



Decision Support Tool for Program Selection

BY: KALEN GOO, ALYSSA LIU, BRANDON LIN, MAYUR SHIVAKUMAR, ARUN
PRASAD SRINIVASAN MANOHARAN, ADAM PERLIN, JOHN WAIDHOFER,
SNEHITH JONNAIKODE

Agenda

- Background
 - Problem
 - Our Solution
 - Word Embeddings
 - Classification Models and Algorithms
- Methods
 - Data Acquisition
 - Applying Embeddings
 - Applying Classifiers
 - Front-end Interface
- Model Evaluation
- Future Work

Background: Problem

- In the U.S, prospective undergraduate students have to navigate the complex collegiate system
 - 44,000 accredited programs
 - 15.9 million undergraduate students enroll annually
 - 30% of the students switch their major within the first 3 years
- Problem Space: Lack of tools to support and guide their curriculum leading to ill-informed decisions and outcomes
- **How might we support prospective students in selecting the right program?**

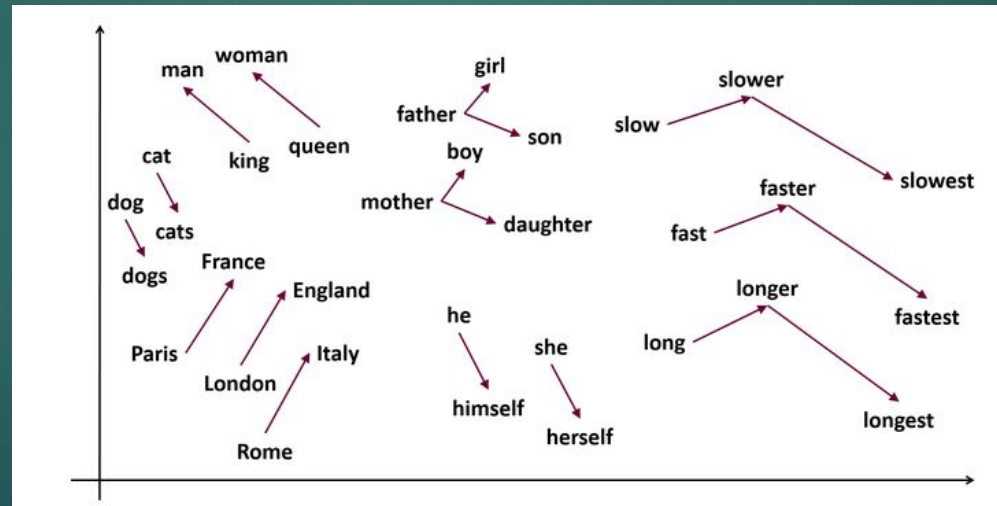
Background: Our Solution

- Prevent students from having to spend hours scouring different course catalogs and read about each program individually
 - Tedious process and not always clear

The decision support tool will help students find the right programs given their interests.

Background: Word Embeddings

- Need to convert words or sentences into a vector for input into neural network
- Want semantically or syntactically similar words to be close in vector space
- Early work: word2vec
 - Learned word similarities on a document basis

