AALTO-YLIOPISTO
AALTO UNIVERSITY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

ELEC-E5550 Statistical Natural Language Processing D

Final report

CourseMatch

Building a Course Selection Recommender System Using

Sentence-BERT

HELSINKI, APRIL 2023

Member list & Workload

No.	Full name	Student ID	Percentage of work
1	Matteo Circa	100485156	33%
2	Marco Di Francesco	100632815	33%
3	Tianxing Wu	100589324	33%

At the outset of the project, every team member was actively involved in laying out the roadmap for the project, testing the first BERT model, and gathering valuable references. The team's collective efforts were instrumental in setting the project off to a great start.

As the development phase began, Tianxing took the initiative to bootstrap both the backend and the frontend. With the project framework in place, Matteo stepped in and made improvements to both components, while also working on fine-tuning. Every team member played a role in reviewing and refining the code, ensuring that the system was as robust and efficient as possible.

In addition to development, Marco was responsible for the critical evaluation phase of the project. Overall, the team's collaborative efforts and individual contributions were key to the project's success, and we are proud of what we have accomplished together.

Contents

Abstract

Our team has developed a truly revolutionary course selection recommender system that uses cutting-edge natural language processing and machine learning techniques to help students discover the most relevant and interesting courses to take. We are proud to say that our system, which employs Sentence-BERT (SBERT) [1] to match user queries with a database of course descriptions accessible via the Aalto API, outperforms other state-of-the-art recommendation techniques by a discrete margin, including baseline methods usually applied to this field, like basic filter techniques.

But that is not all. We believe in the power of human feedback to improve our system. That is why we made sure that the human was in the loop during the development of the model. We expect to improve performance even further with the first feedback coming in. Our approach will lead to even better course recommendations and an improved educational experience for students.

Our work highlights the potential of natural language processing and machine learning techniques in enhancing educational experiences and supporting student success. We believe that the use of such techniques is the way forward in improving the quality of education and making it more accessible to a wider range of students. We are thrilled about the possibilities of our system and cannot wait to see it used in the real world.

Go and try it now at this link or go to the next url: `https://course-match-ui.pages.dev`.

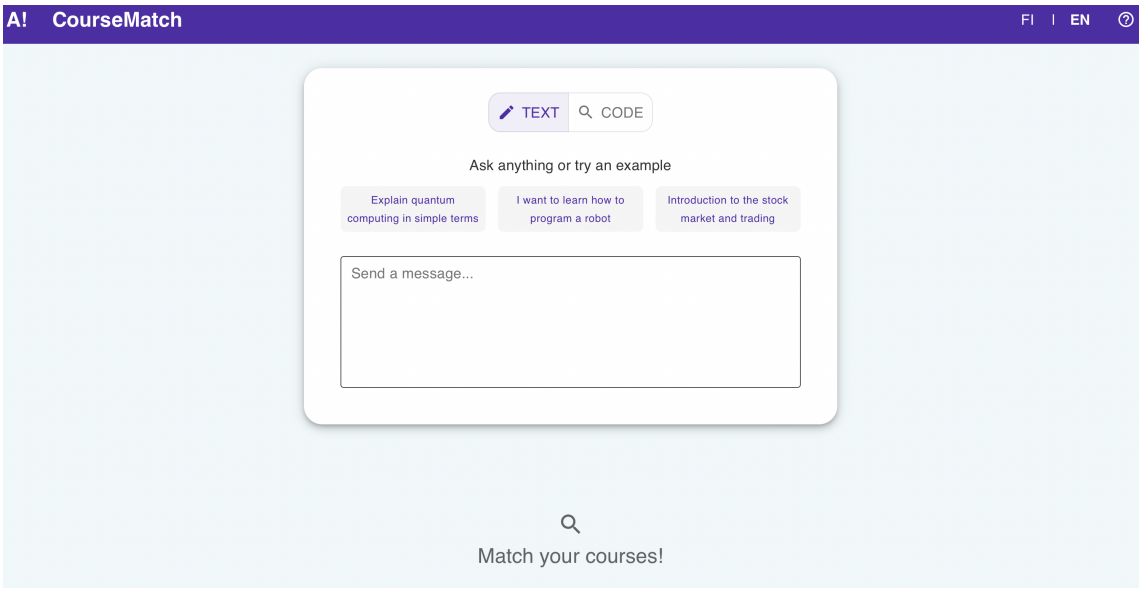

Figure 1: CourseMatch landing page.

