BUZZ-AI 🐝
Enhancing Georgia Tech Course Discovery Using Transformer-Based Semantic Search

Akshat Karwa
akarwa7@gatech.edu
Georgia Institute of Technology

Mehul Rastogi
mehulrastogi@gatech.edu
Georgia Institute of Technology

Pranay Begwani
pbegwani3@gatech.edu
Georgia Institute of Technology

Vidushi Maheshwari
vmaheshwari32@gatech.edu
Georgia Institute of Technology

Abstract

Course selection is a critical yet challenging process for students, often hindered by the complexities of navigating through extensive course catalogs and identifying relevant educational opportunities. This project introduces BUZZ-AI, an innovative course recommendation system, that leverages advanced transformer-based semantic search techniques to revolutionize course discovery at Georgia Tech. By integrating state-of-the-art natural language processing methods, BUZZ-AI transforms traditional course recommendation approaches. The system employs sophisticated vectorization techniques to capture the semantic and contextual nuances of course details. By utilizing Facebook AI Similarity Search (FAISS), it provides precise and personalized course matching. BUZZ-AI efficiently identifies and ranks the most relevant courses based on user queries. The core innovation of BUZZ-AI is its ability to understand the deeper meaning behind course content and provide intelligent, context-aware recommendations. The system supports targeted filtering to generate tailored recommendations. Evaluations demonstrate BUZZ-AI's potential to significantly enhance the course selection experience and offer students a more intuitive and personalized pathway to discovering educational opportunities. This paper contributes to the emerging field of intelligent recommendation systems, showcasing the potential of state-of-the-art transformer models in generating context-aware recommendations.

1 Introduction

1.1 Problem Statement

With the increasing number of courses offered in academic institutions, choosing the most relevant courses has become increasingly challenging. The overwhelming abundance of options often leads to decision fatigue and leaves several valuable courses unexplored [Esteban et al. (2021)]. Factors such as limited visibility into course offerings, non-personalized discovery tools, and a lack of alignment with the student's goals leads to unnecessary wastage of time and efforts spent during the selection process. Additionally, the lack of awareness regarding the courses that align with a student's interests furthers magnifies the problem. For instance, a student interested in Game Theory may struggle to discover the essential mathematics courses to pursue, instead of received quick, tailored recommendations based on contextual and semantic insights during course registration. Several filtering-based approaches exist, but only a few recommendation systems have been developed, and these

primarily rely on historical enrollment trends and simplistic keyword matching. Thus, they fail to capture the semantic meaning and contextual relationships within course content and its parallels to real-world topics.

1.2 Motivation & Significance

With the rapid growth of interdisciplinary programs, online learning platforms, and micro-credentialing opportunities, the need for intelligent, personalized course recommendation systems has never been more critical. Such systems would alleviate students' cognitive overload, empower them to make informed academic choices, reduce dropout rates, and enhance overall educational outcomes. As students ourselves, we have experienced the challenges of navigating through extensive course catalogs, often struggling to find courses meaningful to us. This firsthand understanding motivates us to address these pain points by creating a smarter and more efficient system.

BUZZ-AI would encourage students to explore new domains aligned with their evolving interests, fostering curiosity and interdisciplinary learning. It will transform course discovery into a seamless and intuitive experience and bridge gaps between academic offerings and students' aspirations. Ultimately, it will streamline the educational journey while simultaneously providing students with a clear understanding of the full range of options available to them. By the time they graduate, students will not only fulfill their degree requirements but also enrich their overall knowledge and be better prepared for the future.

1.3 Overview of Approach

We use semantic embeddings of Course Title, Description, Professor Research Area, and Course Syllabus to store course content in a vector space and use semantic similarity on the user's query to find the best matches. Buzz-AI also allows user to filter based on major, degree and other attributes that a student might have constraints on when picking classes. Moreover, the recommendations are personalized to the user by maintaining user profiles that update over time.

To evaluate the effectiveness of Buzz-AI, we conducted a user study of 10 Georgia Tech students to measure the accuracy of the recommendations. Additionally, we compared different embedding models to identify the most effective approach for our use case, balancing accuracy and computational efficiency.

2 Background

2.1 Existing Research & Approaches

Harvard's classes.wtf aims to simplify course search, but the problem lies in the accuracy needed to search through their database. Their focus lies in improving speed of search, however, this paper addresses the accuracy of search and personalization of search.