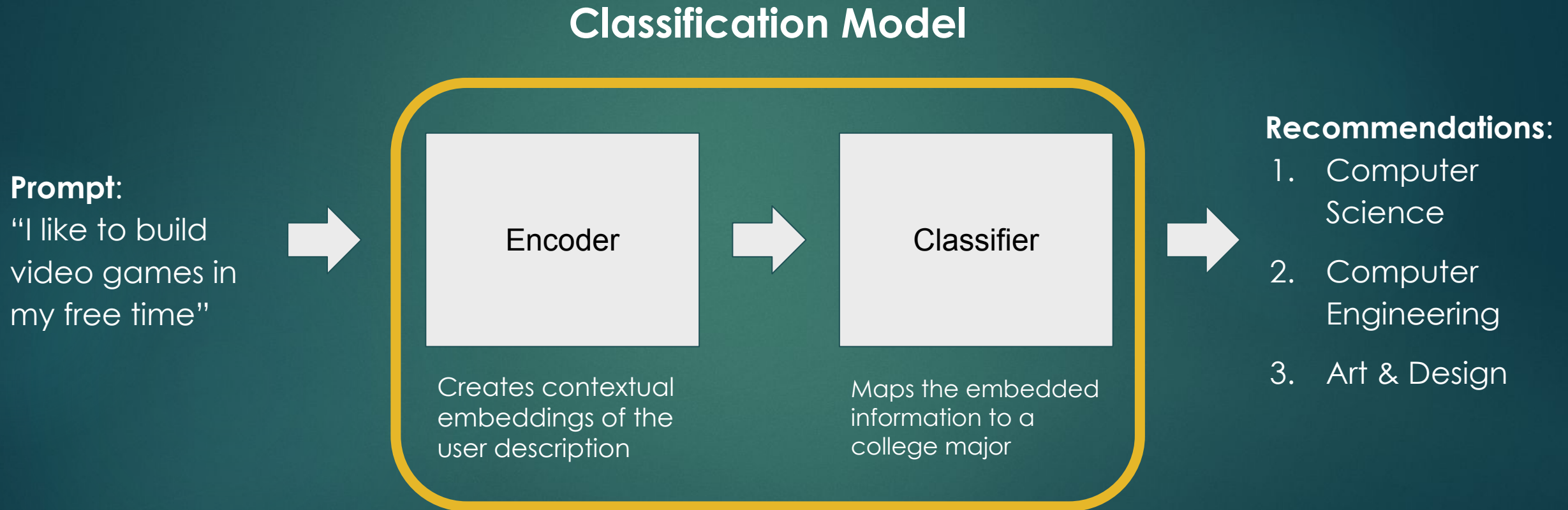


Background: Word Embeddings

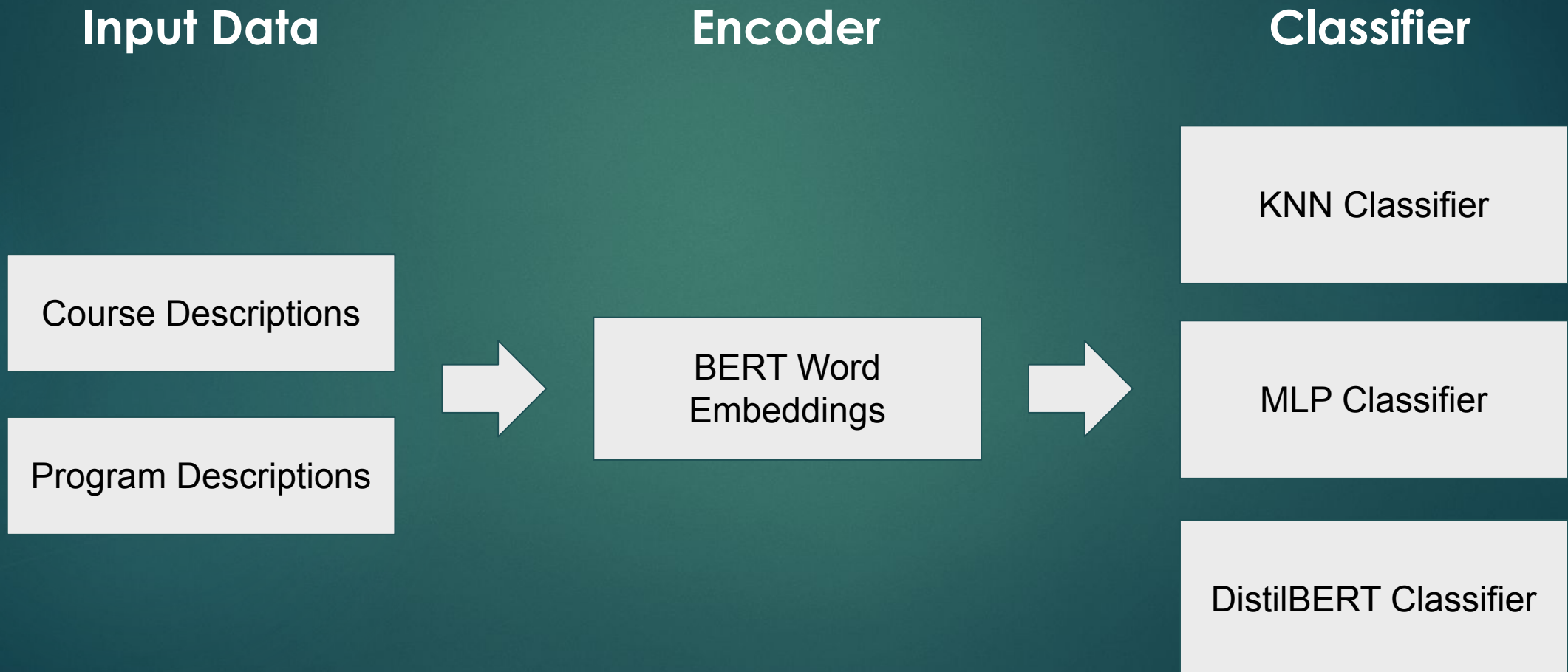
- Large language models capture the context of natural language inputs in their latent space embeddings.
 - Latent embedded representations are used as a proxy for the semantic understanding of natural language.
- Generally, modern language models use an attention-based autoencoder architecture.
 - Through decoupling the encoder model from the decoder, the developers can produce latent embeddings compared numerically.
- Example: Google's BERT model and GloVe