1 Introduction

In today's world, the abundance of available information and the rapid pace of technological advancements make it increasingly challenging to find relevant and high-quality resources. In the field of education, students face the daunting task of selecting courses that align with their academic goals and interests. However, with limited information and time, this can prove to be a difficult challenge.

There are different approaches to building a course selection recommender system. Some popular approaches include:

1. Collaborative filtering. This approach involves finding similar users and recommending courses that have been highly rated by those similar users.

2. Content-based filtering. This approach involves analyzing the content of courses and recommending courses that are similar in content to courses that the user has previously shown an interest in.

3. Hybrid approach. This approach combines collaborative filtering and content-based filtering techniques to provide a more personalized recommendation to the user.

4. Matrix factorization. This approach involves decomposing the user-item rating matrix into two lower-dimensional matrices to identify latent features that are common across users and items.

5. Deep learning-based approaches. These approaches involve training a neural network to learn the user-item interactions and generate personalized recommendations based on that.

In this project, we use a content-based filtering approach with Sentence-BERT (SBERT) to build a course selection recommendation system based on word embedding techniques.

SBERT is a state-of-the-art method for embedding sentences into high-dimensional vector space, making them comparable and allowing for semantic similarity calculations. By using SBERT to analyze the textual descriptions of available courses, we can match a user's search query with the most relevant courses and recommend them to the user.

Our proposed system is designed to provide students with personalized course recommendations based on their academic interests and preferences. To achieve this, we first created a dataset containing descriptions of the courses we aim to suggest, then semantic embeddings to understand the meaning and context of the course descriptions, and a similarity measurement to calculate the degree of resemblance between courses. We leverage the power of SBERT and utilize the Aalto API to access a vast database of course descriptions.

Overall, this project aims to demonstrate the efficacy of SBERT for building personalized course selection systems and provide a practical solution to a relevant and challenging problem in education.

2 Methods

The paper "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks" (Reimers et al., 2019)[1] discusses various research projects related to the study of semantic textual similarity (STS) between sentences. One such project involves the development of a modification of the BERT network called Sentence-BERT (SBERT). SBERT uses siamese and triplet network structures to derive semantically meaningful sentence embeddings, which can be compared using cosine-similarity. This reduces the time required to find the most similar pair from 65 hours to just 5 seconds, while maintaining the accuracy of BERT. SBERT outperforms other state-of-the-art sentence embedding methods on common STS tasks and transfer learning tasks.

The paper also compares the performance of three different systems for generating sentence embeddings: RoBERTa, SBERT, and BERT. RoBERTa outperforms SBERT and BERT on the STS benchmark dataset, while SBERT outperforms BERT on the NLI dataset and STS benchmark dataset. BERT performs better than SBERT on the AFS corpus after being fine-tuned on the STSb dataset.

Additionally, the paper discusses the results of an ablation study of different aspects of SBERT, including different pooling strategies and concatenation methods. The authors found "strong empirical evidence for the quality of SBERT sentence embeddings".

2.1 BERT vs. Word Embedding: Understanding the Differences

BERT (Devlin et al., 2018)[2], which stands for Bidirectional Encoder Representations from Transformers, is a state-of-the-art pre-trained language model developed by Google. It is designed to better understand the context and nuances of natural language by training on large amounts of text data. BERT uses a bidirectional transformer architecture, which allows it to take into account the context of a word by considering both the words that precede and follow it in a sentence.

In comparison, word embedding is a general framework for representing words as vectors in a high-dimensional space. It is a technique used in natural language processing to capture the semantic relationships between words and their context. Word embedding models like Word2Vec and GloVe use unsupervised learning to learn word representations from large text corpora.

