Analyzing Student's Problem-solving Approaches in MOOCs using Natural Language Processing

by

ByeongJo Kong

B.E., Kyung Hee University (2010)

Submitted to the System Design and Management Program and the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of
Master of Science in Engineering and Management
and
Master of Science in Electrical Engineering and Computer Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
May 2022

Author . . . . . .
System Design and Management Program and
Department of Electrical Engineering and Computer Science
May 6, 2022

Certified by . . . . . .
Una-May O'Reilley
Principal Research Scientist, MIT Computer Science & Artificial Intelligence Lab
Thesis Supervisor

Certified by . . . . . .
Erik Hemberg
Research Scientist, MIT Computer Science & Artificial Intelligence Lab
Thesis Supervisor

Accepted by . . . . . .
Joan S. Rubin
Executive Director,
System Design and Management

Accepted by . . . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Analyzing Student's Problem-solving Approaches in MOOCs using Natural Language Processing

by

ByeongJo Kong

Submitted to the System Design and Management Program and the Department of
Electrical Engineering and Computer Science
on May 6, 2022, in partial fulfillment of the
requirements for the degrees of
Master of Science in Engineering and Management
and
Master of Science in Electrical Engineering and Computer Science

Abstract

Problem-solving processes are an essential part of learning. Knowing how students approach solving problems can help instructors improve their instructional designs and effectively guide the learning process of students. This thesis proposes a natural language processing (NLP) driven method to capture online learners' problem-solving approaches while using Massive Open Online Courses (MOOCs) as a learning platform. It employs an online survey to gather data, NLP techniques, and existing educational theories to investigate this in the lens of both computer science and education.

The thesis considers survey responses from students enrolled in a computer programming course taught on edX in Spring 2021. A total of 7,482 free-text responses are selected from 44,864 responses collected through the survey. The thesis shows how NLP techniques, i.e. preprocessing, topic modeling, and text summarization, must be tuned to extract information from a large-scale text corpus. The proposed method discovered 18 problem-solving approaches from the text data, such as using pen and paper, peer learning, trial and error, etc. By using datasets from 2020 and 2021, we also learned that there are strong topics that appear over the years, such as clarifying code logic, watching videos, etc. Lastly, we used existing educational theories to discuss the findings from a viewpoint of education.

Thesis Supervisor: Una-May O'Reilley
Title: Principal Research Scientist of MIT Computer Science & Artificial Intelligence Lab

Thesis Supervisor: Erik Hemberg
Title: Research Scientist of MIT Computer Science & Artificial Intelligence Lab

Acknowledgments

Throughout the course of this journey, I have received a great deal of support and guidance. I would not have achieved this much without the kind support and love of many others.

I would first like to express my sincere gratitude to Erik Hemberg and Una-May O'Reilly for giving me the opportunity to work with them on this research. They provided incredible support and guidance to further improve the research. I am truly grateful for their constant help and advice, which taught me a great deal and enabled me to advance in research. I would also like to thank Ana Bell for providing help which substantially helped our research. I must express my gratitude to my parents and sister who provided continuous support and encouragements at every moment. Their unwavering love helped me stay strong and positive through the tough times. I also owe my sincere thanks to Myoung Seung Park, my aunt, and dear friends Sung-Hwan Choi, Keonyoung Ahn for encouragement, faith and love.

I humbly extend my gratitude to all concerned persons who showed support and love along my journey in life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Massive Open Online Courses (MOOCs) are a prevalent mode of educational delivery for higher education. Particularly with the outbreak of Covid-19 in early 2020, universities and many students around the world are now conducting their educational activities on online platforms like MOOCs. In 2021, approx. 19.4K MOOCs were announced or launched by around 950 universities worldwide. In 2021 alone, around 3.1K courses were added [44].

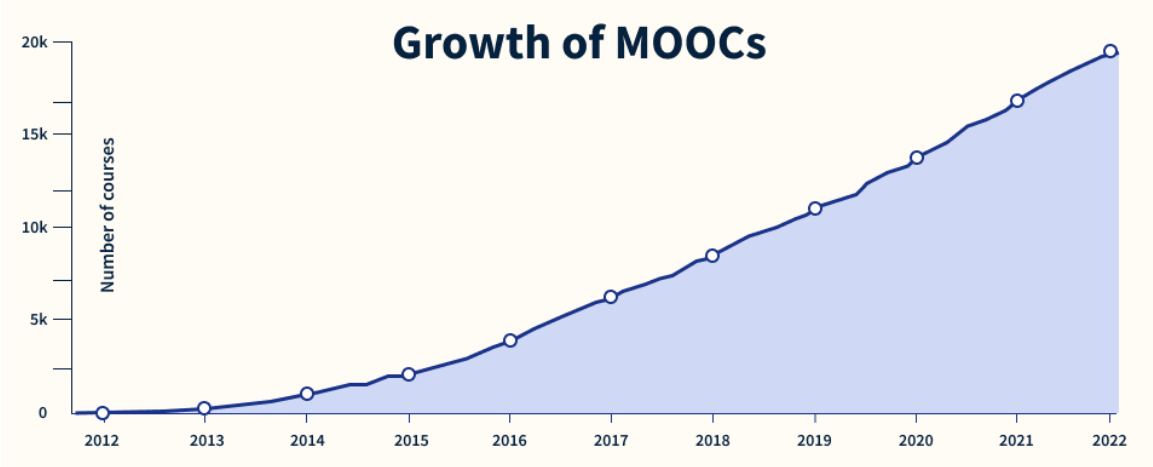


Figure 1-1: Growth of MOOCs [44][1]

Online learning, however, requires different learning engagement and behaviors of students compared to traditional classroom learning. Classroom learning is usually teacher-centered, or passive learning, where the instructor controls classroom dynamics. Often

[1]Statistics do not include China. Source: class central

it is considered restrictive, inflexible, and impractical [40], whereas, in online learning, students independently analyze the information and adopt a diverse set of skills and self-directed learning strategies to successfully assimilate learning content [43]. In such setting, students often proactively seek learning resources that are outside the course boundary to facilitate their learning processes.