Methods

Decision Support Tool: User Flow



Decision Support Tool: Model Pipeline



Data Acquisition

Course Descriptions

CSC 202. Data Structures.

4 units

Term Typically Offered: F, W, SP

Prerequisite: CPE/[CSC 101](#) with a grade of C- or better; or consent of instructor.

Introduction to data structures and analysis of algorithms. Abstract datatypes. Specification and implementation of advanced data structures. Theoretical and empirical analysis of recursive and iterative algorithms. Software performance evaluation and testing techniques. Not open to students with credit in CSC/CPE 108. 3 lectures, 1 laboratory. Crosslisted as CPE/[CSC 202](#).

CSC 203. Project-Based Object-Oriented Programming and Design.

4 units

Term Typically Offered: F, W, SP

Prerequisite: CPE/[CSC 202](#) with a grade of C- or better or consent of instructor.

Object-oriented programming and design with applications to project construction. Introduction to class design, interfaces, inheritance, generics, exceptions, streams, and testing. 3 lectures, 1 laboratory. Crosslisted as CPE/[CSC 203](#).

CSC 225. Introduction to Computer Organization.

4 units

Term Typically Offered: F, W, SP

Prerequisite: CPE/[CSC 202](#).

Introduction to computer systems. Simple instruction set architecture and the computer hardware needed to implement that architecture. Machine and assembly language programming. 3 lectures, 1 laboratory. Crosslisted as CPE/[CSC 225](#).

CSC 231. Programming for Engineering Students.

2 units

Term Typically Offered: F, W, SP

Prerequisite: [MATH 142](#); [PHYS 121](#) or PHYS 131 or [PHYS 141](#).

Programming techniques and procedures with applications to engineering problems. Introduction to numerical methods and simulation. Credit not allowed for CSC, Software Engineering or CPE majors. Course may be offered in classroom-based or online format. 2 activities.

Program Descriptions

COMPUTER SCIENCE

College of Engineering

Computer science is the study of computers and computer systems including the design and development of software. From smart phone apps to artificial intelligence technology, computer scientists are at the forefront of innovation and key to the advancement of fields such as medicine, economics, finance and more.

SOFTWARE ENGINEERING

College of Engineering

Much of our modern-day world is controlled by software from items in our personal lives like cars, cell phones and computers, to more global areas such as medical devices, business systems or even national defense. Software engineers design, build and maintain the software used in these devices and more.