Zhao et al. (2024) introduces the concept of using LLMs to make recommender systems. While this work plays pivotal role in introducing recommendation utilities of LLMs, it doesn't address their use case in the education industry, and it doesn't account for the differences that may arise as a result of embedding differences.

Sun et al. (2019) uses BERT to develop recommendation systems and is helpful in introducing the fact that embeddings can be utilized for recommendation improvements, thus providing motivation to test out the accuracy changes with different embedding models.

Rao & Lin (2024) introduced the concept of utilizing LLMs to improve recommendations for educational purposes. While this lays the foundation for such work, its scope is restricted to MOOCs (massive open online courses) and doesn't account for school specific curricula and course offerings.

While all of the above mentioned papers and resources are crucial in their own sense, because they introduce novel methods for improving recommendation systems and the lay the foundations of utilizing LLMs for the same, they don't specifically address the problem faced by college students in selecting the courses offered to them.

3 Data Description

3.1 Data Sources & Collection Methods

The paper focuses on the Georgia Tech course roster for *Spring 2025*. The latest json blob of courses was taken from GT-Scheduler-Contributors (2024) who maintain the crawler that powers GT Scheduler.

Professor Bios were extracted from their profile on their respective college's websites. Using BeautifulSoup, professor bios were scraped from the college's website. For example, for Computer Science, Computer Engineering, and Computational Media professors, data was scraped from this link - `https://www.cc.gatech.edu/people/{professor_name}`. In the case when multiple authors are present, Professor Research areas are added to to the string with the `Another Professor` separator.

Course Titles were extracted by scraping OSCAR. Again, BeautifulSoup was leveraged in-order to scrape the course titles. The CRN for the respective courses is taken in-order to extract the title from Oscar.

The entire text blob is also extracted from class syllabuses so that vector embeddings are much more context aware. There's a custom script that extracts data from pdfs that leverages `pdfplumber`. This way we are able to contextually use the entire syllabus text data in-order to make vectors.

```
https://oscar.gatech.edu/pls/bprod/bwckschd.p_disp_detail_sched?term_in=202502&crn_in={CRN}
```

There's also a way to input *Course Name, Description, & Syllabus Path* manually. There's a script that unions all the raw data from the crawler, scraped data, and manual data to create a robust and cohesive dataset.

3.2 Data Cleaning and Pre-processing

From the JSON blob, First the course Name and the Description are extracted. Then, the raw section information is extracted. The raw section information is then filtered and cleaned. The information gives us information about Professor Names, and Campus Index (Atlanta, OMSCS, Lorraine, etc.).

The course code has been taken the key in our data model. For example, `CS 2110` is a key in the data model. The main problem comes with Special Topics classes - both with data and indexing. To mitigate the data issue, we've written scripts that scrape for 1) Class Title from OSCAR, 2) Professor Research Area from Georgia Tech's College Websites, and 3) Manual Inputs for Name, Description, and Syllabus. These mechanisms are used for other classes as well, but are of particular importance for Special Topics classes. To solve for the indexing issue, special topics classes have keys with sections appended to them. An example key is: `CS 8803-BC`. So, all in all for each class it has the `Name, Description, Syllabus, and Section Information`. Section Information internally has information for each section the: `Professor(s), Campus Index, CRN, and Professor Research Area(s)`. Followed by this, manual data merging is done to ensure that a common source of truth Dataset is constructed. Irrelevant independent research based classes with 9000 codes have been removed from the data.

3.3 Data Model

3.3.1 Example Data Model for a Regular Class


```
    "CS 3510": {
        "Name": "Dsgn & Analysis-Algorithms",
        "Description": "Basic techniques of design...",
        "Section Information": {
            "A": {
                "Professors": [
                    "Gerandy Brito (P)"
                ],
                "Campus Index": 0,
                "CRN": "21289",
                "Research Area": "[Prof Research Area]"
            },
            "B": {
                "Professors": [
                    "Zongchen Chen (P)"
                ],
                "Campus Index": 0,
                "CRN": "28210",
                "Research Area": "Zongchen Chen is an assistant professor ...."
            }
        },
        "Syllabus": "https://path_to_pdf.com"
    }
```


3.3.2 Example Data Model for a Special Topics Class


```
"CS 8803-GEE": {
        "Name": "Global Entrepreneurship",
        "Description": "This course will provide you with real-world .... ",
        "Section Information": {
            "GEE": {
                "Professors": [
                    "Brian McGreggor (P)",
                    "Michael Best"
                ],
                "Campus Index": 0,
                "CRN": "29205",
                "Research Area": "Dr. Michael L. Best is Executive .... "
            }
        },
        "Syllabus": "../Syllabus/8803-GEE.pdf"
    }
```


4 Methodology

4.1 Tokenization & Embedding Methods

Most of these models are based on BERT, which uses a full-sized transformer, introduced by Vaswani et al. (2023).