There are many previous studies analyzing the types of activities students carry out during the online learning, which mainly focused on the activities recorded by the learning platform, e.g., watching video lectures, submitting assignments, and discussing in forums [27, 24, 8]. They often conduct students' behavioral pattern analysis using the event log and clickstream data. However, students' activities that take place outside the learning platform have received relatively little attention despite their potential influence on the learner retention [13]. A qualitative study conducted based on a small-group (13 individuals) survey, as shown in Fig 1-2, suggests that students often carry out learning activities outside the MOOC platform, such as social media interactions, notetaking, and in-person discussion with friends [49]. The study also claims that the understanding of the full spectrum of learners' activities in open online courses is limited by the extensive dependence on the log files and clickstream data analysis.

This thesis seeks to shed a new light on the less explored part of online learning activities by proposing a framework that captures the holistic spectrum of student's learning activities carried out both inside and outside the MOOC platform. It particularly delves into identifying the problem-solving approaches as they encompass various insights in students' approaches in the discovery of learning and bridging the knowledge gaps [6]. By *Problem-solving approaches*, here, it means the types of strategies and methods students use to solve course problems in MOOCs. For instances, student may ask friends for help, re-watch video lectures, look discussion forums, use IDE, etc. The suggested framework is designed with Natural Language Processing (NLP) methods to process and extract insights from the large amount of text data generated by students. For the data collection, we used an online survey method to ask students to describe their problem-solving approaches in text. Then to clean the data and extract relevant and readable information, we processed the collected free-text responses with NLP techniques, such as Preprocessing, Topic Modeling

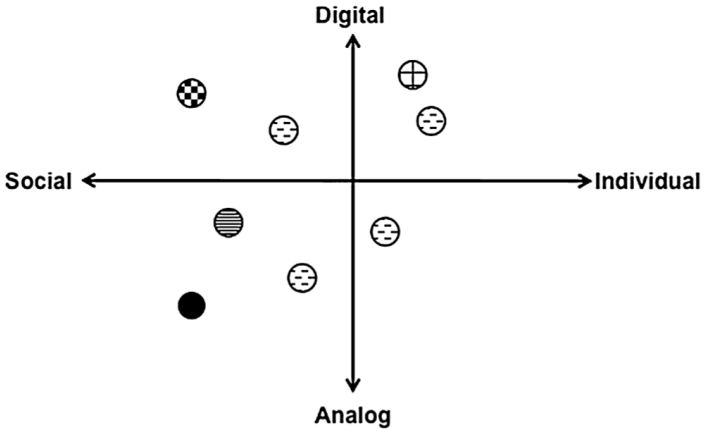


Figure 1: Massive open online course (MOOC) learners' experiences and practices on a digital–analog continuum and a social–individual continuum.

⊕, individually watching a MOOC lecture; 😶, notetaking; ●, social connections about the MOOC with family and friends; ◍, face-to-face study groups with other MOOC participants; ⚽, social connections with other MOOC participants.

Figure 1-2: George Veletsianos et al. discovered learner interactions in social networks outside of MOOC platform by surveying 13 individuals [13]

and Text Summarization.

The preprocessing is an essential part of this research as we are dealing with user-generated data with noise, such as grammatical and spelling errors, missing values, etc. The topic modeling technique is used to discover topics (in our case problem-solving approaches) from large-scale text data, while we used the text summarization technique to generate readable narrative summaries of discovered topics to make them contextually rich and easy to understand. Furthermore, educational theories, such as task-structure and active/passive learning frameworks, are used to translate the findings into educational implications.

1.1 Research Questions

The research questions focus on developing a scalable method to investigate problem-solving approaches of students in MOOCs by utilizing NLP techniques. The thesis also explores ways to derive educational implications from the findings using existing educa-

Table 1.1: Key terms and descriptions

Terms	Description
Massive Open Online Course (MOOC)	An online course aimed at unlimited participation and open access across the world without prerequisites
Problem-solving approach	In this research, it refers to a strategy or a method used by a student to solve course problems in the MOOC
Natural Language Processing (NLP)	A subfield of AI concerned with giving computers the ability to understand, process, and analyze massive text data
Preprocessing	A method to clean and remove noise from text data and make it ready to be fed into the model
Topic modeling	A method to extract topics from text corpus by automatically clustering word groups that best represent the topic
Text summarization	A method to shorten long sentences or paragraphs without omitting critical information
Task-structure framework	A framework that describes possible ways a task can be decomposed into subtasks that need to be solved to complete the entire task.
Active/Passive learning framework	Types of learning methods categorized based on the students' level of participation in the learning process

tional theories. The following list includes the questions that the thesis aims to answer:

Research question 1. (Overarching question) Can we capture the students' problem-solving approaches carried out both inside and outside the MOOC platform? *See Section 6.1*.

Research question 2. (Technical question) Do NLP-driven methods identify accurate problem-solving information? *See Section 5.2 through 5.4*

2.1 Can they maintain high accuracy with extremely noisy, large-scale text data?

2.2 Can topic modeling accurately discover the problem-solving methods?

Research question 3. (Educational question) What are the educational implications of identified problem-solving approaches? *See Section 5.5*

3.1 Using the task-structure framework.

3.2 Using the active/passive learning framework.

1.2 Contributions

The thesis makes the following contributions:

1. Introduces an NLP pipeline that incorporates online survey, topic modeling, and text summarization for discovering student's problem-solving methods carried out both inside and outside the MOOC platform

2. Combines various preprocessing techniques that contributed to reducing the noise of the text data, which noticeably improved the performance of topic modeling.

3. Demonstrates that transformer-based (BERT) topic modeling achieves significantly improved accuracies in identifying problem-solving methods from large text data

4. Showcases how text summarization technique can contribute to enhancing the limited readability of topic modeling results, which are based on a list of keywords

5. Uses task-structure and active/passive learning frameworks to translate the findings into educational implications, which can be useful for instructors to improve their teachning strategies

The overall structure of the paper is organized as follows:

- **Chapter 1 Introduction:** Brief overview of research gap & goals, and contributions.
- **Chapter 2 Related Work:** Literature review of Background information about the concepts and techniques used in the research.
- **Chapter 3 Methods:** Detailed explanation about the research design and methods.
- **Chapter 4 NLP pipeline Sensitivity Analysis and Tuning:** Description of the experimental setup
- **Chapter 5 Experiments:** Results and findings of the research, discussions and limitations.
- **Chapter 6 Conclusion and Future Work:** Summary and future research directions.
- **Appendix A:** List of Generated Bigrams.
- **Appendix B:** Survey Analysis: Multiple-choice Questions.