BERT Embeddings

- Language model built by Google in 2019
- Converts sentences into contextual embeddings with semantic locality

Sentence Embeddings

1. Tokenize sentence
2. Encode tokens into latent space
3. Concatenate the output of the last 4 layers
4. Average the element-wise values for each embedded word

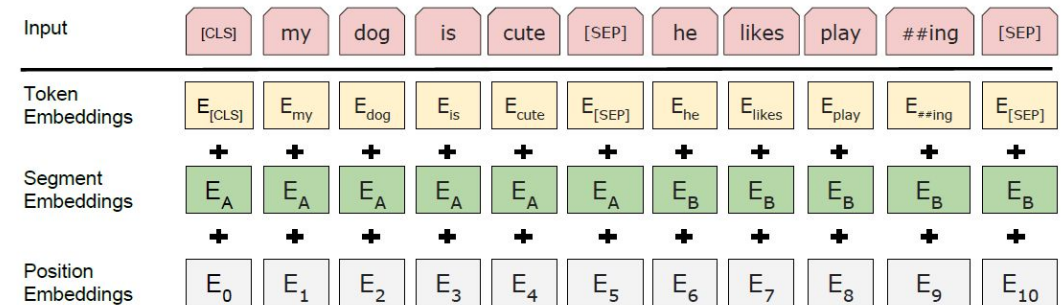
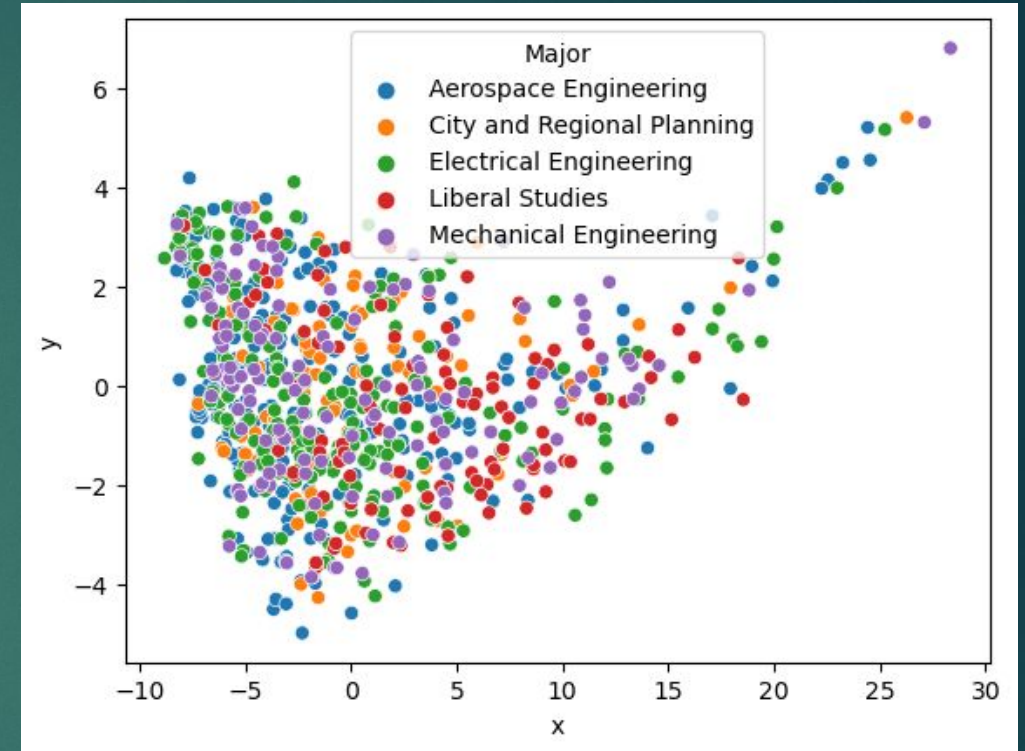


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Source: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

KNN Classifier

- Predict the most frequent class based on the 5 closest embedded sentences
- Advantage: Simple and explainable
- Disadvantage: No clear locality in embeddings by major