- **Bert-base-uncased Devlin et al. (2019)** This is a bidirectional encoder representing transformer based model which uses the full transformer with 12-24 layers. The primary focus tends to be next sentence prediction focused on sequential tasks. This is trained on a rather small dataset and a pooling layer was added after the

transformer for dimensionality reduction, capturing limited, but focused information.

- **RoBERTa Liu et al. (2019)**: Uses a similar architecture to BERT. The key difference lies in how RoBERTa is trained. The dataset used to train it, is almost 10 times larger than that of BERT, making it more robust than BERT. Finally, another key point of difference between RoBERTa and BERT is the fact that, unlike BERT, RoBERTa doesn't do sentence prediction. Aroca-Ouellette & Rudzicz (2020) shows that next sentence prediction may in fact be detrimental to BERTś performance. Finally, RoBERTa uses dynamic masking, to generate new mask patterns during training, and uses larger batch sizes and more extensive hyperparameter tuning, making it more efficient and effective than BERT.
- **DistilRoBERTa**: DistilRoBERTa uses the core structure of BERT but it reduces the number of layers in the model to 6 and uses knowledge distillation Hinton et al. (2015). Knowledge distillation, in this case uses the teacher-student architecture, where the transfer of information takes place from a larger (teacher) neural network to smaller (student) neural network Hu et al. (2023).
- **MiniLM Wang et al. (2020)**: MiniLM does further model compression specifically targeting the self-attention mechanism of the transformer architecture. It performs distillation of the key attention layers, reducing complexity of key attention layers while aiming to preserve their functionality. MiniLM also reduces the size of output embeddings, resulting in a model that is smaller and faster.
- **Multi-qa-mpnet** - This model is trained on on a smaller dataset with multiple question and answer sequences.

4.2 Similarity Score (FAISS)

Once the embeddings were generated, our objective was to develop an efficient system for the retrieval of nearest neighbors in high-dimensional space. Traditional search methods often struggle with high-dimensional data due to "curse of dimensionality." As the number of dimensions increases, the volume of the space increases so rapidly that the available data becomes sparse, making reliable neighbor identification challenging. Additionally, exhaustive searches over large datasets are computationally prohibitive, leading to increased latency, resource utilization, and a suboptimal user experience Radovanović et al. (2009).

To address these challenges, we utilized FAISS (Facebook AI Similarity Search), an optimized library for fast and scalable similarity search and clustering of dense vectors Douze et al. (2024).

Our approach involved constructing a FAISS index to store embedding vectors, where each vector represents the class details for each course at Georgia Tech. The details of how the the embedding vector is created are in Vectorization sub-section 5.1. The parameters were derived as outlined in the Data section 3. FAISS stores vectors without any compression and also has the ability for GPU acceleration.

To respond to user queries, we embed the search input and perform a top-K similarity search within the index, identifying the embeddings most closely aligned with the query. Using these embedding IDs, we mapped the results back to their corresponding courses, which are then presented to the user.

4.3 Collaborative Filtering/Adaptive Recommendations

Collaborative Filtering is an important component of modern recommendation systems. This technique allows to use user rating for every user to cluster users and provide personalized recommendations to them based on their group.

To perform collaborative filtering, each user has their own "user vector" which is a one dimensional vector of length of number of classes. The "user vector" stores normalized ratings given by the user to various courses, scaled in the range of [0, 1]. When a user

