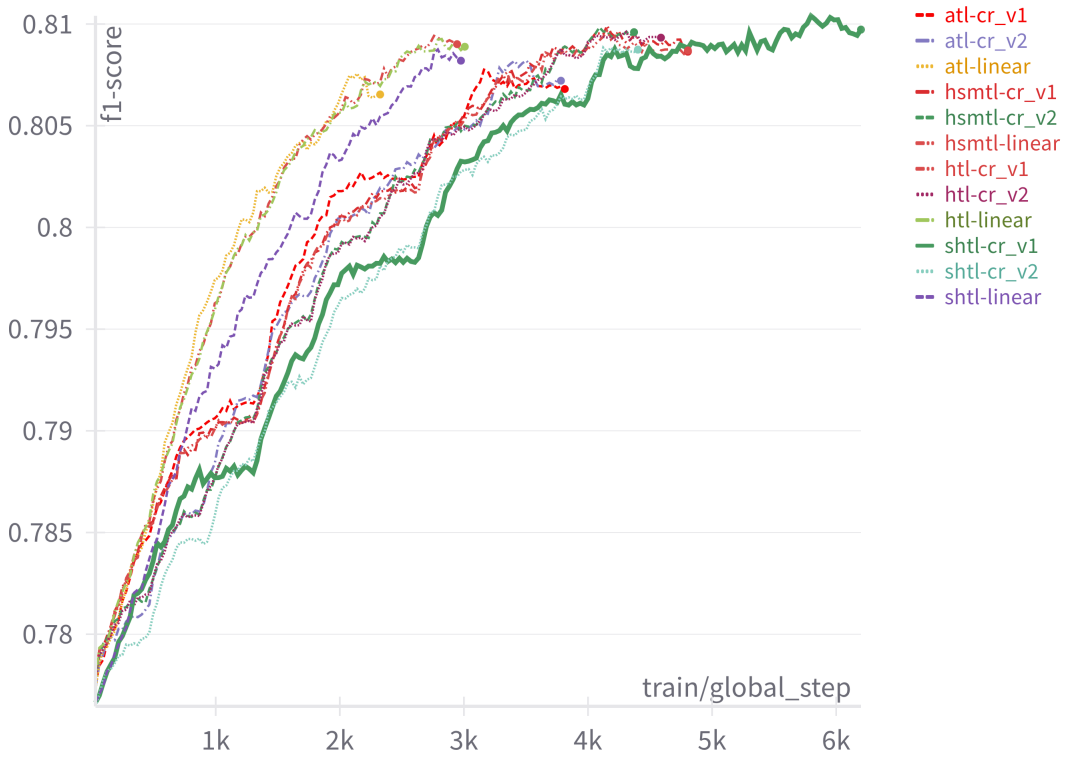


Figure 4.2: Validation F_1-Score

of performance data. The training progress was logged and visualized, allowing for both qualitative and quantitative analysis of the different strategies.

Visualization of Training Dynamics

Figure 4.2 presents the validation F_1-score, calculated on the binary classification evaluation task, as a function of the global training step for each of the twelve experimental runs. This visualization provides a clear, qualitative comparison of the learning dynamics associated with each configuration.

A visual inspection of the learning curves reveals several distinct patterns. The con-

figurations utilizing `BatchHardSoftMarginTripletLoss` seem to consistently demonstrate superior performance. They exhibit some of the steepest learning curves, indicating faster convergence, and reach some of the highest peak F_1-scores, surpassing all loss function types. The `BatchHardTripletLoss` family of functions seem to perform similarly, albeit slightly worse. `AllTripletLoss` configurations display the steepest learning curves, but show a performance collapse much earlier and at much lower F_1-score peaks than any of the other functions. Finally, the `BatchSemiHardTripletLoss` configurations exhibit the slowest learning curves, and do not seem to seem to reach the highest peak scores on average. Nonetheless, it appears that its more erratic behavior near its plateau can produce results that surpass all other functions. These visual trends strongly suggest that the choice of triplet mining strategy is a major factor in determining the training speed and final performance of the fine-tuned model for this task.

Quantitative Performance Summary

To provide a more precise and formal comparison, the exact peak validation F_1-scores achieved by each configuration are summarized in Table 4.4. This table distills the dynamic information from the learning curves into a single, quantitative metric for each experimental condition, enabling a direct and data-driven selection of the optimal configuration. The quantitative data presented in Table 4.4 reveals a nuanced outcome that adds precision to the qualitative analysis of the learning curves. While the `BatchHardTripletLoss` and `BatchHardSoftMarginTripletLoss` configurations consistently achieved high scores

Table 4.4: Peak Validation F1-Score of Fine-Tuning Configurations on BGE Model

Loss Function	**Linear Scheduler**	**Cosine Restarts v1**	**Cosine Restarts v2**
`BatchAllTripletLoss` (atl)	0.8076	0.8078	0.8082
`BatchHardSoftMarginTripletLoss` (hsmtl)	0.8094	0.8099	0.8098
`BatchHardTripletLoss` (htl)	0.8093	0.8097	0.8096
`BatchSemiHardTripletLoss` (shtl)	0.8088	**0.8104**	0.8088

across all schedulers, the single best-performing configuration was the combination of `BatchSemiHardTripletLoss` and the `cr_v1` learning rate schedule, which achieved the highest peak validation F_1-score of 0.8104. This result suggests that while aggressive hard-negative mining provides a strong and reliable performance floor, a more moderate semi-hard mining strategy, when paired with an appropriate cosine annealing schedule, can find a slightly better optimum for this specific task.