Embeddings of top 5 most frequent majors in our dataset reduced to 2D using PCA

Embedded sentences



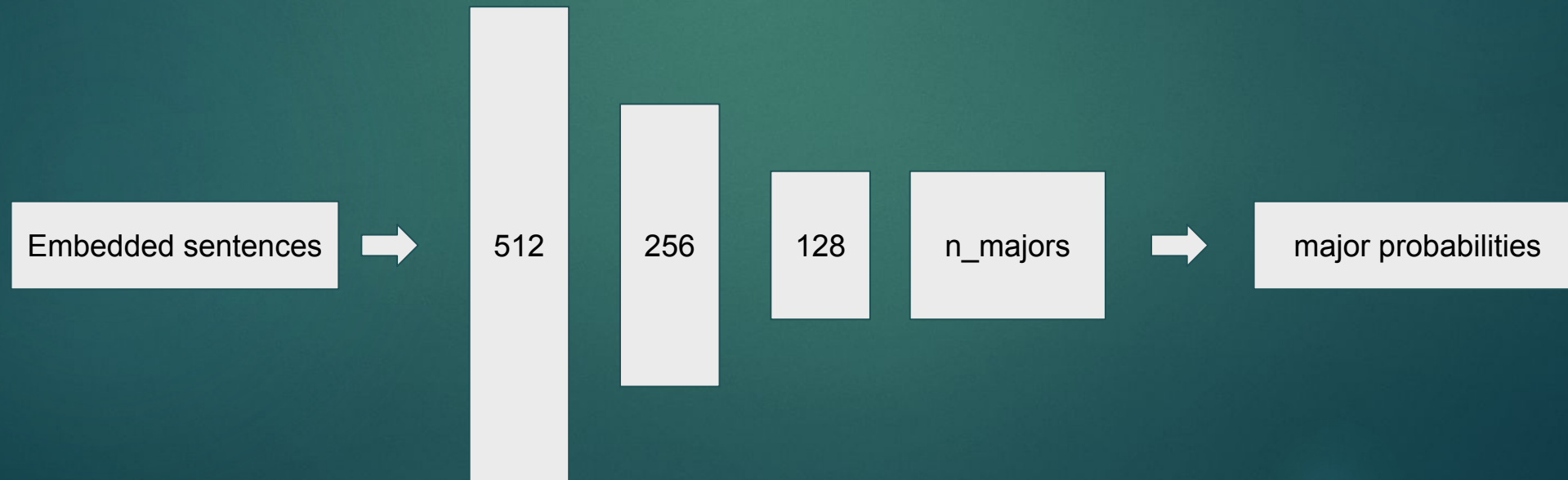
`argsort(nearest_class_count)`



Predictions

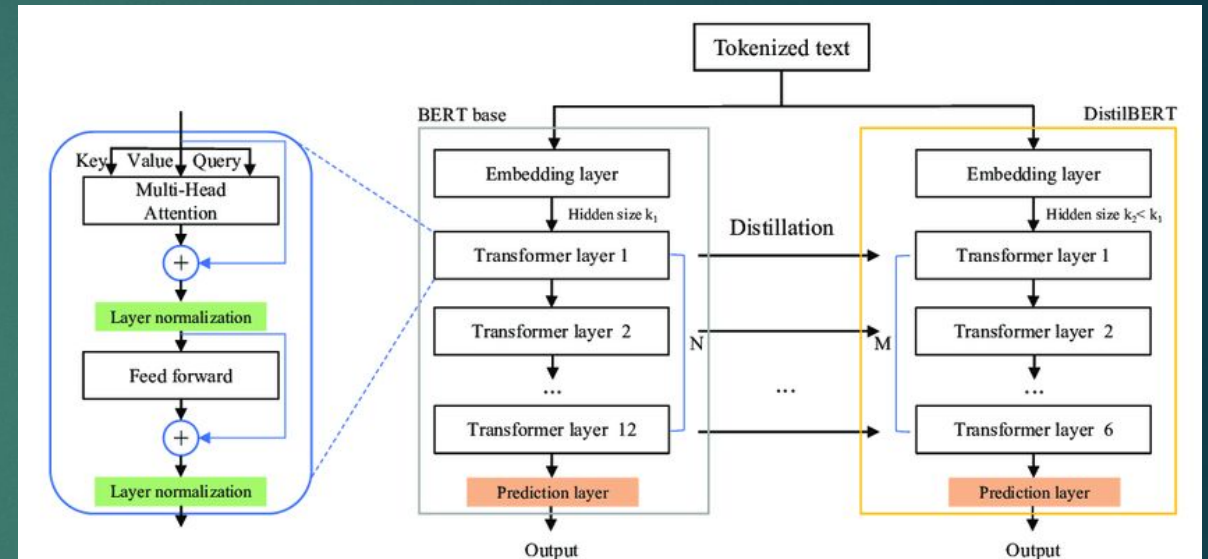
Major MLP Classifier

- Multi-layer Perceptron
- Loss: Categorical-Cross Entropy
- Activation: ReLU
- Class weighing: balanced
- Advantage: Can deal with nonlinear embedding transformations