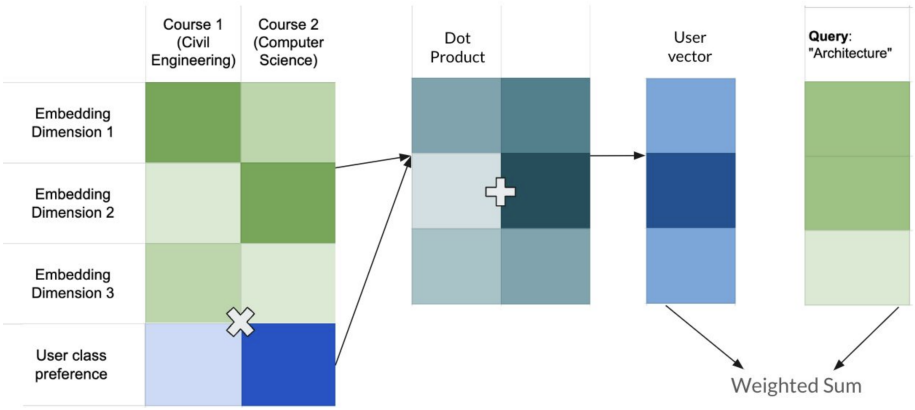

Figure 1: Collaborative Filtering: Each class has an embedding vector, and the user's preferences are represented as a vector. The user vector is computed as the dot product of their preferences with the class vector. This result is then weighted by the Query vector to perform the final search.

submits a query for course recommendations, the system combines their embedded query with the user vector using a weighted addition, producing a refined vector for similarity-based search.

While our current implementation focuses on individual user vectors, collaborative filtering in large-scale applications often clusters users dynamically to prevent every user having their own vector and instead assign vectors to clusters. As our user base expands, we plan to implement such clustering to enhance recommendation accuracy and scalability.

4.4 Course Description Enhancement through Generative Language Models

In addition to generating recommendations, it's important to provide detailed information about the course including the nuanced content, the practical applications of the course content, and specific learning outcomes of the course. This detailed information would help in:

- Reducing uncertainty about course relevance to academic or career goals.
- Reducing the lack of understanding regarding what the depth and scope of the course content would be.
- Reducing the risk of making uninformed course registration decisions.

Thus, our project also addresses information gaps through the generative fine-tuned Llama language model. By utilizing the pre-trained **Llama-3.2-1B** model and further training it on Georgia Tech's specific course data, BUZZ-AI also works as an intelligent system to generate comprehensive, context aware course descriptions that go beyond the standard catalog entries. By training Llama further specifically on Georgia Tech's course dataset, it has been refined so that it does not hallucinate. The model's tendency to generate hallucinated content has been significantly reduced and factually correct descriptions are witnessed that are closely aligned with the original courses data.

Through the Llama model, Buzz-AI expands the course recommendations into more informative descriptions. The user can pass in any additional prompt (even a question) to get a deeper understanding of what the course would cover, its key topics, its learning outcomes, and its content's practical applications. This helps students in understanding better the course's academic and professional significance.

The Llama model utilizes its learned contextual understanding from the training data to generate a detailed, in-depth description that can provide rich insights into the course recommendations. This would allow students to make more informed decisions when selecting courses to take.

5 Architecture

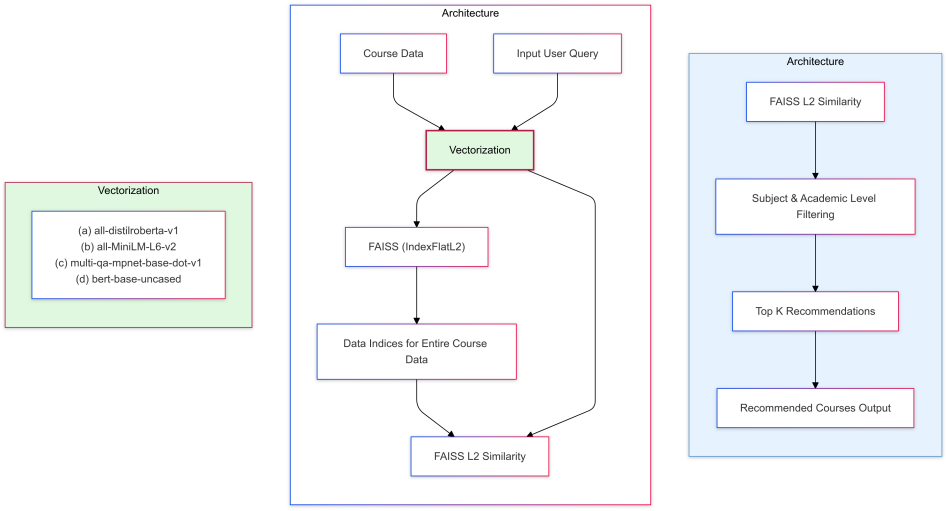

Figure 2: Buzz AI Model Architecture

The architecture above leverages the power of vectorization, nearest neighbor search, data indexing, and targeted filtering to provide robust and efficient course recommendations. Several interconnected components work together to generate personalized course recommendations for users. The architecture works in the following way:

5.1 Vectorization

The vectorization component of the architecture takes in the course data and applies a vectorization technique out of the ones mentioned to transform the textual course data into numerical vector representations. These vector representations capture the semantic and contextual information of the course content.

