# A Tool for Creating an Online Curriculum using Machine Learning

CSE6242 Spring 2023: Team 143 Final Report

Hanlin Chen[a], Aubrey Green[b], Steven Luu[c], Rachel Small[d], Sataporn Worasilpchai[e], Maoz Zisman[f]

## 1 INTRODUCTION

The current method of creating an educational curriculum is done manually by a committee of professional educators and is generally inflexible. In addition, these committees and the courses in the curriculum are siloed based on the course providers. Our goal is to create a recommendation system that will be able to suggest similar courses from all course providers based on a user's input of their goals and develop a course plan to help them meet their goals.

## 2 PROBLEM DEFINITION

With multiple online course providers, it is cumbersome for a user to search across multiple platforms. There are no known tools to link related courses across different platform. Our project aims to solve this problem by creating an interactive tool that facilities the search for courses and/or course paths for users who seek to learn new skills. The application will allow users to filter by course description, skill keywords, job title, and course difficulty to identify a network of courses or course curricula that are tailored to the user's need.

## 3 LITERATURE SURVEY

[1]This chapter outlines the current practice and steps in building a NLP model. This will provide a helpful framework for our research problem.[a]

[2]This article describes recommendation systems used in e-commerce websites (knn, MDP). The shortcoming is relation between objects, which is our unique approach in NLP seeks to improve.[e]

[3]This article supports online self directed learning. This tells us that we are building the right tool. The shortcoming is metrics for success, which we will address in our final product.[e]

[4]This study looked at how career earnings are impacted by changing job markets and advances in necessary skills, showing that STEM career skills are fast-changing.[a]

[5]This text describes various short text similarity methods including string, knowledge, corpus, and hybrid-based. Includes detail on Word2vec and cosine similarity, but does not provide applied examples/results.[b]

[6]This article provides background on designing a curriculum analytics tool to improve an educational program's quality.[f]

[7]This covers best practices and design consideration of electronic surveys. The examples provided are helpful in designing surveys and resolving challenges.[a]

[8]This text discusses how users perceive MOOCs in terms of course content, delivery, system design, quality, and peer-instructor interactivity. This article identifies risks but does not include payoffs.[b]

[9]This article provides an overview of the field of NLP. Upward trends in research grants and publications and the growing importance of NLP machine learning techniques demonstrate the timeliness of our project. Exploration can provide improvements in current techniques.[d]

[10]This article emphasizes the need for a rigorous and well-rounded curriculum. A shortcoming is the accuracy of baseline labels, this will be address with stats.[e]

[11]This chapter introduces applied text cleansing steps such as tokenization, stemming, lemmatization, stop word removal, and spell check using the NLTK library. This will be used in our experiments.[b]

[12]The authors propose a hybrid method of abstractive-extractive text summarization RL which showed improvement over word embedding techniques alone on a text summarization task. A hybrid approach may be useful for our models. As a limitation, results were less relevant when using small datasets.[d]

[13]This chapter presents a different methodology for t-SNE vectorization. This is useful in outlining a methodology and visualizing t-SNE through dimensionality reduction. Limitations due to data overlap will be considered in our t-SNE component.[d]

[14]This article can validate the success of our curriculum optimizer tool by computing the ROI of a student's newly learned skills from taking suggested MOOCs.[f]

[15]This study could be used to extract discriminative

words from course descriptions. Specific keywords are extracted from a text to form a feature vector, given a word-weighting metric, and then run through an SVM classifier to categorize the vectors.[f]
[16]This article covers past research on word embeddings and Word2vec providing context. It is a good reference for implementation/debugging of our model.[c]
[17]This article compares word embedding techniques such as GloVe, Word2vec, and FastText, which is useful to assess the performance of our word embeddings.[c]
[18]This article looks at embeddings in the context of word similarity matchmaking, which is what we are trying to achieve in our project.[c]
[19]This paper contains the logic and mathematics theory behind t-SNE.

## 4 PROPOSED METHOD

### Intuition

The core design of our product is a recommendation system for online courses. Typical state of art recommendation systems are item-based, user-based, or use collaborative filtering (CF). According to TechCrunch, big tech companies such as Amazon, Facebook, Twitter, LinkedIn, Spotify, and Google News are all using CF for their recommendation systems. Our target audience generally are at the very early stages of their job search or study since they will use our tool to help them build a curriculum. The lack of existing user-specific data for our audience will pose a difficult challenge when building an Item-User Matrix needed for CF. However, our approach can provide course recommendations by text mining their desired occupation and course descriptions, without an existing user's profile or history.

Since our target users are at an early stage of their learning journey, they may not be knowledgeable enough to gather the keywords to begin their research. We have gathered courses from five different online learning platforms and provided them with (1) interactive filtering, (2) course visualization, and (3) a curriculum composer. This approach is far more suitable than the modern-day workflow of doing their own research on skills with which they are unacquainted on websites they may never have used.

### Data Preparation & Cleansing

We collected data sets (sourced from Kaggle.com) across various online learning platforms such as Udemy, Udacity, Coursera, EdX, and Skillshare. Because the data sets came from different sources, it was important to remove columns and manipulate data to ensure that all of our data subsets had the same schema before performing a union to concatenate into the final training data. The joined data set was cleaned to standardize columns such as course duration and resolve missing values. We used the NLTK SDK to manipulate the text from the course title and description. The NLP process was performed to get our text corpus ready for Word2vec training, which included word tokenization, stemming, removing stop words, and using regex to only keep alphanumeric characters. As a result, punctuation, foreign languages, and unsupported symbols were removed.

### Algorithms

Our approach was to train a word embedding on the course corpus, then use t-SNE to perform a linear transformation to create an N -> 2 mapping (we mapped the high n-dimension word embedding into a two dimension space) for visualization purposes. The Word2vec vector is the hidden layer of a neural network, where each node is a feature. With so many features, inspection of hundreds of charts to make sense of our Word2vec model is not feasible. A 2-D graph generated by t-SNE will be useful to help our audience understand and visualize the semantic and contextual relationships between the vector representing the courses.