4.5 Classifier Evaluation

4.6 Dimensionality Reduction

4.7 Comparative Analysis

4.8 Summary

Bibliography

[1] Public Agenda. *Beyond Transfer: Insights from a Survey of American Adults*. https://publicagenda.org/resource/beyond-transfer/ (visited on 06/30/2025).

[2] Akiko Aizawa. "An information-theoretic perspective of tf-idf measures". In: *Information Processing & Management* 39.1 (2003), pp. 45–65. ISSN: 0306-4573. DOI: https://doi.org/10.1016/S0306-4573(02)00021-3. https://www.sciencedirect.com/science/article/pii/S0306457302000213.

[3] ASSIST. *Frequently Asked Questions*. https://resource.assist.org/FAQ (visited on 06/30/2025).

[4] ASSIST. *General Information*. https://resource.assist.org/About/General-Information (visited on 06/30/2025).

[5] Leticia Tomas Bustillos et al. *The Transfer Maze: The High Cost to Students and the State of California*. The Campaign for College Opportunity, Sept. 17, 2017.

[6] California Community Colleges Chancellor's Office. *Management Information Systems Data Mart*. 2024. https://datamart.cccco.edu/Students/Student%5C_Headcount%5C_Term%5C_Annual.aspx (visited on 09/13/2024).

[7] California State University Office of the Chancellor. *Enrollment*. 2024. https://www.calstate.edu/csu-system/about-the-csu/facts-about-the-csu/enrollment (visited on 09/13/2024).

[8] Yi-Pei Chen, Kuanchao Chu, and Hideki Nakayama. "LLM as a Scorer: The Impact of Output Order on Dialogue Evaluation". In: *ArXiv* abs/2406.02863 (2024). https://api.semanticscholar.org/CorpusID:270258565.

[9] Ming Cheung. "A Reality check of the benefits of LLM in business". In: *ArXiv* abs/2406.10249 (2024). https://api.semanticscholar.org/CorpusID:270560730.

[10] National Student Clearinghouse. *College Transfer Enrollment Grew by 5.3% in the Fall of 2023*. https://www.studentclearinghouse.org/news/college-transfer-enrollment-grew-by-5-3-in-the-fall-of-2023/ (visited on 06/30/2025).

[11] National Student Clearinghouse. *College Transfer Enrollment Grew for Third Straight Year*. https://www.studentclearinghouse.org/news/college-transfer-enrollment-grew-for-third-straight-year/ (visited on 06/30/2025).

[12] National Student Clearinghouse. *DATA DIVE: Returning Learners Lead Transfer Population*. https://www.studentclearinghouse.org/nscblog/data-dive-returning-learners-lead-transfer-pop/ (visited on 06/30/2025).

[13] Kevin Cook. *California's Higher Education System*. 2024. https://www.ppic.org/publication/californias-higher-education-system/ (visited on 09/06/2024).

[14] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL]. https://arxiv.org/abs/1810.04805.

[15] Federico Errica et al. "What Did I Do Wrong? Quantifying LLMs' Sensitivity and Consistency to Prompt Engineering". In: *ArXiv* abs/2406.12334 (2024). https://api.semanticscholar.org/CorpusID:270562829.

[16] The Pytorch Foundation. *CosineAnnealingLR*. https://docs.pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.CosineAnnealingLR.html (visited on 06/30/2025).

[17] Walter Gerych et al. "Who Knows the Answer? Finding the Best Model and Prompt for Each Query Using Confidence-Based Search". In: *AAAI Conference on Artificial Intelligence*. 2024. https://api.semanticscholar.org/CorpusID:268717587.

[18] Inc. Hugging Face. *Loss Overview*. https://sbert.net/docs/sentence_transformer/loss_overview.html (visited on 06/30/2025).

[19] Inc. Hugging Face. *Losses*. https://sbert.net/docs/package_reference/sentence_transformer/losses.html (visited on 06/30/2025).

[20] Inc. Hugging Face. *Samplers*. https://sbert.net/docs/package_reference/sentence_transformer/sampler.html (visited on 06/30/2025).

[21] Weijie Jiang and Zachary A Pardos. "Evaluating Sources of Course Information and Models of Representation on a Variety of Institutional Prediction Tasks." In: *International Educational Data Mining Society* (2020).

[22] Pengfei Liu et al. "Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing". In: *ACM Comput. Surv.* 55.9 (Jan. 2023). ISSN: 0360-0300. DOI: 10.1145/3560815. https://doi.org/10.1145/3560815.