For each class, the vectorization is done for the following stream of text:

```
CourseVector =  "Name:" + (Prof Name)  + ". Course Description:"
+ (Course Description) +  ". Syllabus Text: "
+ (Syllabus Text) + ". Professor Background: " + (Professor Bio & Research Area)
```

5.2 Data Indexing using FAISS (IndexFlatL2)

The next key component is the Facebook AI Similarity Search IndexFlatL2 mechanism. FAISS is an efficient library for nearest neighbor search and it is crucial for the recommendation system. The vectorized course data is indexed using the FAISS IndexFlatL2 algorithm that allows for fast and scalable storage and similarity search operations. Then, the FAISS index is is created for the vectors of the entire course dataset. The index stores the vectors which have all the metadata and attributes about the courses, such as course titles, descriptions, subject areas, academic levels, and other relevant information. This index mechanism enables efficient retrieval and filtering of course data during the recommendation process.

5.3 Input User Query

When a user submits a query, the input is first processed through the Vectorization component to convert it into a numerical vector representation. Then the index for the user query is generated and passed to the FAISS L2 Similarity algorithm.

5.4 FAISS L2 Similarity

The FAISS L2 Similarity metric is based on the Euclidean distance between indices and is used to identify the courses that are most similar to the user's query. FAISS L2 Similarity is calculated by leveraging the pre-built FAISS index.

5.5 Subject and Academic Level Filtering

The next step in the architecture involves course filtering based on the subject and academic level. The courses identified by FAISS are filtered to get the ones that match the requirements specified in the user query.

5.6 Recommendations

Finally, the architecture outputs the Top K Recommendations, which are the most relevant courses for the user. Ranking and sorting mechanisms are applied to present the user with the most personalized and meaningful course suggestions.

6 Challenges

Integrating Special Topic Courses

- A challenge faced was integrating Special Topics (8803 / 4803) classes as these courses are specialized and sometimes from unique research areas but are all offered under a generic name *Special Topics*.
- To address this, multiple web scrapers were built to scrape descriptions, class names, professor bios and syllabi from different websites like OSCAR, college websites, etc.

Algorithm Selection To Determine Content Similarity

- Initially, KMeans was performed using vanilla scikit-learn. However, significant performance limitations were faced due to scikit-learn's inability to use GPU.
- Consequently, FAISS was chosen instead of using sklearn.

Data Limitations

- Lack of comprehensive data on student preferences and course feedback limited the ability to further improve the model's accuracy.
- To mitigate this, future work involves collecting detailed data and further fine-tuning to provide better results.
- The objective was to train on the syllabi of all classes offered at Georgia Tech. However, due to time and computational constraints this task was not pursued. Moving forward, steps will be taken to address these limitations.

7 User Flow & Examples

Buzz-AI takes in 3 Parameters:

- `-q`: User Query - `String` (Mandatory)

- `-l`: Level - ``grad", ``undergrad" (Optional)
- `-c`: Courses - ``MGT", ``CS", ``CSE" (Optional)

User query is a String. Level is Undergrad, Grad, or None. Courses are a list of subjects interested in: "MGT", "CS" - can also be None.