Chapter 2

Related Work

This chapter discusses previously published articles and papers of related research to show how our work fills in gaps in previous work. We particularly looked into studies that are related to problem-solving and learning behavior analysis in the online learning setting, NLP applications in MOOCs, and education theories that can help us guide the interpretation of problem-solving approaches in the educational perspectives.

2.1 Problem-solving and learning behavior analysis

Observing learners' behavioral information is considered an important prerequisite to analyze the student's learning experience, performance, motivation, and so on [48]. Many earlier studies we reviewed focused on analyzing the learner's behaviors based on the data collected from some MOOC platforms. Some expanded from investigating the students' behavioral data to building predictive models of the student performance [42, 29, 26, 36]. Chamizo-Gonzalez et al. (2015) [25] and Al-Musharraf et al. (2016) [1] studied on the correlation between the students' learning behaviors and academic performance by analyzing learners' records, such as uploading assignments, posting on forums, taking online quizzes, and so on. Many studies showcased interesting insights on how the learners' behaviors are represented in the learning process in online courses and their influences on the academic performances. However, many of the studies solely focused on the learners' activities that took place on the MOOC platform. As some researchers claimed [13, 49], we saw the

lack of in-depth discussion on the learners' activities that take place outside the MOOC platform.

Furthermore, relatively few studies focused on identifying the students' problem-solving approaches in MOOCs. Among the previous studies we reviewed, many examined the problem-solving patterns of students by analyzing the activity logs. Lee [32] conducted a clustering analysis on the problem-solving patterns of students using hierarchical clustering algorithms. The study used the clickstream data and the number of problems solved to analyze how they are correlated with the course completion rates. In another study [28], students were clustered based on the measures of performances, which include the number of attempted and solved problems, the number of steps taken, time spent, and final grades. The study recognized the difficulties in finding discernible clusters that could better predict performance and serve as a basis for personalization. While these studies focus on the quantitative analysis based on the log records from the platform, Alexander et al. [45] used a qualitative analysis using an online survey and natural language processing. It uses topic modeling method to discover the topics from the survey responses. Based on the topic keywords obtained through the topic modeling, the study identifies common problem-solving procedures described by the students.

2.2 NLP applications in MOOCs

In recent years, there is a fast-growing number of studies that adopt NLP methods in their research on MOOCs. Topic modeling is one of the widely used methods among the previous studies. Liu et al. [35] and Peng et al. [41] used Behavior–Emotion Topic Model (BETM) method to discover the students' sentiments towards course contents from the course reviews. Other topic modeling techniques such as Brown clustering, Naive Bayes Classifier, and Latent Dirichlet Allocation (LDA) were also used to categorize discussion forum threads into sentiments and topics [51, 53, 50, 12], which provided valuable information for studying the relationships between sentiment and course dropout. There is a recent movement towards using language transformers, such as Bidirectional Encoder Representations from Transformers (BERT), to enhance the topic modeling performance [5, 21].

Atagün et al. [5] observed the performance of topic modeling by clustering vector representations obtained from BERT and LDA models. According to the study, transformer-based approach proved to have contributed significantly to improving the performance of topic modeling.

2.3 Task-Structure framework

Task-structure framework is a concept that helps identify problem solving strategies and subtasks, which must be fulfilled to complete the entire task. It describes possible ways a task can be decomposed into more manageable subtasks that need to be solved to complete the entire task. Chandrasekaran [10, 11] and Stroulia et al. [47] used a task-structure approach to design problem-solving for a complex AI problem. They both developed a task structure by conducting a task analysis, which involves decomposing a task into subtasks, applicable methods for it, and the knowledge requirements for the methods. Silverman et al. [46] used task-structure in the context of physical education to examine the interrelationships between the ways teachers structure practice tasks and practice variables for students of differing skill levels. The study suggested that skill level, task structure, and practice are clearly related.

Filling the Research Gap

After an extensive literature review, see Table 2.1 for a list of key examples, we identified two potential research gaps: (1) Many previous research focused on the use of event log or clickstream data for learners' behavioral analysis or academic performance prediction, but we saw relatively small number of research dealing with students' problem-solving approaches. (2) Topic modeling is one of popular NLP techniques used to analyze the student's reviews or posts, but they were conducted mainly by statistical models such as LDA or LDA-modified models. These models may be less accurate as they do not have an ability to capture contextual information.

This thesis aims to fill the research gap by (1) suggesting a research method that can analyze the MOOCs users' problem-solving approaches at scale. (2) proposing an NLP

Table 2.1: Key examples of research papers investigated in the literature review

Problem-solving & Learning Behavior Analysis		
Author	Data	Description
Jasmine Paul at al., (2019) [1]	548 student grades	Compared academic performance of online learners vs. classroom learners
Radhika Santhanam et al., (2008) [2]	134 student GPA records	Analyzed the impact of self-regulatory skills on the learning outcomes of online learners
Khe Foon Hew (2016) [3]	Review comments of 965 students	Analyzed the design factors of a MOOC that influence the student's engagement
Philip J. Guo et al. (2014) [4]	6.9 million video watching sessions from four MOOCs	Analyzed used engagement time and problem attempt records to measure the impact of video production on student engagement
Fernanda Cesar Bonafini, (2017) [5]	817 video watching records of students	Built a predictive model to predict the probability of MOOC completion
Matthieu Cisel, (2018) [6]	7,614 survey responses	Conducted the qualitative analysis of responses to study the interactions between course users in MOOCs
George Veletsianos et al., (2015) [7]	Interviews from 13 students in MOOCs	Examined the learners' activities and experiences in MOOCs
Xiaoliang Zhu et al., (2021) [10]	Behavior data of 76,843 learners from a MOOC platform	Analyzed the students' behavior data, such as progress of quiz, scores, etc., to predict the students' academic performance
Youngjin Lee, (2018) [19]	Log files of 4,337 students	Analyzed the log files that captured activities of weekly homework and quiz problems to cluster students by problem-solving patterns
NLP Applications in MOOCs		
Author	Data	Description
Sannyuya Liu et al., (2019) [22]	12,524 course reviews	Used BSTM model to detect topics discussed in the reviews
Miaomiao Wen et al., (2014) [24]	36,590 discussion posts from three MOOC courses	Conducted a sentiment analysis of words using a sentiment lexicon map to observe the relation between opinion expressed by students and the dropout rate
Xiao Yang-Cai et al., (2021) [25]	7,836 course reviews	Used LDA model to conduct topic modeling
Jovita M. Vytasek, (2017) [26]	813 posts from a MOOC's discussion forum	Used MALLET toolkit to conduct topic modeling

pipeline that applies advanced preprocessing methods, contextual language model, and text summarization model to enhance the accuracy and readability of topic modeling's results.