First, we trained a Word2vec embedding on the corpus of the course title and the course description. We focused on the skip-gram variation of the algorithm. In skip-gram, when w is the center word and o is context word, their corresponding vectors are $u_w$ and $v_o$. We obtain the probability of o being in w's context by applying the inner product of the two vectors, then we applied a softmax layer to normalize the probability into [0-1] range, $P(w_{t+j}|w_t) = \frac{exp(u_{w_t} * V_{w_{t+j}})}{\sum_{k \in V} exp(u_{W_t} * V_k)}$. In this model, a word and its context words within the window size are positive training samples, and words outside are negative samples.

Once the Word2vec model was obtained, we applied that model to transform every course and occupation text into vectors. Our goal was to find a list of courses most relevant to a user selected occupation. Relevancy is defined by the distance between the corresponding

vectors; the lower the lower the distance, the more relevant the corresponding courses are to each other. The distance metric we selected was cosine similarity, which calculates the cosine of the angle between two vectors in a multi-dimensional space, $\cos(\theta) = \frac{A \cdot B}{\|A\|_2 \|B\|_2}$. A high cosine similarity between an occupation vector and a course vector could signify the course is highly relevant to the occupation, whereas the opposite would be true if there was a low cosine similarity between the occupation and course vectors. To obtain the list of relevant courses, we stored the course and its cosine similarity score in a max heap, and the output will provide us with a list of relevant courses.

We used a non-linear dimension reduction algorithm named t-SNE to map the high dimension word embedding vectors into a 2-D space. The benefit of t-SNE is its ability to map high dimensional data into lower dimensional space while the distance in their original space is preserved. First, the algorithm calculates the Euclidean distances of each point from all of the other points in the data's original dimension. Second, these distances were transformed into conditional probabilities that represent the similarity between every two points. The conditional probability of center point $x_j$ to another point $x_i$ is represented by a Gaussian centered at $x_i$ with some standard deviation $\sigma$; the probability will be represented as P(X). Third, these points were rebuilt in a low-dimensional space and a joint probability distribution, Q(y), was calculated. Lastly, we used KL divergence to measure how much the P(X) and Q(y) distributions differ from one another. A KL divergence of zero signifies the two distributions are identical. Knowing this, the points were re-positioned with the objective function to minimize KL divergence, which can be done using gradient descent.

Experiments were performed to fine-tune the Word2vec model, see section 5 (Experiments & Evaluation) for more detail.

**User Interface**

The results of the model were stored as CSV files and visualized through Power BI. The schema output for the course generator is (job-title, original-title, original-description, difficulty-level, platform-source, duration-min, duration-hour, rating, rating-group, and cosine-sim). The schema output for the course explorer is similar, but also includes (t-SNE-x-coord, t-SNE-y-coord). The home page of the user interface provides the user with two selection options: 1.) Course Generator and 2.) Course Explorer (see Appendix, section I). The Course Generator displays a curriculum across difficulty levels that are most similar to the job title that the user selects. The Course Explorer displays a 2-D scatterplot of clustered, similar courses that are color-coded by job title. The user can hover-over each point on the scatter plot to get information on the course title, source, difficulty level, rating, and duration as well as utilize the same filters to narrow their search.

In addition to the Power BI user interface, we have included a basic web interface that allows for model inference. A user can either input a query string in the UI or send an API request to our notebook. This will preprocess that string, generate a word embedding, calculate cosine similarities, and return a course plan for the user.

**Deployment**

Typically, a Power BI report would be released to the Microsoft Power BI Service to allow users to view and interact with the dashboard. However, limitations in accessing the service through a university-provided email account prevent us from doing so. Instead, the .pbix file will be provided to the user for interaction. In addition to the Power BI file, we will temporarily deploy a production model where the user can dynamically query courses by typing in a job description. This model will be hosted by Gradio and runs directly off our Google Colab notebook.

## 5 EXPERIMENT & EVALUATION

The tool was developed via the following steps: 1) Cleaned and prepared text from Udacity, Udemy, SkillShare, Coursera, and EdX; 2) Trained Word2vec word embedding model on course titles and descriptions to generate word vectors; 3) Computed the sentence vector through the average of word vectors; 4) Created visualization on a 2-D space by performing feature transformation which provides a more user-friendly experience; and 5) Created a course plan containing the most similar course from each difficulty level, including alternatives.

The result for our first model was promising. We obtained a list of related courses from a unique query string description of a course that a user wants to take. The hyperparameters of our models were tuned to increase the consistency of the related courses from the model. Using multiple sets of ground truth, which are

pairs of synonyms (determined by us), the hyperparameters were evaluated by determining which values give the maximum summation of cosine similarity across all class pairs.

The main method of visualizing the output of the Word2vec model is through the t-SNE graph. The length of each course vector is 300, which corresponds to the number of nodes in the hidden layer. The t-SNE graph converts the high dimensional space of the vector into a 2-D space for easy visualization. The relative distance between each represented course is determined by the perplexity variable.

We evaluated our Word2vec embedding experiment by checking for its ability to complete two different NLP tasks, with the former being an intrinsic evaluation and the latter being extrinsic. The goal was (1) to experiment with our model to find the configuration which outputs the best metrics (2) to evaluate our model to prove that the model is useful for our NLP tasks.

**Hyperparameters for Word2Vec**

- Epoch: Epoch is the number of iterations of all the training data when training the Word2vec model.
- Window size: When training the Word2vec model, the window size determines how many words before and after the target word are included in the word context sampling procedure.
- Vector size: The vector size refers to the number of nodes in the single hidden layer neural net.
- Min count: The model will ignore all words with a total frequency smaller than this integer parameter.
- Negative: Negative sampling determines how many "noise words" will be drawn.