7.1 Example 1

```
python3 search.py -q "Mutex Locks" -c "CS, CSE" -l "grad"
```

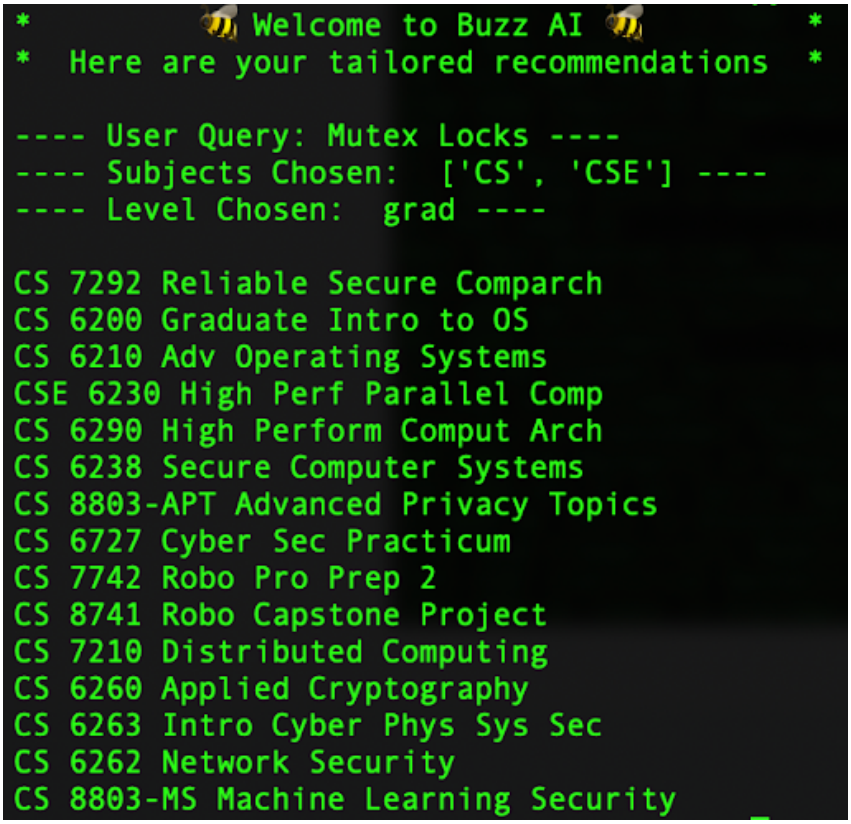

Figure 3: Example 1 - Mutex Locks Query

For this example, 3, we have a niche query like *"Mutex Locks"*. And we're just searching for CS/CSE grad classes. If we carefully look at the outputs, we can safely assume this is what an expert on Georgia Tech courses/Academic advisor would expect.