Chapter 3

Methods

This chapter describes the details of data collection and NLP techniques used to capture students' problem-solving information from the collected free-text survey responses. It describes the technical concepts integrated in the NLP pipeline, such as preprocessing, topic modeling, and text summarization. The last section of this chapter discusses the education theories that are used to interpret the findings in the lens of education. The full workflow of the research is illustrated in Fig 3-1.

The research consists of five main phases. The first phase is the data collection, where an online survey was used to collect responses from the students enrolled in a MOOC. The second to fourth phases are inter-connected NLP pipelines designed to extract the problem-solving information from the collected responses; these three phases are preprocessing, topic modeling, and text summarization. The last fifth phase uses educational frameworks to analyze the results through the educational lens and draw educational implications. Each of phases is designed to deliver specific tasks as described below.

1. **Data collection**: Free-text responses are collected from students taking a MOOC, which describe students' approaches in solving problems. The survey was done through the built-in A/B Testing feature in the MOOC platform (https://www.edx.org/). See Section 3.1 for details.

2. **Preprocessing**: Data cleaning and transformation are conducted to minimize the noise and remove redundant information. This phase involves 8 steps of preprocess-

ing; sentence tokenization, stopwords removal, expansion of contraction, spelling correction, lemmatization, collocation detection, POS filtering, and key phrase extraction. See Section 3.2 for details.

3. **Topic modeling**: This phase detects the types of problem-solving methods students discussed in the survey responses. This paper explores two different types of topic modeling approaches: Bag-of-words and Embedding based models. See Section 3.3 for details.

4. **Text Summarization**: A narrative summary is generated for each topic. This task is performed to help end-users easily understand the context and details of the topic by providing readable summaries rather than keywords. See Section 3.4 for details.

5. **Analysis on Educational Implications**: The identified problem-solving approaches are translated into educational implications using existing educational theories, i.e. task-structure and active/passive learning frameworks. See Section 3.5 for details.

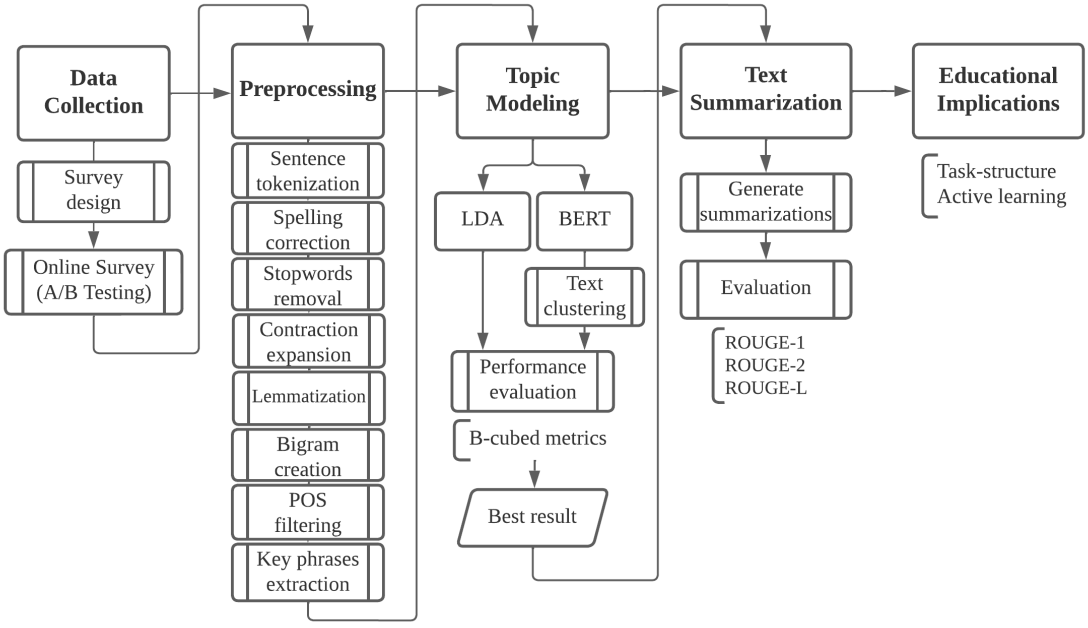


Figure 3-1: Research workflow of capturing students' problem-solving approaches in MOOCs. The workflow has five phases each with inter-connected sub-components.

3.1 Data Collection of Problem-solving Approaches

Data was collected from the MIT 6.00.1x course, a MOOC hosted by edX (https://www.edx.org/) using curriculum developed by MIT. It was taught over nine weeks in Spring 2021. After completing an exercise, students were randomly selected to respond to eleven survey questions asking for feedback using the built-in A/B testing feature in edX.

In total, 44,864 responses were collected from 5,121 students. Among these responses, we selected 7,482 (16.7%) responses for Question 2 to conduct our research. The question asks about the students' problem-solving approaches. Please see Section 3.2.2 for details.

3.1.1 Course

The course 6.00.1x is an introductory computer science course which teaches basic Python programming skills and general computer science concepts. Students can watch lectures and complete coding exercises based on the presented material. The exercises have two different formats. Finger Exercises ("FEX") are shorter, ungraded assignments meant to help students understand a concept. Problem Sets ("PS") are longer, graded assignments meant to test knowledge and demonstrate applications of certain skills. The survey questions were asked in response to Finger Exercises and Problem Sets 1, 2 and 4, where the number denotes the week which they occurred in the course. Table 3.1 provides a description of each exercise where the surveys were conducted.