**Hyperparameters for T-SNE**

- The hyperparameter we focused on for t-SNE was perplexity. Perplexity is the standard deviation $\sigma$ from Gaussian distribution in the high-dimensional space. Typically, the recommended range is 5-50.

**Experiment 1**

We also evaluated our embedding on its ability to complete a real task (i.e., classification). In our Udemy dataset, we were fortunate to have a category label such as IT Software, Business, Lifestyle, Finance Accounting, Design, etc. Our hypothesis was that if our embedding was useful, we would be able to use Word2vec representation of the Udemy course title and description along with a simple/weak-learner algorithm to correctly classify their corresponding course categories. We ran the simulation with multiple iterations of this classification task as a grid search by isolating the decision tree classifier. Only the hyperparameter of the Word2vec model was tuned to find the configuration which could give us the highest model accuracy.

| GMM+ Transform Feat | | | | |
|---|---|---|---|---|
| Rank | Accuracy | VectorSize | window | MinCount |
| 1 | 57.87% | 300 | 6 | 8 |
| 2 | 57.63% | 300 | 4 | 10 |
| 3 | 57.52% | 300 | 8 | 6 |
| 4 | 57.48% | 300 | 8 | 8 |
| 5 | 57.46% | 300 | 10 | 4 |

Since there are 13 unique categories in our Udemy dataset, we proposed a naive baseline dummy model which guesses category randomly, which means this model would have about 1 out of 13 chance or 7.69% chance of accurately identifying the correct course category. When applying our Word2vec alongside an untuned decision tree, a 57.87% accuracy score was observed, which is a 652.54% lift increase over the naive dummy model. Since we did no tuning to the decision tree, and the fact we achieved a massive lift in our KPI over the naive model further validates the usefulness of our Word2vec model.

**Experiment 2**

The second approach was based on the hypothesis that if our word embedding would output a high correlation between sets of synonyms, then the model is performing as expected. We tuned the hyperparameters of the Word2Vec model by finding the configuration which provided us with the highest summation of cosine similarity across each pair of synonyms and then divided it by the number of pairs. Among the set of synonyms, our model was able to assign a high degree of correlation, for example, an 86% correlation was observed between [recipe - desserts] and 73% correlation was seen between [stock - crypto], see Appendix, section II.

In addition to experiments 1 and 2, we attempted to improve the performance of the Word2vec model by isolating the hyperparameters: window size, min/word-count, negative sampling, and epochs. When isolating each parameter, other hyperparameters are set to the

default value from the Gensim library. The default parameters are as follows: window size = 5, min/word count = 5, negative sampling = 5, and epochs = 5. The performance metrics we used were defined by the ratio of the average ground truth of matched cosine sim, to the average of the ground truth of the mismatched cosine similarity.

$$performance = \frac{\sum cos(\theta) knownMatched}{\sum cos(\theta) knownMismatched}$$

The table below summarizes the results of the supplemental experiment. The best value that produces the highest ratio for each respective hyperparameter is shown below.

| Hyperparameter Experiment | | | | |
|---|---|---|---|---|
| Param | COSIM+ | COSIM- | Ratio | Param Value |
| Window | 0.779 | 0.042 | 18.547 | 11 |
| Count | 0.761 | 0.038 | 20.026 | 20 |
| Negative | 0.750 | 0.035 | 21.428 | 16 |
| Epoch | 0.770 | 0.040 | 19.250 | 7 |

The table above shows the maximum ratio value which was utilized as our performance evaluator. It can be seen in the line graph (see Appendix, section III. Figure. 1) that the window size's impact on the ratio converges at around window = 5 and has a downward trend after window = 11. Since the window size is the number of words away from the target word during the embedding process then, when the widow is small, it overfits the meaning of the target word to its nearest neighbor. When the window size is large (n>11), the embedding loses its ability to carry the meaning of the word. We believe that window size could be useful if we had a larger set, but for our dataset, the default n=5 is adequate for our application. The min word count and the negative parameters level out after n = 10, in that there is no significant gain in performance when we remove words that have a low word count or increase noise (negative sampling).

One key parameter in training a deep learning model is the number of iterations of the training data (epoch). In our experiment, the number of epochs that gave us the highest performance was n = 7. When n > 7, the performance of the model did not improve (see Appendix, section III. Figure 2). Since Word2vec is a single-layer neural net, more epochs could easily overfit the training data, which is the case when n > 7. One improvement would be to increase the number of ground truth pairs that were used to calculate the cosine similarity score. For example, the hyperparameter experiment uses 7 courses that were handpicked that we believe are most related to "Java object-oriented programming for beginners". To be more rigorous in our experiments, we need additional pairs that query other subjects and expand our ground truth pairs.

**Evaluation 1**

Curricula were determined by finding a multiple of courses in each level (beginner, intermediate, advanced) with the highest cosine-similarity. Some courses across the platforms are similar, i.e., Machine Learning with Python on edX and Scikit-learn in Python for Machine Learning on Udemy. To address this and ensure variety in the curriculum, the cosine-similarity scores were again calculated using the subset of courses based on job title search and course level. Within each subset, the courses with the lowest similarity were used to populate the curricula. This method proved useful for topics that were well-represented in the dataset, but possibly detrimental for topics that were less represented. This approach could be further refined in future to handle these cases differently.

**Evaluation 2**

The scatter plot in the Course Explorer of Second t-SNE (on the y-axis) vs First t-SNE (on the x-axis) shows a clustering of courses, color encoded by job title search (see Appendix, section IV). In general, jobs in the Arts sector cluster on the left, Finance and Business cluster in the center, and Science and Tech-related jobs cluster on the right.

Focusing in on a view of the courses for the "Data Science" job title selection (see Appendix, section V) shows a clustering of course subjects by skill set with courses related to Python Programming on the left and those related to Data Processing on the right. Accordingly, the "Data Science" cluster area overlaps with the clusters for "Python Developer" on the left and "Data Product Engineer" on the right. For job title searches in the technology sector, it appears the model well-classifies similar courses.