While both BERT and word embedding aim to capture the context and meaning of words, BERT goes a step further by also taking into account the specific context of a word in a sentence. This makes BERT more effective in tasks like natural language understanding, sentiment analysis, and question-answering.

2.2 SBERT Network Architecture

SBERT is a modification of the BERT network that uses siamese and triplet network structures to derive semantically meaningful sentence embeddings. The network is built by taking a pre-trained BERT model and fine-tuning it using a siamese or triplet network structure.

In the siamese structure, two copies of the pre-trained model are used to encode two different sentences, and the similarity between the resulting embeddings is calculated [1] (Figure 2). In the triplet structure, three copies of the pre-trained model are used to encode an anchor sentence, a positive sentence (similar to the anchor), and a negative sentence (dissimilar to the anchor). The network is trained to maximize the distance between the anchor and negative sentence embeddings, while minimizing the distance between the anchor and positive sentence embeddings. This results in sentence embeddings that capture semantic similarity between sentences. The embeddings can be compared using cosine similarity to find the most similar pair of sentences.

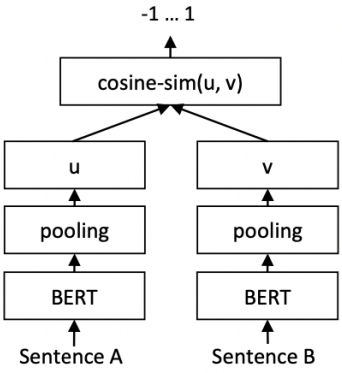

Figure 2: SBERT architecture at inference, for example, to compute similarity scores.

2.3 Semantic Search

The concept of semantic search involves understanding the context of a search query in order to improve search accuracy. Unlike traditional search engines which only rely on lexical matches, semantic search can also identify synonyms to enhance the search results.

The idea behind semantic search is to embed all entries in your corpus, whether they be sentences, paragraphs, or documents, into a vector space. At search time, the query is embedded into the same vector space and the closest embeddings from your corpus are found [3] (Figure 3). These entries should have a high semantic overlap with the query. Semantic search is a useful technique for finding similar items. For example, it can be used to find similar publications or to identify similar questions and answers, as in our case.

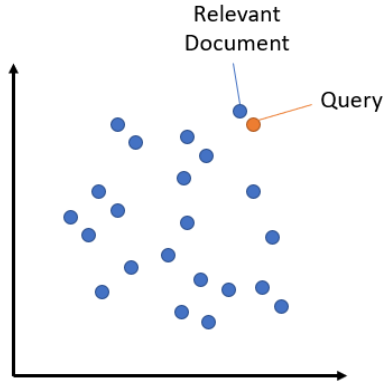

Figure 3: Semantic Search illustration.

Choosing the right model for your task is critical. In our case, the suitable models were for asymmetric semantic search. Asymmetric semantic search is used when you have a short query, such as a question or some keywords, and you want to find a longer paragraph that answers the query. For example, if you search for "*What is machine learning*" and you want to find the paragraph "*Machine learning (ML) is a field devoted to understanding and building methods that let machines learn ...*".

2.4 Asymmetric Semantic Search Models

Suitable models for asymmetric semantic search are the Pre-Trained MS-MARCO Models. These models have been trained on the MS-MARCO corpus, which is a collection of over 8.8 million passages. They can retrieve information by encoding queries and passages and computing similarity scores. There are different models available, with varying levels of performance measured by the metrics NDCG@10 and MRR@10. The specific model to use depends on the task at hand. The models are tuned for either cosine similarity or dot product.

In the third version of the models, performance has been improved by using a cross-encoder to classify retrieved passages. This enhancement led to an overall improvement in model performance [3].

These models from the Sentence-Transformers library are a valuable resource for conducting semantic searches and retrieving useful information.

3 Experiments

For our course selection recommendation system, we conducted a series of experiments to evaluate the performance of our model. The code is available in the repository CourseMatchPrototype, accessible from our organization's link on GitHub at this or at the following url: `https://github.com/SNLP-project-team2023`.