Week	Problem	Description
1	FEX 1	Boolean operations
1	PS 1	String processing
2	FEX 2	Bisection search
2	PS 2	Bisection search applications
4	FEX 4	Recursion
4	PS 4	Game creation

Table 3.1: Programming exercises by weeks where the surveys were conducted

3.1.2 Survey questions

The survey questions were consisted of 5 free-text questions and 6 multiple-choice questions. Survey participants were asked mainly about their learning experiences, particularly related to the course difficulties and the self-reflections. The list of questions asked in the survey is shown in Table 3.2. For this research, the free-text question (Q2) is used, which asks: *"Please outline your approach to solving this exercise. For example, you can describe how you may have corrected your problem-solving process."*

For reference, we included the survey results of Multiple-choice Questions (MCQ) in Appendix B.

Table 3.2: Questions asked in the survey

No.	Questions	Format
Q1	Reflecting on these exercises, is there any other feedback you would like to provide?	Free-text
Q2	Please outline your approach to solving any of these exercises. For example, you can describe how you may have corrected your problem-solving process	Free-text
Q3	How challenging were these exercises?	MCQ
Q4	How prepared did you feel for these exercises?	MCQ
Q5	What previous/external resources were helpful for these exercises? Select all that apply	MCQ
Q6	If you answered OTHER above, you may explain here	Free-text
Q7	How motivated are you to continue this course?	MCQ
Q8	Do you find these exercises useful?	MCQ
Q9	Can you describe a specific situation in which you will use this knowledge?	Free-text
Q10	Please choose 2 or 3 values that are most important to you	MCQ
Q11	How does taking these exercises reflect and reinforce your most important values? Focus on your thoughts and feelings, and don't worry about spelling, grammar, or how well written it is.	Free-text

3.2 Text Preprocessing

There is a large amount of noise in the text data such as stopwords, misspellings, irrelevant information, etc. To reduce the noise level, we employed a chain of preprocessing steps consisted of following eight steps: 1) sentence tokenization, 2) spelling correction, 3)

stopwords removal, 4) contraction expansion, 5) lemmatization, 6) collocation detection, 7) POS filtering, and 8) key phrases extraction. 78% of the data was discarded during the preprocessing, but the performance of topic modeling improved significantly thanks to the improved data quality.

3.2.1 Sentence Tokenization

Each response is tokenized (split) into sentences. This is because a single response may contain multiple sentences that discuss about different topics. By tokenizing the responses into sentences, it can reduce the conflicting topics within a single entry and each sentence with an independent topic can serve as a separate single entry. To perform the sentence tokenization, *sent_tokenize()* tokenizer from the Natural Language Toolkit (NLTK) library[1] is used. It divides a string (response) into a list of substrings based on the predefined delimiters. It contains trained data to identify sentence structures, as well as considers the punctuation (e.g. periods, commas, etc.) in the sentences.

3.2.2 Spelling Correction

SymSpell [20], which is based on the Symmetric Delete spelling correction algorithm, is used to correct the spelling errors. SymSpell relies on frequency dictionaries derived from a large collection of English books, which is used originally for [30]. SymSpell is selected over other available algorithms for its fast speed and ability to handle complex expressions. SymSpell was able to recover sentences like *"Wekeep readin th eslides inthe lcture"* to *" We keep reading the slides in the lecture"*. SymSpell took approximately 821 words / second to scan and correct our text data. As a side effect, it removes punctuation. For this reason, the sentence tokenization must be conducted in prior to this step as it relies on some of punctuation to perform the task.

[1]NLTK: https://www.nltk.org/

3.2.3 Remove Stopwords and Contraction/Abbreviation Expansion

To remove stopwords, we used *simple_preprocess()* function from the Gensim Library[2] and the stopwords list from Natural Language Toolkit (NLTK). *simple_preprocess()* converts a document into a list of tokens. Then they are iterated through the NLTK's stopwords list to check if a token is a stopword or not; the token is removed if it is a stopword.

Along with this process, the contractions and abbreviations are expanded. Contractions are words or combinations of words which are shortened by dropping letters and replacing them by an apostrophe. For example, *we're = we are; we've = we have; I'd = I would.*

Figure 3-2: Contraction Expansion

This process is necessary to recognize contractions as abbreviations for a sequence of words, otherwise *we're* and *we are* are considered as two completely different words. This could make the computation more expensive by increasing the dimensionality of the document-term matrix. We used English contractions found at [14] to perform this task. Additionally, we included the following abbreviations in Table 3.3 for expansion, which were commonly found in our data.

Abbreviation	Expansion
ai	artificial intelligence
a.i.	artificial intelligence
ml	machine learning
ar	augmented reality
vr	virtual reality
cs	computer science
no.	number
T.A.	teaching assitant
TA	teaching assistant
pyhton	python
phyton	python
sypder	ide

Table 3.3: Abbreviations included in the contraction list for expansion

[2]Gensim: https://pypi.org/project/gensim/

The list was also used to correct misspellings which were unrecognized by SymSpell, such as *'phyton'* and *'pyhton'*, and to convert a brand name into a general term, such as *'spyder'* to *'ide'*.

3.2.4 Collocation Detection

A collocation is a phrase of two or more words representing a common expression. By looking into the collocations in the responses, we can gain insights into phrases which frequently appear in the student responses. Here, we used *models.Phrases()* function from the Gensim library to detect and create Bigrams, which are two adjacent words with a high occurence. It automatically detects common phrases, such as multi-word expressions and word n-gram collocations, from a stream of sentences by scoring the words occuring together using the equation below.

$$\boldsymbol{score(w_i, w_j) = \frac{count(w_i w_j) - \delta}{count(w_i) \times count(w_j)}} \tag{3.1}$$

Where,

$count(w_i w_j)$ is the frequency of word w_i and w_j appearing together

$count(w_i)$ is the frequency of words w_i's appearance

$count(w_j)$ is the frequency of words w_j's appearance

δ is a discounting coefficient which prevents from creating too many phrases

To run *Phrases()*, you need two parameters; *min_count* and *threshold (δ)*. The δ is used as a discounting coefficient and prevents too many phrases consisting of very infrequent words to be formed. The bigrams with score above the chosen threshold are then used as phrases [38]. The higher the values of these parameters, the harder for words to be combined. Using *min_count=4, threshold=10* produced the most sensible outputs based on the human judgment. Examples of created bigrams are shown in Appendix A.1.