7.2 Example 2

```
python3 search.py -q "Poker Theory" -l "undergrad"
```

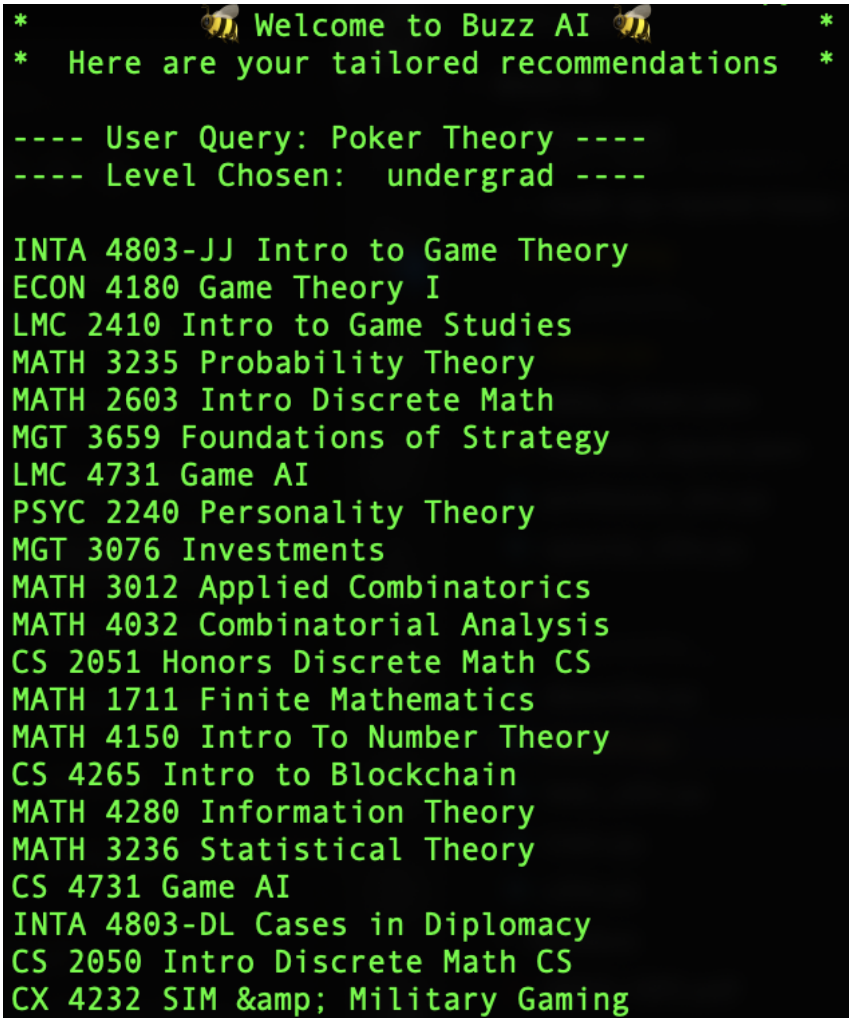

Figure 4: Example 2 - Poker Theory Query

For this example, 4, we have a very specific query *"Poker Theory"*. Buzz-AI is able to offer tailored recommendations on the classes you could potentially take as an undergrad if you're interested in learning about Poker Theory.

7.3 Example 3

For this example, 5, a new user logs in and asks for courses related to *"Architecture"*. The user gets recommendation of classes mostly related to the Architecture Major, and one class of ECE. The user rates all Architecture classes 1 (lowest rating possible), and ECE class 5 (highest rating). The user when asks for *"Architecture"* the next time, then they get most classes related to Computer Architecture!

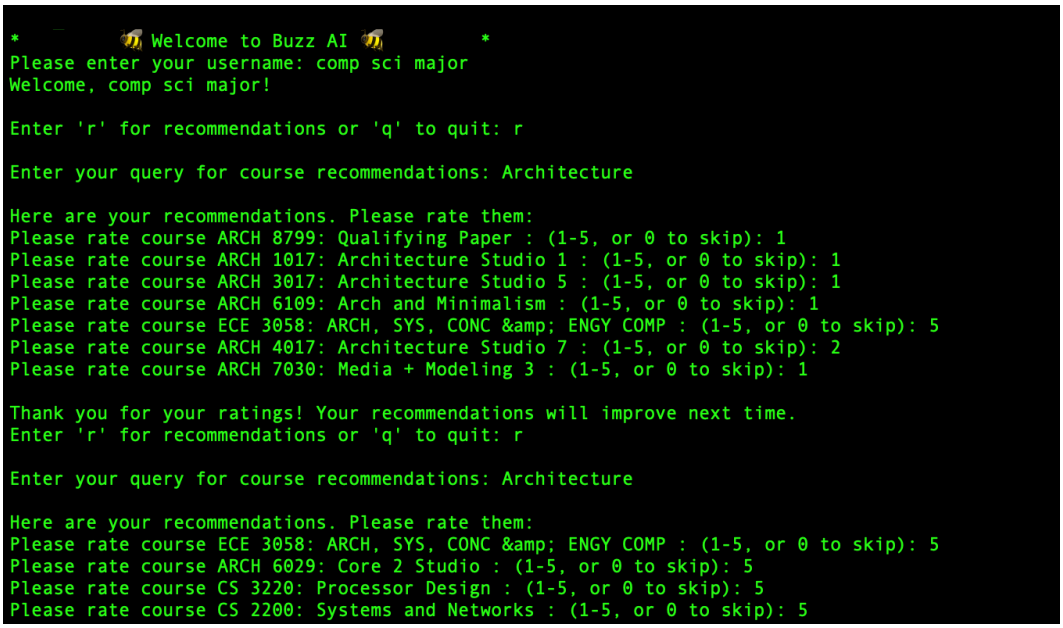

Figure 5: Example 3 - Adaptive Recommendations

Queries like these can help students find classes about niche interests and find undiscovered classes. Example 1 and 2 are great examples of how context aware Buzz-AI is, and the fact that it can operate on extremely specific queries like “Mutex Locks” and “Poker Theory”. Example 3 represents its user aware understanding, and providing help in user’s context.

8 LLM Evaluation & Testing

8.1 Metrics

We held a user study to build a ground truth dataset where we provided users with 20 queries and asked them to provide the top relevant classes. We then utilized the majority vote to map the query with top-5 vectors. To calculate accuracy, we find the intersection of the user suggested answers and model predicted answers divided by 5. The evaluation then measured the percentage alignment between model outputs and the established ground truth across the four different embedding models.