**Limitation**

Not all job title searches yielded relevant results. For example, the search results for "Fishing" include course subjects in Antiretroviral Therapy and Optical Physics (see Appendix, section VI). Searching "Pastry Chef/Cook"

resulted in a curriculum that includes Business Management and Icelandic Sagas. The least relevant results appear in the intermediate- and advanced-level categories. This limitation is likely due to the data set having fewer courses related to these jobs, especially those of intermediate and advanced levels. This exercise is an illustration of the importance of robust data. As more courses are created in the MOOC platforms and this data becomes available, the tool can continue to improve.

## 6 CONCLUSIONS & DISCUSSION

Many insights were formed in the pursuit to build a comprehensive MOOC curriculum tool that enables various stakeholders to choose the curriculum that best suits their needs. Using NLP algorithms like Word2Vec for determining similar courses and forming curricula to recommend as well as t-SNE to allow visualization of similar clusters of courses, enabled us to create a valuable novel tool that will make choosing online courses more manageable. Power BI was the platform chosen to visualize the model results and create an interactive user interface.

Model performance was analyzed by running experiments on, 1) word embedding correlation between sets of synonyms, 2) course classification by category, 3) tuning Word2vec hyperparameters. When training the Word2vec model, limited model performance improvements were realized from tuning the default hyperparameters. However, based on the chosen hyperparameters, the model produced several distinct job clusters of courses as presented in the t-SNE course explorer representation.

For further enhancement of the tool some future improvements could be explored. With a larger dataset of online courses we could study the effects of increasing the Word2vec window size when tuning the model's hyperparameters to ultimately achieve higher model accuracy. Additional improvements that could be explored include finding a complete MOOC dataset that contains difficulty classification rankings for each course. As an alternative, a survey could be provided to the users of our tool in which it asks what the difficulty of a given course was. The survey data can then be fed back into the model to improve future course queries. The lack of courses with an intermediate or advanced ranking classification was most apparent when the model attempted to select an appropriate course path for specific jobs such as "Fishing" or "Pastry Chef/Cook." While there are limitations in its current form, refining the course curriculum tool will ensure that learners receive the best possible curriculum to provide them the greatest return on investment in terms of time and money.

## 7 STATEMENT OF EFFORT

Please see the Gantt chart in the Appendix, section VII, for detailed activities for each group member and approximate completion date. All team members have contributed a similar amount of effort towards the project.

## References


[1] S. Vajjala, B. Majumder, A. Gupta, and H. Surana. *Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems*, chapter 2. NLP Pipeline. O'Reilly Media, 1st. edition, 2020.

[2] Malte Ludewig and Dietmar Jannach. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, 28:331–390, October 2018.

[3] Chih-Hsuan Wang, David M. Shannon, and Margaret E. Ross. Students' characteristics, self-regulated learning, technology self-efficacy, and course outcomes in online learning. *Distance Education*, 34(3):302–323, October 2013.

[4] David J. Deming and Kadeem Noray. Earnings dynamics, changing job skills, and stem careers. *The Quarterly Journal of Economics*, 135(4):1965–2005, June 2020.

[5] Dimas Wibisono Prakoso, Asad Abdi, and Chintan Amrit. Short text similarity measurement methods: a review. *Soft Computing*, 25(3):4699–4723, January 2021.

[6] Isabel Hilliger, Camila Aguirre, Constanza Miranda, Sergio Celis, and Mar Pérez-Sanagustín. Lessons learned from designing a curriculum analytics tool for improving student learning and program quality. *Journal of Computing in Higher Education*, 34:633–657, March 2022.

[7] Martine Van Selm and Nicholas W. Jankowski. Conducting online surveys. *Quality and Quantity*, 40(3):435–456, 2006.

[8] Yash Daultani, Mohit Goswami, Ajay Kumar, and Saurabh Pratap. Perceived outcomes of e-learning: identifying key attributes affecting user satisfaction in higher education institutes. *Measuring Business Excellence*, 25(2):216–229, February 2021.

[9] Xieling Chen, Haoran Xie, and Xiaohui Tao. Vision, status, and research topics of natural language processing. *Natural Language Processing Journal*, 1:100001, 2022.

[10] Christian T. Doabler, Ben Clarke, Hank Fien, Scott K. Baker, Derek B. Costy, and Mari Strand Cary. The science behind curriculum development and evaluation: Taking a design science approach in the production of a tier 2 mathematics curriculum. *Learning Disability Quarterly*, 38(2):97–111, February 2014.

[11] Nitin Hardeniya, Jacob Perkins, Deepti Chopra, Nisheeth Joshi, and Iti Mathur. *Natural Language Processing: Python and NLTK*, chapter Module 1 Chapter 2. Packt Publishing, Birmingham, UK, 1st. edition, 2016.

[12] Qicai Wang, Peiyu Liu, Zhenfang Zhu, Hongxia Yin, Qiuyue Zhang, and Lindong Zhang. A text abstraction summary model based on bert word embedding and reinforcement learning. *Applied Sciences*, 9(21):4701, November 2019.

[13] Chelimilla Natraj Naveen and Kamal Kumar. Text visualization using t-distributed stochastic neighborhood embedding (t-sne). In *Conference Proceedings of ICDLAIR2019*, volume 175 of *Lecture Notes in Networks and Systems*, pages 46–52. Springer, Cham, February 2019.

[14] Fabian Stephany, Ole Teutloff, and Vili Lehdonvirta. What is the price of a skill? revealing the complementary value of skills. *MPRA Paper*, (114874), September 2022.

[15] Marius Sajgalik, Michal Barla, and Maria Bielikova. Searching for discriminative words in multidimensional continuous feature space. *Computer Speech Language*, 53:276–301, September 2019.

[16] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013.

[17] Billy Chiu and Simon Baker. Word embeddings for biomedical natural language processing: A survey. *Language and Linguistics Compass*, 14(12):e12402, 2020.