3.2.5 Part of Speech (POS) Filtering

POS filtering was used to detect and filter out sentences with poor structures. We particularly looked for sentences that do not have a Verb or a Noun and removed them from the dataset. To build the filter, we first used *SpaCy* library[3] to classify the POS tag of a word; a process of marking up the words in a sentence to a particular part of speech based on its definition and context. Fig. 3-3 is an example of POS tagging performed by the *SpaCy* library.

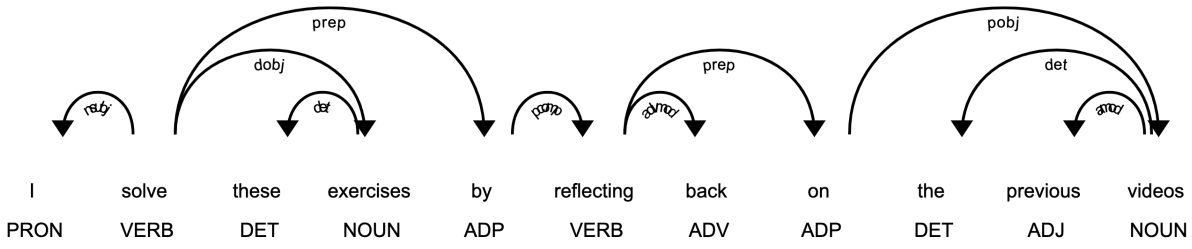


Figure 3-3: Part of Speech tagging and syntactic dependencies of a sample sentence

Usually, words can fall into one of the following major categories: N(oun), V(erb), Adj(ective), Adv(erb). It is shown in Fig 3-3 how each word in a sentence can be categorized into one of the POS. Once the POS tagging is completed, we used this information to identify and remove poorly written sentences, which don't have a Verb or a Noun.

3.2.6 Denoising through Key Phrase Extraction

The key phrase extraction is an automated process of extracting words and phrases that are considered most relevant to the input text. This is an important preprocessing step that helps remove irrelevant information (noise) from the dataset. There are many available packages and methods to perform this task (e.g., Rake, TF-IDF, TextRank, etc.). Here, we used KeyBERT [23] for its ability to capture contextual information and produce outstanding results. It uses BERT embeddings to get document-level and word-level representations, then uses cosine similarity to find the words that are most similar to the document [23]. Please see Figure 3-4 for the inner process of KeyBERT.

[3]SpaCy: https://spacy.io/

There are two key parameters to configure:

- *keyphrase_ngram_range*: This sets the length (words) range of the result. It takes in the tuple format of (X, Y), where X = min, Y = max INTEGER values. The values are set to (3,11) as it showed the best performance in the text clustering. If the maximum value (Y) is bigger than 11, the clustering accuracy started to decrease.
- *nr_candidates*: This increases the diversity in the key phrases by setting a higher value. There is a tradeoff between accuracy and diversity. If you increase the nr_candidates, you will get diverse keywords, but that are not very good representations of the document. would advise you to keep nr_candidates less than 20% of the total number of unique words in your document. [19].

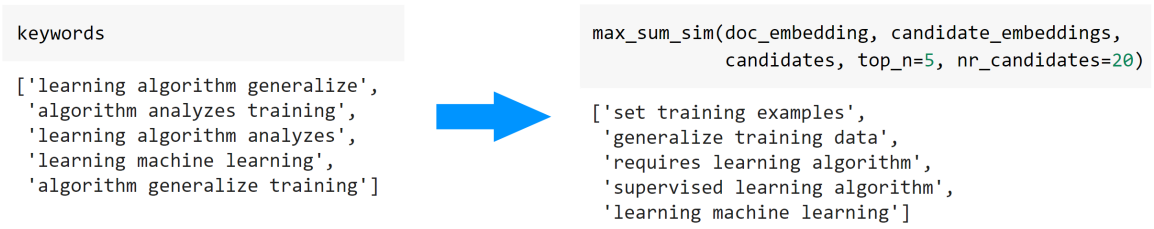


Figure 3-4: Examples of keyBERT outputs. A relatively high nr_candidates will create more diverse keyphrases [19]

The maximum value of *keyphrase_ngram_range* has a significant impact on the topic modeling. For this reason, we created a dedicated section that covers this issue in Section 4.2.

3.3 Grouping Problem-solving Approaches with Topic Modeling

The following two approaches are implemented to perform the topic modeling:

1. **Latent Dirichlet Allocation (LDA)**[7] and **Gibbs Sampling Dirichlet Multinomial Mixture (GSDMM)**[54], which operate based on the Bag-of-Words representation and probability distribution of words

2. **Bi-directional Encoder Representations from Transformers (BERT)**[16] and **Density-based spatial clustering of applications with noise (DBSCAN)**[17] algorithm, which operate based on the embedding and semantic similarity measurements

3.3.1 Bag-of-Words based Topic Modeling

In this approach, LDA and GSDMM models are used. Both LDA and GSDMM are popularly used to discover hidden topics from a large-scale text corpus.

- LDA is an unsupervised learning model that considers documents as bags of words. It assumes that the each document is a mixture of topics and each topic is a mixture of words.
- GSDMM is a modified LDA and well-known for its outstanding performance on short text. With 63.8% of our text inputs being less than 10 words long, we decided to include GSDMM to compare the performance with LDA.