3.1 Data Collection and Cleaning Process

We have approximately 2,300 courses with content, workload, learning outcomes, and assessment methods. These courses are fetched using the Aalto API Gateway and stored locally.

Our cleaning process involves the following steps:

- Removing duplicated courses and those without an English description (since the models are mainly designed for this language). After this step roughly 1000 courses are left.
- Combining learning outcomes with course content to create a single description that can be used as a sentence for SBERT.
- Decoding HTML.

3.2 Challenges in Using BERT for Sentence Representation

The search began with the simplest possible model, namely `bert-base-uncased`. The set of hidden states in this model, stored in the `hidden_states` object, is a little bit too challenging. That is 219,648 unique values just to represent a single sentence!

Now, what do we do with these hidden states? We would like to get individual vectors for each of our tokens (words), or perhaps a single vector representation of the entire sentence, but for each token in our input we have 13 separate vectors, each of length 768. To get the individual vectors, we will have to combine some of the vectors from the neural network layers - but which layer or combination of layers provides the best representation? Unfortunately, there is no single answer... [4]

Several application-dependent strategies are available to obtain a single vector for the entire sentence, but a simple approach is to average the second to last hidden level of each token, producing a single vector of length 768 for the entire sentence, in our case the individual course descriptions. From these vectors, it was straightforward to compute similarity with a user query, first encoding the query with BERT, as for courses, and then measuring cosine similarity with all courses and returning the most "similar" ones. However, the results of this technique were not satisfactory, and the matching did not work very well. This fact is confirmed by the words of author Jacob Devlin of BERT: "*I'm not sure what these vectors are, since BERT does not generate meaningful sentence vectors. It seems that*

this is doing average pooling over the word tokens to get a sentence vector, but we never suggested that this will generate meaningful sentence representations."

3.3 SBERT Multi-QA and MS-MARCO Models

The research then continued with SBERT, explained in detail above, specifically on the asymmetric semantic search models, such as the Pre-Trained MS-MARCO Models, and the Multi-QA Models, trained on 215M question-answer pairs from various resources and domains, including StackExchange, Yahoo Answers, Google & Bing search queries, and many others.

To conduct as heterogeneous experiments as possible and better evaluate the performance of these models on our task, we considered two models among the Multi-QA models, the multi-qa-mpnet-base-dot-v1, tuned to be used with dot-product, and the multi-qa-mpnet-base-cos-v1, tuned to be used with dot-product, cosine-similarity and Euclidean distance. Two other models then from the MS-MARCO ones, the msmarco-distilbert-base-tas-b, again with dot-product, and the msmarco-distilbert-base-v4, with cosine-similarity. These models are the best performing in their respective categories.

After a brief evaluation, it was determined that the most effective model for the task at hand was `multi-qa-mpnet-base-dot-v1`. However, a detailed analysis of the performance of this model is provided in the subsequent section.

A simplified version of the code is given below.

```
from sentence_transformers import SentenceTransformer, util

# load the model
model = SentenceTransformer('sentence-transformers/msmarco-distilbert-base-tas-b')

# calculate embeddings for all courses
corpus_embeddings = model.encode(courses['description'].tolist())

# calculate embeddings for the user query
query = "A course that teaches the basics of how the brain works and how the different
electronical signals and neurons interact with each other. I want to get a basic
understanding of the dynamics and how these concepts are related to artificial neural
networks."
query_embeddings = model.encode(query)

# find the 5 most similar courses
```

```
hits = util.semantic_search(
    query_embeddings, corpus_embeddings, score_function=util.dot_score, top_k=5)
```

The following results are obtained for this query:

- Structure and operation of the human brain D.
- Genesis and Analysis of Brain Signals D.
- Information Processing in Neural Circuits D.
- Artificial Intelligence D.
- Analysis and Design of Electronic Circuits.

As can be seen, the first three are very relevant while the last two less so. Hence the next step, fine-tuning.