[18] Fengqi Yan, Qiaoqing Fan, and Mingming Lu. Improving semantic similarity retrieval with word embeddings. *Concurrency and Computation: Practice and Experience*, 30(23):e4489, 2018.

[19] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9, 11 2018.

# A Appendix

I: Top most contextual similar word according to our Word2Vec model

a. Top most contextual similar word to "recipe"

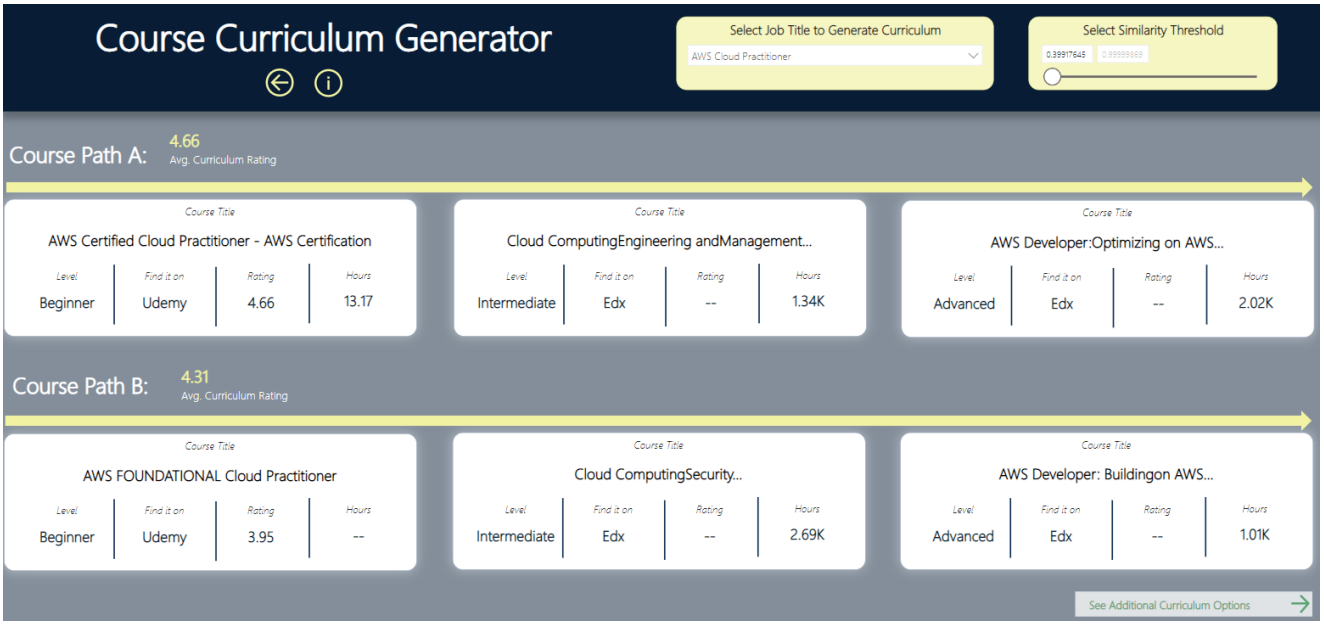
```
[('delici', 0.9006413817405701),
 ('dessert', 0.863211989402771),
 ('bread', 0.8573224544525146),
 ('dish', 0.8373355269432068),
 ('sourdough', 0.8260807991027832),
 ('bake', 0.8259247541427612),
 ('homemad', 0.8239259123802185),
 ('vegan', 0.8216069936752319),
 ('cook', 0.8160303235054016),
 ('cooki', 0.799690306186676)]
```

b. Top most contextual similar word to "stock"

```
[('vix', 0.7477042078971863),
 ('crypto', 0.7344565987586975),
 ('indic', 0.7286353707313538),
 ('etf', 0.7246533036231995),
 ('commod', 0.7222992777824402),
 ('forex', 0.7175624370574951),
 ('candlestick', 0.7065955400466919),
 ('fibonacci', 0.6998773813247681),
 ('intraday', 0.6961836814880371),
 ('altcoin', 0.6911096572875977)]
```