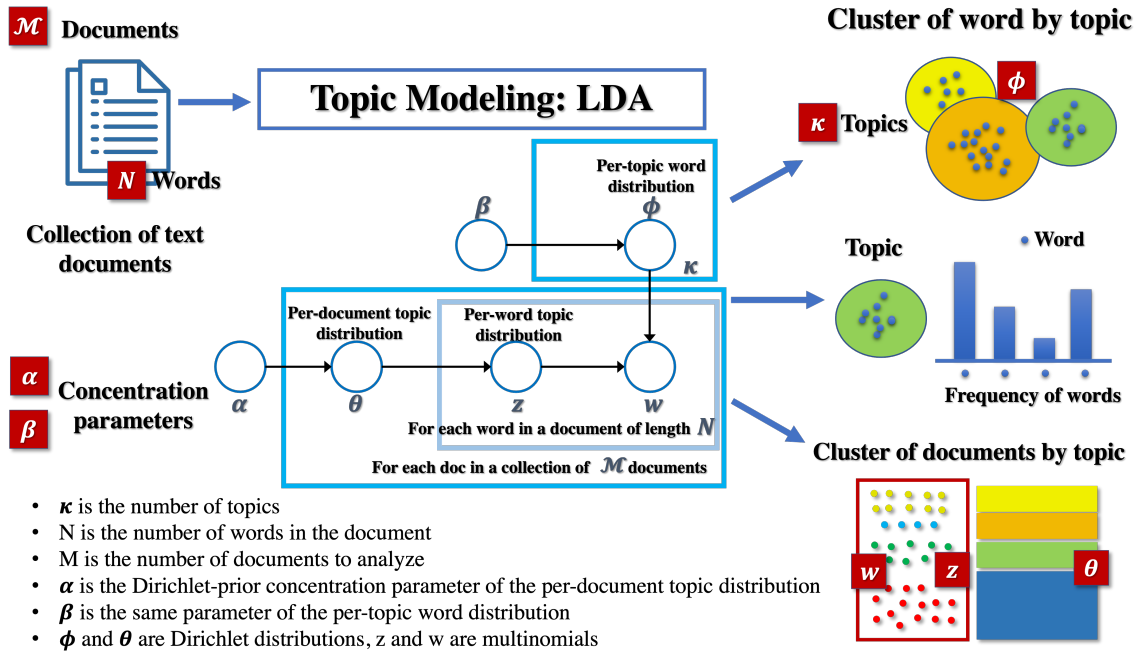


Figure 3-5: Topic modeling process of LDA [3]

We used the Gensim library [4] to implement the LDA, and MovieGroupProcess library [5] to

[4]Gensim: https://pypi.org/project/gensim/
[5]MovieGroupProcess: https://github.com/rwalk/gsdmm

implement GSDMM. There are three main hyperparameters for both LDA and GSDMM; α controls the number of topic expected in the document, β controls the distribution of words per topic in the document, and K defines how many topics we need to extract. After running tests, we used a setting of α = 0.3, β = 0.005 for their good performance. See Figure 3-5 for detailed inner workings of topic modeling.

K-value selection

The number of topics (K-value) is one of the crucial hyperparameters that determines the results of LDA and GSDMM. The K-value is manually entered into the model. The most popular method to select the optimal K-value is to use the coherence scores. The coherence scores in topic modeling measure how interpretable the topics are to humans. Here, we used the coherence metrics called *CV*. It creates content vectors of words using their co-occurrences, then calculates the score using normalized point-wise mutual information (NPMI) and the cosine similarity.

NPMI is the normalized variant of pointwise mutual information (PMI). PMI is a rank measure the likelihood of co-occurrence of two words, taking into account the fact that it might be caused by the frequency of the single words. Hence, the algorithm computes the (log) probability of co-occurrence scaled by the product of the single probability of occurrence as follows [2]:

$$\boldsymbol{PMI(a,b) = log(\frac{P(a,b)}{P(a)P(b)})} \tag{3.2}$$

Cosine similarity is a measurement that quantifies the similarity between two or more vectors. It is the cosine of the angle between vectors and described mathematically as the division between the dot product of vectors and the product of the euclidean norms or magnitude of each vector.

$$\boldsymbol{Cosine\ Similarity\{A,B\} = \frac{A \cdot B}{\parallel A \parallel \parallel B \parallel} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}} \tag{3.3}$$

Here, A_i and B_i are components of vector A and B respectively [15]:

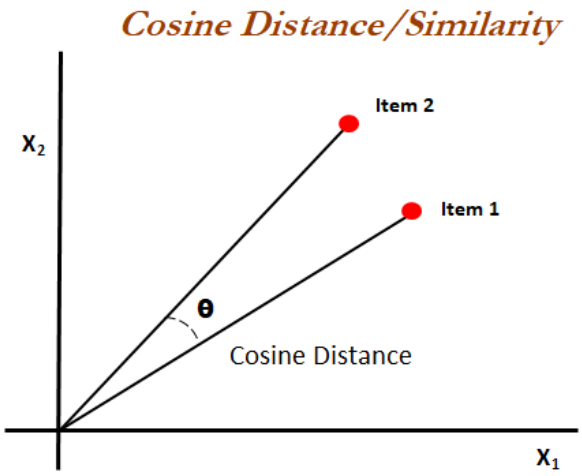


Figure 3-6: Cosine similarity measurement [15]

Fig. 3-7 shows the coherence scores of LDA and GSDMM by topic numbers in *range (start=2, limit=30)*. Picking the high topic number merely to get the higher coherence score is not a recommended approach. It is recommended to pick the one where there is a change of slope from steep to shallow (an elbow). Here, K=12 seems to be a good pick for both LDA and GSDMM, which has the score of 0.634 and 0.3292 respectively.

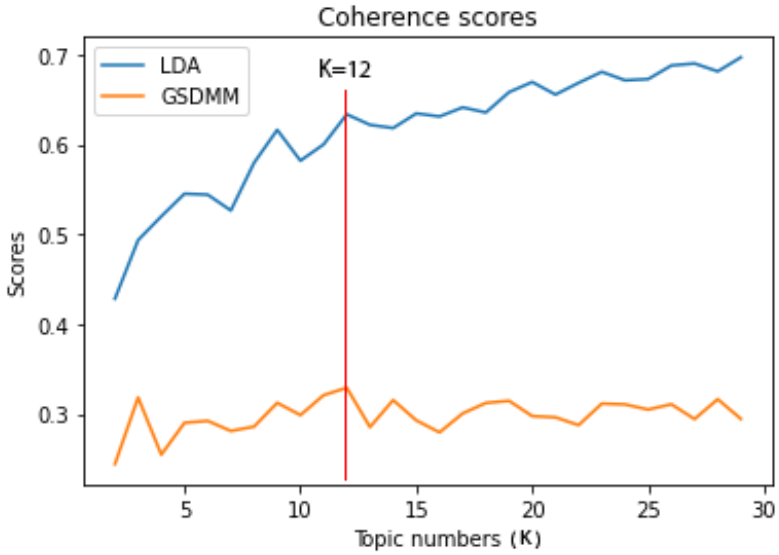


Figure 3-7: Coherence Scores per topic number (K) of LDA and GSDMM

3.3.2 Transformer-based Topic Modeling

A shortcoming of bag-of-words based topic modeling is that it cannot capture contextual information as it solely relies on the probability distributions of words. To overcome this problem, we implemented a transformer-based topic modeling using BERT embedding and DBSCAN clustering algorithm. This can reflect contextual information in the topic modeling process. The process is composed of four main stages as illustrated in Fig. 3-8.