DistilBert Classifier

- DistilBert is a compressed version of BERT
- Applied transfer learning to fit model to our corpus
- Adapted training process to accept class weights
- Loss: Weighted Cross-entropy
- Advantage: Trains both classifier and embeddings in tandem



Source: Alhassan Mabrouk



Model Evaluation

Evaluation Parameters

All Models

- 80/20 train-test split
- predict top 40 most frequent classes (majors) ~ 3,000 examples

KNN

- $n=5$

Major MLP

- epochs=200

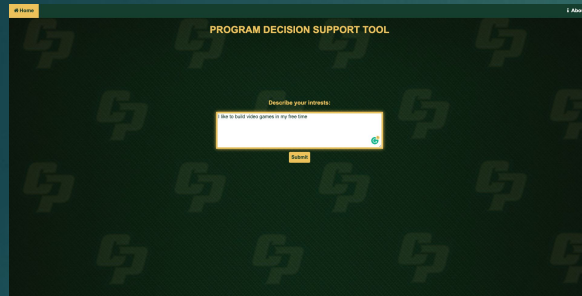
DistilBert

- epochs=25

Results

Model	Precision	Recall	Accuracy	Accuracy (Top 3 Programs)
BERT + KNN	0.29	0.28	0.28	0.47
BERT + Major MLP	0.52	0.49	0.49	0.71
DistilBert Classifier	0.59	0.55	0.55	0.74

Decision Support Tool Architecture



User Input



Flask API

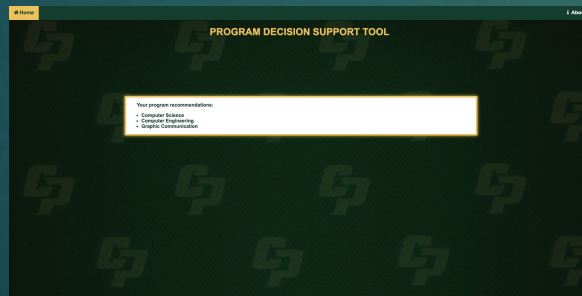
Processed
Input



Decision
Support
Model

Model
Prediction

Program
Recommendations



Web Application

- Users can access our tool through a web application
- The UI has a simplistic design that includes:
 - an input prompt for users to submit their interests
 - upon user submission, a display of the top areas of study that most closely matched the user's interests
 - a small about page


Working - User Input

[Home](#)[About](#)