II. User Interface Views

a. Curriculum Generator



b. Course Explorer


## III. Hyperparameter Experiments

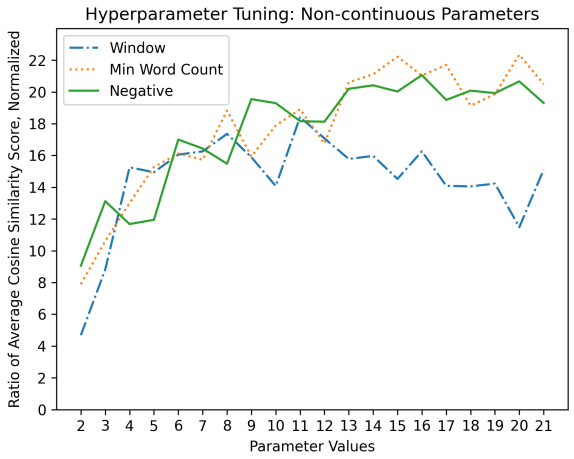



**Figure 1: Embedding Parameters**

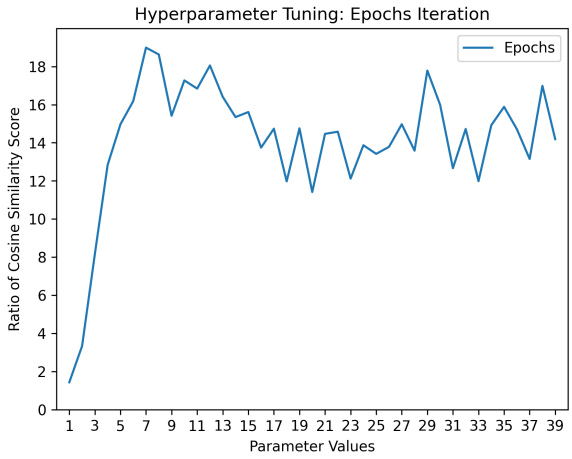
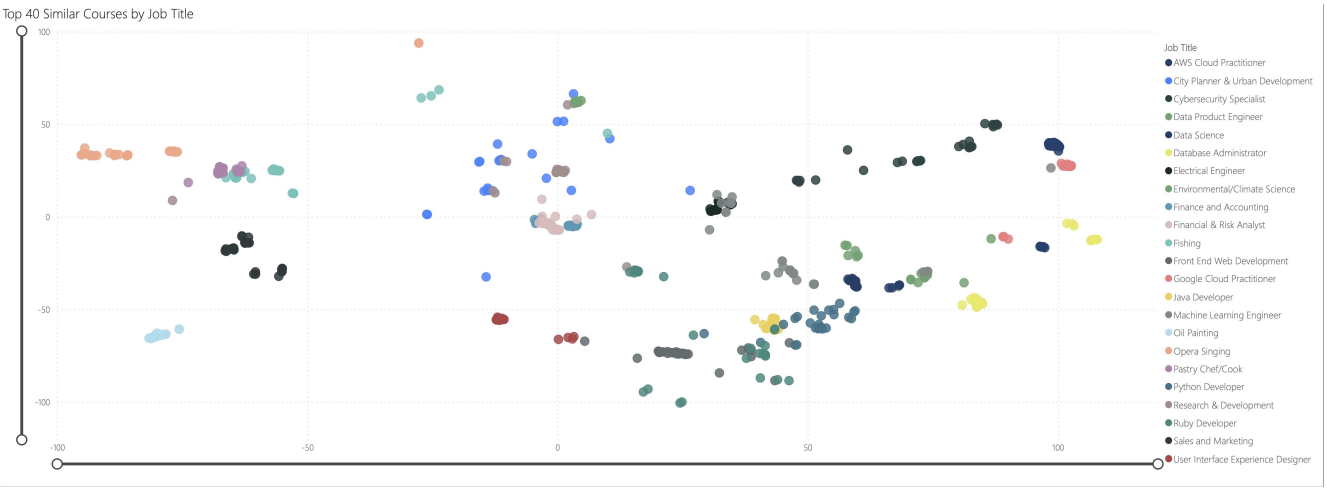