3.4 Fine-tuning our own model

The SentenceTransformers framework is also a useful tool for fine-tuning sentence/text embeddings models. It provides building blocks that can be combined to create embeddings that are customized for specific tasks. However, there is no single loss function that is suitable for all tasks. The choice of loss function depends on the available training data and the target task. To fine-tune a network, it's important to determine which sentence pairs are similar and should be close in vector space, and which pairs are dissimilar and should be far apart. The simplest way to do this is to have sentence pairs annotated with a similarity score, which can be used to train the network with a Siamese Network Architecture.

The implementation is as follows:

```
from sentence_transformers import SentenceTransformer, InputExample, losses
from torch.utils.data import DataLoader

# define the model
model = SentenceTransformer('msmarco-distilbert-base-tas-b')

# define the train examples
train_examples = [InputExample(texts=['My first sentence', 'My second sentence'], label=0.8),
    InputExample(texts=['Another pair', 'Unrelated sentence'], label=0.3)]

```

```
# define the train dataset, the dataloader and the train loss
train_dataloader = DataLoader(train_examples, shuffle=True, batch_size=16)
train_loss = losses.CosineSimilarityLoss(model)

# tune the model
model.fit(train_objectives=[(train_dataloader, train_loss)], epochs=1, warmup_steps=100)
```

We will collect user feedback for the entered question pairs and selected courses, with a score of 0 if the match was not satisfactory and 1 if it was. We will continue to collect feedback to improve the recommendations.

4 Results

We performed the evaluation of the results using two techniques. Firstly, we conducted an empirical evaluation of multiple models and compared their performance on multiple queries. Secondly, we utilized dimensionality reduction techniques to cluster courses by department and compared their performance. Both evaluations can be found in the SentenceTransfiormer_DimReduction notebook.

4.1 Manual Evaluation of Query Results

An initial evaluation of the model was carried out by generating a predetermined set of queries and manually verifying if the first five results contained the relevant courses that matched the query. These queries were intentionally constructed to be challenging for the model to predict, while still being representative of the type of queries a user would make. The queries are presented in Table 1 and the corresponding results in Table 2.

The findings reveal that the `multi-qa-mpnet-base-dot-v1` model performed significantly better than the other models, by matching at least one relevant course for each query in almost all cases, and in most cases, displaying multiple relevant courses.

Query 1	I want to learn about developing 2D video games with Unity. The course should teach me how to create a game from scratch, including the programming.
Query 2	An Art and Contemporary Culture course to explore the dynamic relationship between art, culture, and modern society.
Query 3	A course to provide a comprehensive introduction to the world of stock market investing.
Query 4	Programming in Python, data structures, algorithms, and object-oriented programming.
Query 5	A course that teaches the basics of how the brain works and how the different electronic signals and neurons interact with each other.

Table 1: List of queries for model comparison.

	Multi-QA dot	Multi-QA cos	MS-MARCO tas b	MS-MARCO base
Query 1	**1**	**1**	0	0
Query 2	**4**	**4**	3	2
Query 3	**3**	0	0	2
Query 4	3	1	2	**4**
Query 5	**1**	**1**	0	**1**

Table 2: Number of courses that match the query in the top five results.

4.2 Clustering Courses with Dimensionality Reduction Techniques

To explore the model performance further, we used dimensionality reduction techniques to reduce the number of dimensions to two. The aim was to see if courses belonging to the same department were clustered together. We selected the ten most relevant departments, each with twenty or more courses. We applied two reduction techniques: UMAP and TSNE. Example results for the model that performed the best, model `multi-qa-mpnet-base-dot-v1`, are displayed in Figures 4 and 5, respectively.

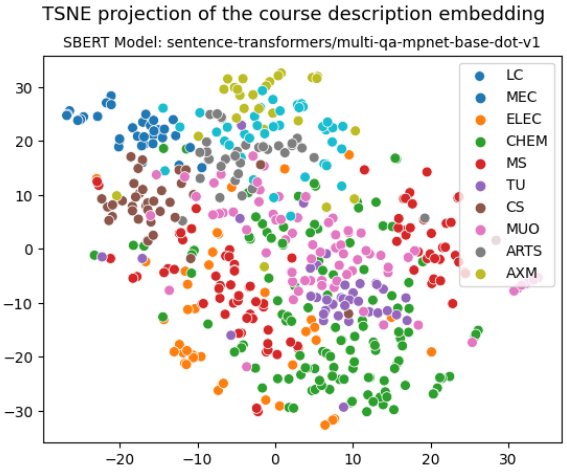

Figure 4: TSNE projection of Multi-QA dot.