PROGRAM DECISION SUPPORT TOOL

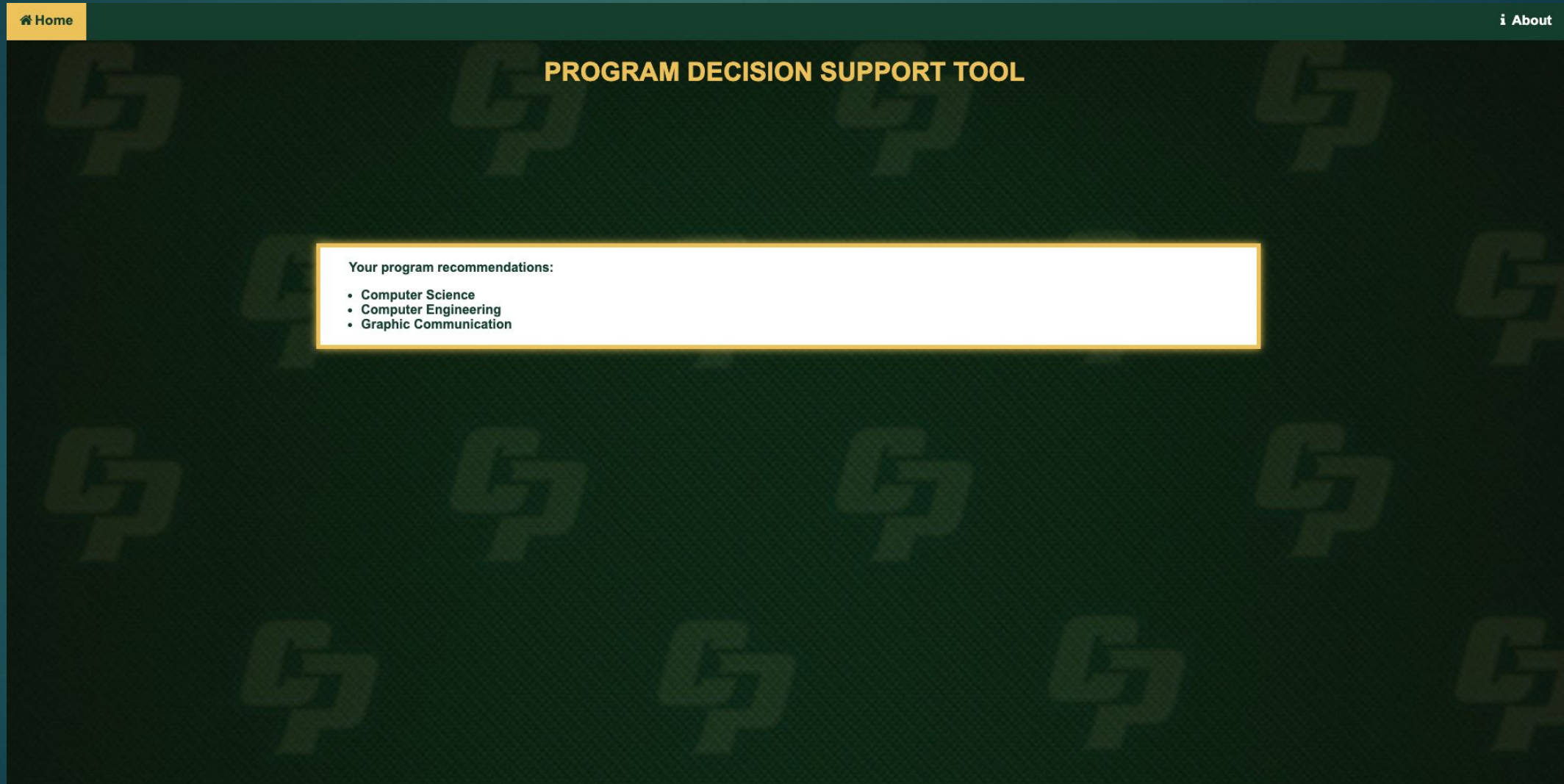
Describe your intrests:

I like to build video games in my free time



Submit

Working - Recommendations



The screenshot shows a web application interface. At the top, there is a dark green header bar with a yellow 'Home' button on the left and an 'About' link on the right. The main content area has a dark green background with a repeating pattern of a stylized 'CP' logo. The title 'PROGRAM DECISION SUPPORT TOOL' is centered at the top of the main area in yellow. Below the title, a white box with a yellow border contains the text 'Your program recommendations:' followed by a bulleted list of three items: 'Computer Science', 'Computer Engineering', and 'Graphic Communication'.

Home

About

PROGRAM DECISION SUPPORT TOOL

Your program recommendations:

- Computer Science
- Computer Engineering
- Graphic Communication

Conclusion

- ▶ Built a dataset from scratch
- ▶ Successfully developed a functional prototype
 - ▶ Included building one model from scratch (Major MLP) and utilizing existing models for further study.
 - ▶ Built out a usable and user friendly front-end tool for students to begin using.
- ▶ The results were within acceptable ranges for all models.
- ▶ Areas we could improve:
 - ▶ Lack of cross-validation
 - ▶ Computation time to compare was large

Future Work

- Building the dataset
 - Collect data from other sources (universities, major-specific reddit, etc)
- Experiment with different
 - Word embeddings
 - Classifier models
- Scale up and apply other parameters
 - Class difficulty
 - DRC accommodations
 - Course builder