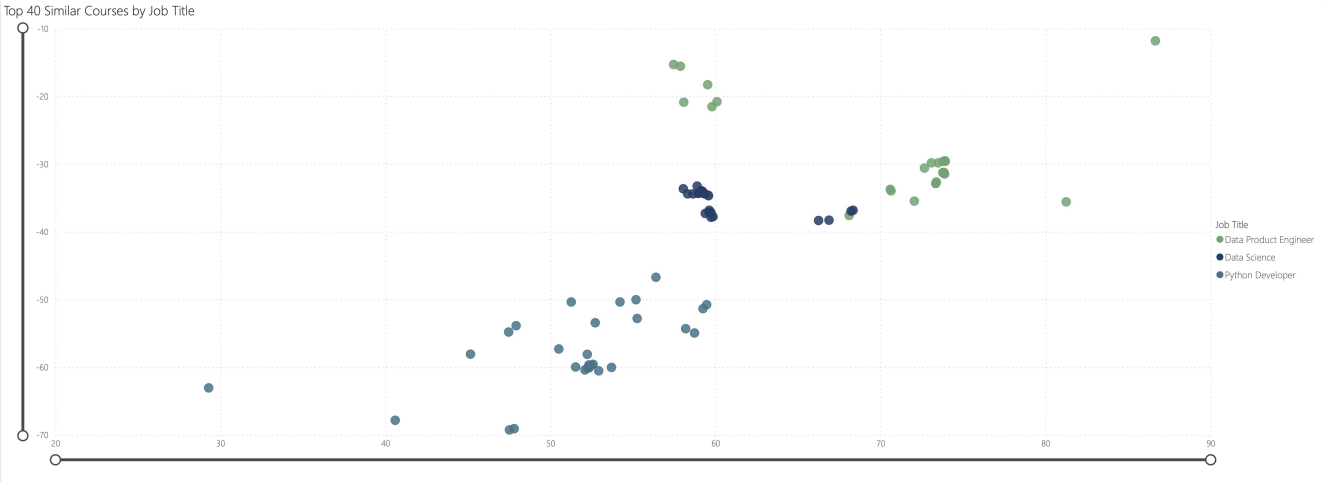

**Figure 2: Epoch Experiment**

IV: Cluster t-SNE Scatterplot of Top Courses

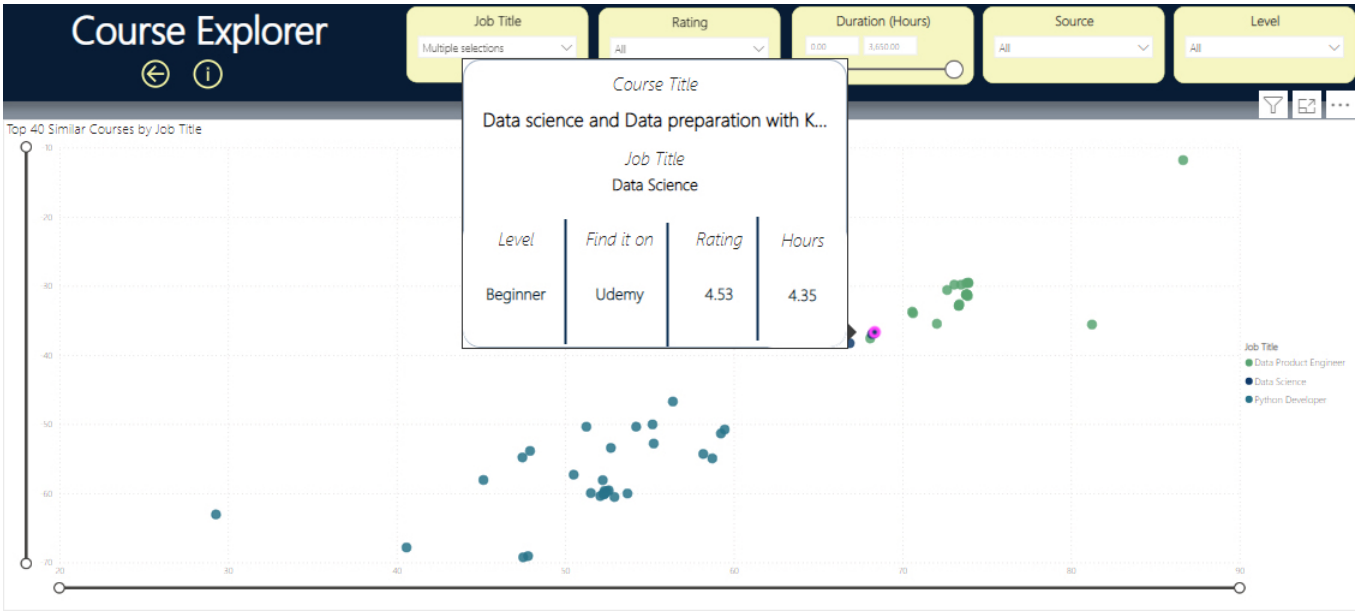



V: Data Science case

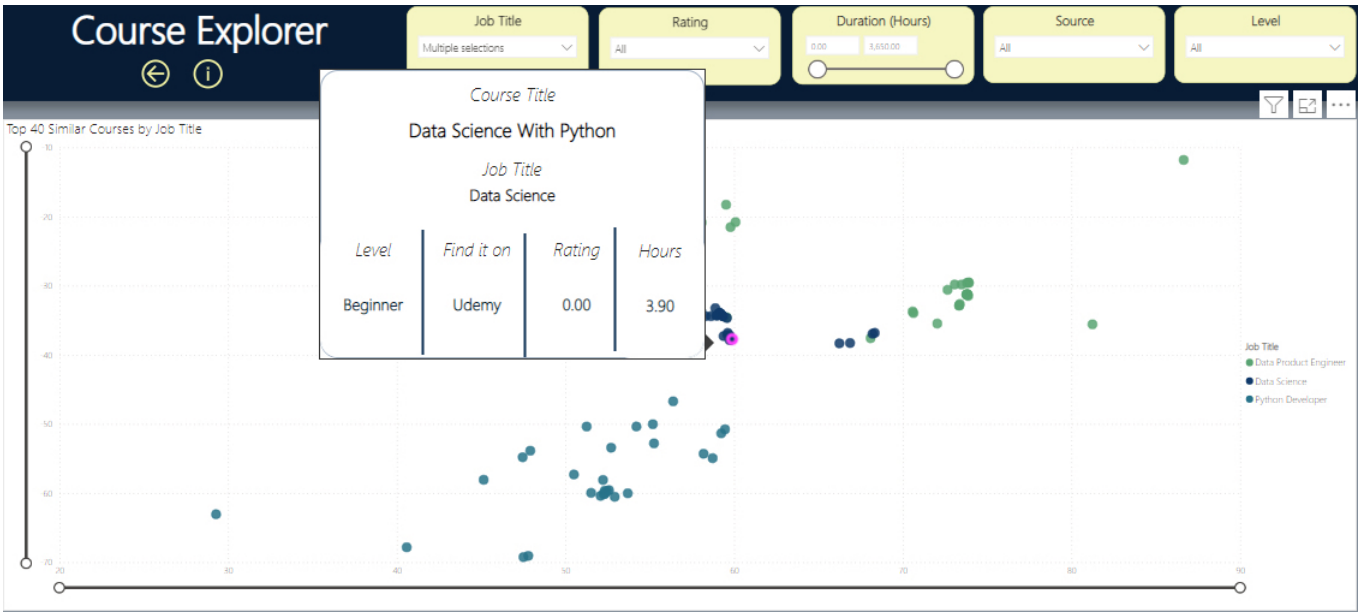
a. Zoom view of course clusters for Data Product Engineer, Data Science, and Python Developer job title searches



b. "Data Science" course most similar to "Data Product Engineer")