To compare collaborative filtering accuracy, we consider each user’s individual course ranking for 20 queries. We use the first 5 of these queries to generate output from the model, and rank the model’s output based on the user data. We then send the next 15 queries without ranking any outputs. We calculate the accuracy on the outputs from the last 15 query outputs for the model.

8.2 Results

Table 1: Vanilla Recommendation with Embedding Models

Embedding Model	**Accuracy (%)**	**# Parameters**
all-MiniLM-L6-v2	84.10	1.17B
all-distilroberta-v1	81.81	1.12B
multi-qa-mpnet-base-dot-v1	47.73	215M
bert-base-uncased	36.36	110M

Table 2: Collaborative Filtering Recommendation with Embedding Models

Embedding Model	**Accuracy (%)**	**# Parameters**
all-MiniLM-L6-v2	93.40	1.17B
all-distilroberta-v1	90.15	1.12B
multi-qa-mpnet-base-dot-v1	73.21	215M
bert-base-uncased	65.79	110M

8.3 Analysis

Firstly, we see a linear relationship between accuracy and number of parameters for embedding models. The degrading accuracy can be attributed to the model's inability to capture and create meaningful embeddings in words.

The poor performance of multi-qa-mpnet-base-dot-v1 was expected since the model is built for question answering and not semantic search. Similarly, bert-base-uncased was built primarily for classification tasks and hence stores dense syntactic information instead of semantic information which is not ideal for this task.

All-MiniLM-L6-v2 and all-distilroberta-v1 were both created primarily for sentence embedding and hence have a good performance. While all-MiniLM-L6-v2 have higher parameters, their embedding length is smaller giving a trade off between the space of index required versus the model required to generate query embedding.

9 Project Goals & Impact

The primary objective of this project was to develop an intelligent, personalized course recommendation system to provide users with highly relevant and tailored educational course suggestions. In this project, the following was acheived:

- Several embedding models (MiniLM, DistilRoBERTa, MPNet, BERT) were utilized to transform textual course information into meaningful vector representations and capture the semantic nuances of the text.
- An efficient course matching mechanism was implemented using FAISS to measure text similarity.
- Advanced filtering mechanisms were incorporated to generate relevant recommendations.
- Personalized course recommendation was achieved using Collaborative Filtering.

The transformer-based course recommendation system has significant potential to transform how students and learners discover and select educational courses. The recommendation

system utilizes state-of-the-art content matching and personalizes recommendations. It increases educational accessibility, and is extremely efficient.

9.1 Future Work and Scalability

In future work, we aim to explore the following:

- Incorporation of user feedback to fine-tune embeddings.
- In the response, receive detailed course content and learning outcomes.
- Add enhanced filtering options to provide even more tailored results.
- Scaling the model by clustering users based on their preferences.
- Provide users with a course prerequisite list in the response.
- Expansion to courses outside of Georgia Tech.
- Detailed course plan based on career goals and query.

10 Author Contributions

All authors made equal contributions to the design and implementation of BUZZ-AI. This includes collaborating on the development of the system architecture, selecting the appropriate methods, brainstorming, and improving the system's performance. Additionally, each author participated in testing, debugging, and refining BUZZ-AI. Through close teamwork and shared responsibilities, the authors collectively shaped BUZZ-AI from conceptualization to state-of-the-art course recommendation system.

References

Stephane Aroca-Ouellette and Frank Rudzicz. On losses for modern language models. *CoRR*, abs/2010.01694, 2020. URL https://arxiv.org/abs/2010.01694.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.

David Esteban et al. Investigating course choice motivations in university environments. *Smart Learning Environments*, 8(1):1–31, 2021.

GT-Scheduler-Contributors. Crawler v2. https://github.com/gt-scheduler/crawler-v2, 2024.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Chengming Hu, Xuan Li, Dan Liu, Haolun Wu, Xi Chen, Ju Wang, and Xue Liu. Teacher-student architecture for knowledge distillation: A survey, 2023. URL https://arxiv.org/abs/2308.04268.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. The emergence and influence of hubs, 2009.

Jiarui Rao and Jionghao Lin. Ramo: Retrieval-augmented generation for enhancing moocs recommendations, 2024. URL https://arxiv.org/abs/2407.04925.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer, 2019. URL https://arxiv.org/abs/1904.06690.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. URL https://arxiv.org/abs/2002.10957.

Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, and Qing Li. Recommender systems in the era of large language models (llms), 2024. URL https://arxiv.org/abs/2307.02046.