[23] Shamrock Solutions LLC. *Transfer Credit Automation: How Universities Are Simplifying Course Equivalency*. Overland Park, KS, USA. https://www.shamrocksolutionsllc.com/post/transfer-credit-automation-universities (visited on 06/30/2025).

[24] Ilya Loshchilov and Frank Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. https://arxiv.org/abs/1711.05101.

[25] H Ma et al. "Course recommendation based on semantic similarity analysis". In: *2017 3rd IEEE International Conference on Control Science and Systems Engineering*. 2017, pp. 638–641.

[26] Z. A Pardos, H Chau, and H Zhao. "Data-assistive course-to-course articulation using machine translation". In: *Proceedings of the Sixth Conference on Learning@ Scale*. 2019, pp. 1–10.

[27] Zachary Pardos, Hung Chau, and Haocheng Zhao. "Data-Assistive Course-to-Course Articulation Using Machine Translation". In: (July 2019). DOI: 10.1145/3330430.3333622.

[28] Zachary A. Pardos, Hung Chau, and Haocheng Zhao. "Data-Assistive Course-to-Course Articulation Using Machine Translation". In: *Proceedings of the Sixth (2019) ACM Conference on Learning @ Scale*. L@S '19. Chicago, IL, USA: Association for Computing Machinery, 2019. ISBN: 9781450368049. DOI: 10.1145/3330430.3333622. https://doi.org/10.1145/3330430.3333622.

[29] Zachary A. Pardos, Zihao Fan, and Weijie Jiang. *Connectionist Recommendation in the Wild: On the utility and scrutability of neural networks for personalized course guidance*. 2018. arXiv: 1803.09535 [cs.AI]. https://arxiv.org/abs/1803.09535.

[30] Zachary A. Pardos, Zihao Fan, and Weijie Jiang. "Connectionist recommendation in the wild: on the utility and scrutability of neural networks for personalized course guidance". In: *User Modeling and User-Adapted Interaction* 29.2 (Apr. 2019), pp. 487–525. ISSN: 0924-1868. DOI: 10.1007/s11257-019-09218-7. https://doi.org/10.1007/s11257-019-09218-7.

[31] Stephen Porter. "Assessing Transfer and Native Student Performance at Four-Year Institutions". In: *39th Annual Forum of the Association for Institutional Research*. June 1999.

[32] Regents of the University of California, The. *Fall enrollment at a glance*. 2024. `https://www.universityofcalifornia.edu/about-us/information-center/fall-enrollment-glance` (visited on 09/13/2024).

[33] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Nov. 2019. `https://arxiv.org/abs/1908.10084`.

[34] Elvis Saravia. "Prompt Engineering Guide". In: *https://github.com/dair-ai/Prompt-Engineering-Guide* (Dec. 2022). `https://www.promptingguide.ai`.

[35] Melanie Sclar et al. "Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting". In: *ArXiv* abs/2310.11324 (2023). `https://api.semanticscholar.org/CorpusID:264172710`.

[36] Natenaile Asmamaw Shiferaw et al. *BERT-Based Approach for Automating Course Articulation Matrix Construction with Explainable AI*. 2024. arXiv: `2411.14254 [cs.LG]`. `https://arxiv.org/abs/2411.14254`.

[37] Sharon Slade and Paul Prinsloo. "Learning Analytics: Ethical Issues and Dilemmas". In: *American Behavioral Scientist* 57.10 (2013), pp. 1510–1529. DOI: 10.1177/0002764213479366. eprint: https://doi.org/10.1177/0002764213479366. https://doi.org/10.1177/0002764213479366.

[38] The National Task Force on the Transfer and Award of Credit. *Reimagining Transfer for Student Success*. Report to Congressional Requesters. American Council on Education, Mar. 2020. https://www.gao.gov/products/gao-17-574.

[39] United States Government Accountability Office. *Higher Education: Students Need More Information to Help Reduce Challenges in Transferring College Credits*. Report to Congressional Requesters GAO-17-574. United States Government Accountability Office, Aug. 14, 2017. https://www.gao.gov/products/gao-17-574.

[40] Yinuo Xu and Zach A. Pardos. "Extracting Course Similarity Signal using Subword Embeddings". In: *Proceedings of the 14th Learning Analytics and Knowledge Conference*. LAK '24. Kyoto, Japan: Association for Computing Machinery, 2024, pp. 857–863. ISBN: 9798400716188. DOI: 10.1145/3636555.3636903. https://doi.org/10.1145/3636555.3636903.

[41] Qinyuan Ye et al. *Prompt Engineering a Prompt Engineer*. 2024. arXiv: 2311.05661 [cs.CL]. https://arxiv.org/abs/2311.05661.

[42] Jian Yu et al. "Deep metric learning with dynamic margin hard sampling loss for face verification". In: *Signal, Image and Video Processing* 14.4 (June 2020), pp. 791–798.

ISSN: 1863-1711. DOI: 10.1007/s11760-019-01612-3. https://doi.org/10.1007/s11760-019-01612-3.