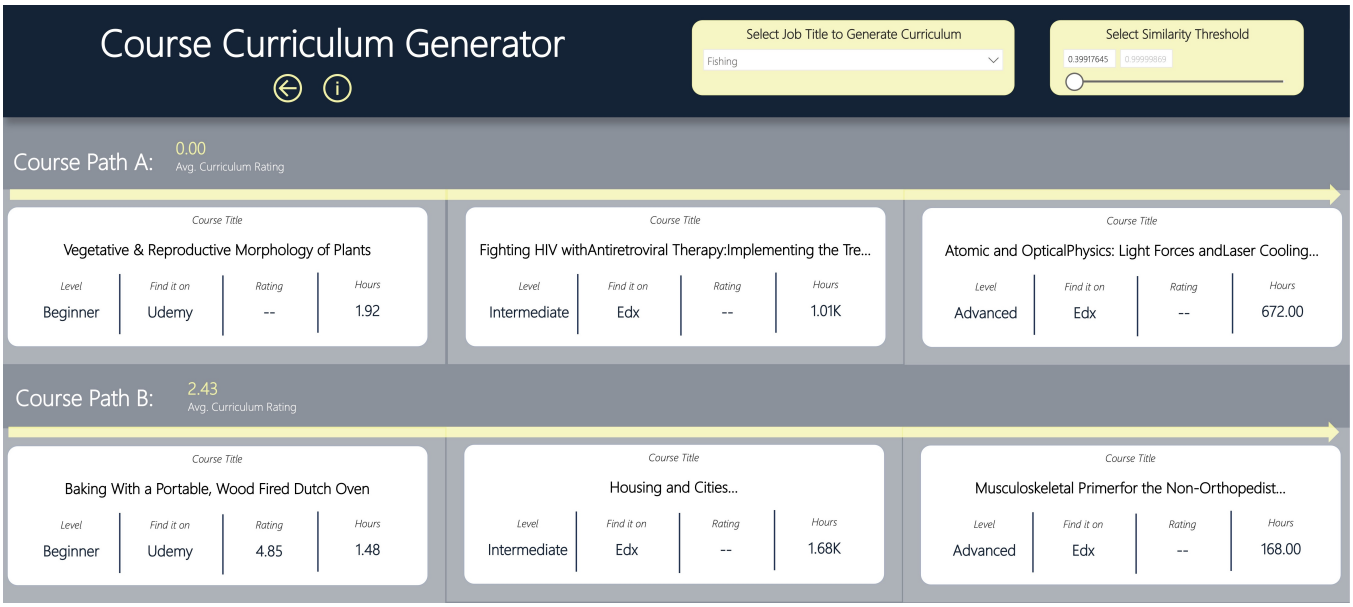



c. "Data Science" course most similar to "Python Developer"

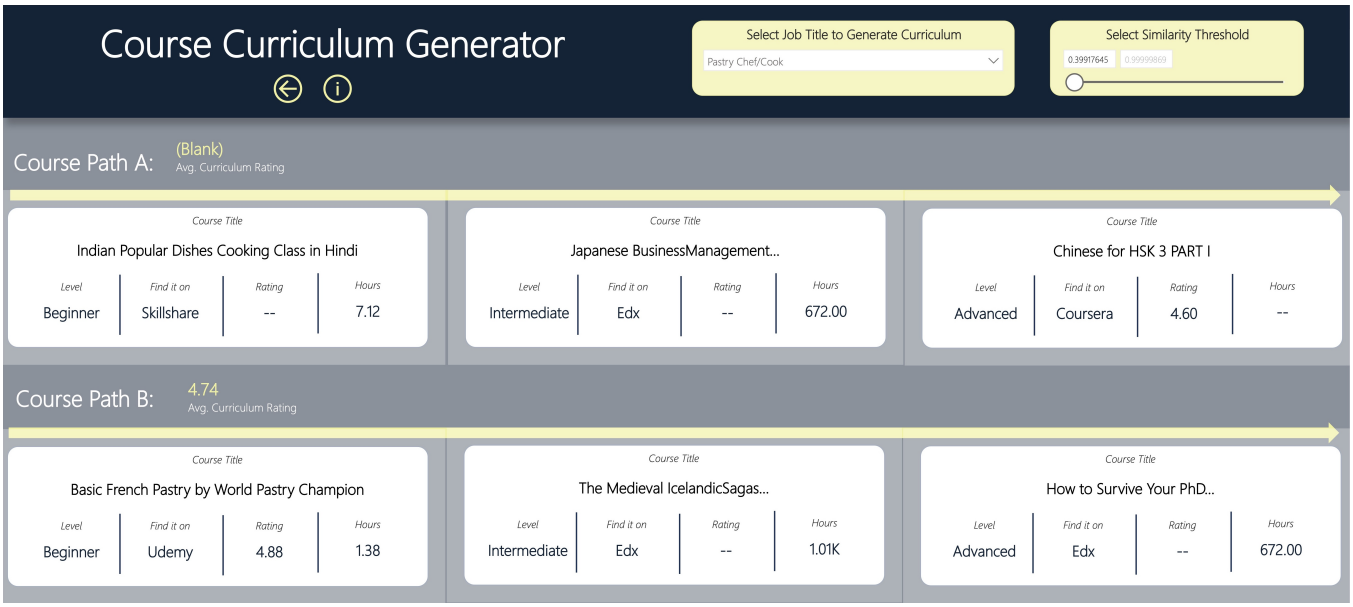


VI: Examples of irrelevant search results

a. "Fishing"




b. "Pastry Chef/Cook"



VII. Gantt chart for project efforts.

| Assignment | Requirement | 13-Feb | 20-Feb | 27-Feb | 6-Mar | 13-Mar | 20-Mar | 27-Mar | 3-Apr | 10-Apr | 17-Apr | 24-Apr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Proposal Document Due March 3rd** | Plan of Activities<br>Expected Innovations<br>Literature Survey<br>Heilmeier's Questions | Team | | | | | | | | | | |
| **Proposal Presentation Due March 3rd** | Proposal Video & Slides<br>Create Unlisted Youtube Video | | | Team | | | | | | | | |
| **Prototyping** | Aggreate Data<br>Train Word2vec model<br>Visualized TNSE Graph | | | Aubrey Green | Halin Chen | | | | | | | |
| **Progress Report Due March 31st** | Proposed Method<br>Experiments and Evaluation<br>Plan of Activities | | | | | Team | | | | | | |
| **Build Final Product** | Evaluate Model Output<br>Optimized word2vec model<br>Port Result to Power BI<br>Build Power BI Interface<br>Relates Job Data to Course Data<br>Implement Filters Mechanism<br>Embed powerBi Iframe to Web application<br>Publish Web application<br>Test Web application | | | | | | | Hanlin Chen<br>Sataporn W.<br>Maoz Zisman<br>Aubrey Green<br>Rachel Small & Hanlin Chen<br>Steven Luu & Rachel Small<br>Sataporn W.<br>Steven Luu<br>Maoz Zisman | | | | |
| **Final Report Due April 21st** | Intro and Motivation<br>Problem Definition<br>Literature Survey<br>Proposed Method<br>Experiments and Evaluation<br>Conclusion and Discussion<br>Submit Zip File | | | | | | | | Hanlin Chen<br>Aubrey Green | Rachel Small<br>Steven Luu<br>Sataporn W.<br>Maoz Zisman | Sataporn | |
| **Final Poster Presentation Due April 21st** | Individual Videos<br>Create Unlisted Youtube Video<br>Post Submission | | | | | | | | | Team | Sataporn | |
| **Peer Assessment Due April 28th** | Review Classmates Submissions | | | | | | | | | | | Indivudial |