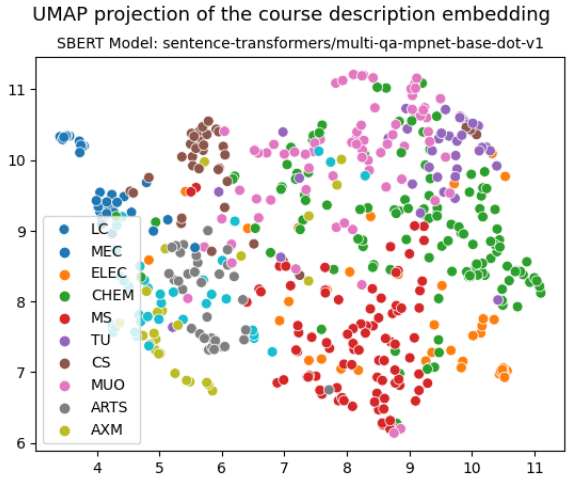

Figure 5: UMAP projection of Multi-QA dot.

We analyzed the projections using two clustering metrics: Silhouette and Calinski-Harabasz. The Silhouette Index (SI) measures how well each point in a cluster is separated from points in other clusters. In our case, this means how well a course, given its department, is separated from other departments. Additionally, we used the Calinski-Harabasz (CH) metric to measure the ratio of between-cluster variance to within-cluster variance. We did not expect a significant distinction since often even courses between different departments have similar content. However, it is still possible to see a clear distinction between courses from different departments. The results are shown in Table 3, and we can conclude that the differences are captured quite well.

	TSNE		UMAP	
	SI	CH	SI	CH
Multi-QA dot	-0.045	**70.290**	**0.055**	**116.397**
Multi-QA cos	**-0.034**	59.640	-0.009	78.974
MS-MARCO tas b	-0.067	48.377	0.028	90.484
MS-MARCO base	-0.052	62.193	-0.016	74.533

Table 3: Model performance on CH and SI metrics.

5 Conclusions

Our experiments show that our course selection recommender system based on SBERT word embeddings performs remarkably well. Given the findings, we consider `multi-qa-mpnet-base-dot-v1` as the best model for the site in production, since the results were consistently accurate.

On the other hand, our system could be outperformed by more complex models, but for the purpose of the project we wanted to keep the processing time very low. This makes it very efficient for practical use, as users can get personalized course recommendations almost instantly.

In this project, we have demonstrated the effectiveness of the SBERT-based course recommendation system. However, there is still room for further improvement. One area of potential improvement is to evaluate the performance of the recommendation system after finetuning with user feedback. Future work could also explore the use of other machine learning techniques to further enhance the recommendation system.

Additional work on this project could include testing the model with queries that include spelling errors. Our user tests showed that most users did not spell the entire sentence correctly. Despite this, the quality of the results was still accurate, but more research is needed to compare the performance of the model with correct and incorrect sentences.

In conclusion, we are proud of the results of our project, which shows that the use of SBERT word embeddings is a highly effective technique for building course selection recommender systems. With the increasing demand for online education and personalized learning, we believe our system has great potential for improving the user experience and helping students find the courses that best fit their needs.

References

[1] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, August 2019. URL https://doi.org/10.48550/arXiv.1908.10084.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019. URL http://arxiv.org/abs/1810.04805. arXiv:1810.04805 [cs].

[3] SentenceTransformers Documentation — Sentence-Transformers documentation, . URL https://www.sbert.net/index.html#.

[4] BERT Word Embeddings Tutorial · Chris McCormick, . URL https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial/#sentence-vectors.