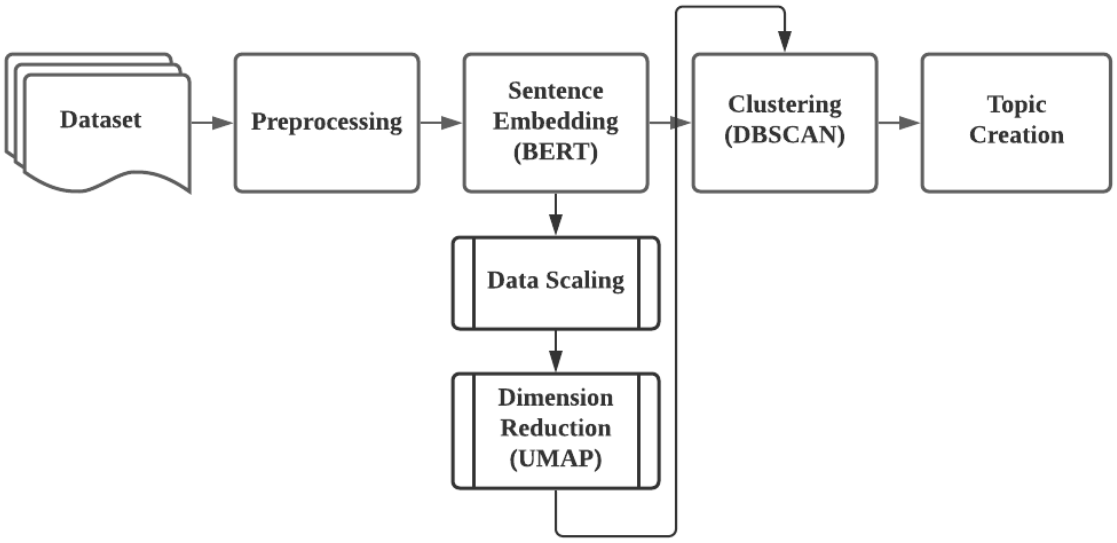


Figure 3-8: Workflow of transformer-based topic modeling

Sentence Embedding

Sentences must be transformed into an embedding representation for the text clustering. These embeddings can then be compared with, e.g., consine-similarity to find the sentences with a similar meaning. The sentence-level embedding is conducted using a pre-trained BERT model from Huggingface[6]. We selected the model *'sentence-transformers/all-MiniLM-L12-v1'* which is bases on PyTorch and Transformers. The model is trained on over 1 billion sentences and maps sentences to a 384 dimensional vector space. This model shows the highest accuracy in clustering texts over other available models . Please see Section 4 for the details of performance evaluations.

[6]Huggingface: https://huggingface.co/models

Data Normalization

By normalizing the inputs we can bring all the input features to the same scale. This is an important process because the input features are often presented in significantly different ranges. This leads to the uneven distribution of weights, consequently increasing the oscillating of the learning algorithm and, hence, the training time before it finds the global minima. To avoid this issue, we used Batch Normalization, which fixes the distribution of the hidden layer values as the training progresses. It makes sure that the values of hidden units have standardised mean and variance. You can implement this by using *torch.nn.BatchNorm1d()* in PyTorch[7].

$$\boldsymbol{y} = \frac{\boldsymbol{x} - \boldsymbol{E}[\boldsymbol{x}]}{\sqrt{\boldsymbol{Var}[\boldsymbol{x}] + \boldsymbol{\epsilon}}} \times \boldsymbol{\gamma} + \boldsymbol{\beta} \tag{3.4}$$

The mean and standard-deviation are calculated per-dimension over the mini-batches. By default, the elements of γ are set to 1 and the elements of β are set to 0 [8].

Dimensionality Reduction

The dimension reduction is an important step before performing the clustering as *the curse of dimensionality* could prohibit the proper use of clustering algorithms in the high-dimensional space [39]. Uniform Manifold Approximation and Projection (UMAP) [33] is used to reduce the dimensionality of embedding. It is fast and scalable, while preserving the global structure of the data. In this process the dimension is reduced to 5 dimensions down from the original 384 dimensions.

Clustering

The DBSCAN (Density-based spatial clustering of application with noise) clustering algorithm is used to perform the clustering of the text embeddings. DBSCAN is an unsupervised learning algorithm that groups together data points that are closely packed together. It can cluster non-linear groups of data points with high accuracy, which cannot be ade-

[7]PyTorch: https://pytorch.org/

[8]Batch normalization: https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm1d.html

quately performed by k-means or Gaussian Mixture EM clustering [52]. See Fig. 3-9 for an example image. There are three main hyperparameters to configure:

- ***metric*** determines the type of distance to use to measure the similarity between instances. It is set to *metric='euclidean'*.
- ***eps*** specifies how close points should be to each other to be considered a part of a cluster. If the distance between two points is lower or equal to this value, they are considered neighbors. It is set to *eps=0.5*.
- ***min_sample*** is the minimum number of points to form a cluster. It is set to the default value of 5.

DBSCAN was chosen for this task as it outperformed other clustering algorithms; OPTICS (Ordering points to identify the clustering structure) and HAC (Hierarchical Agglomerative Clustering) competed against DBSCAN. DBSCAN consistently showed high accuracies throughout the tests. The detailed test results are described in Fig. 4-5.

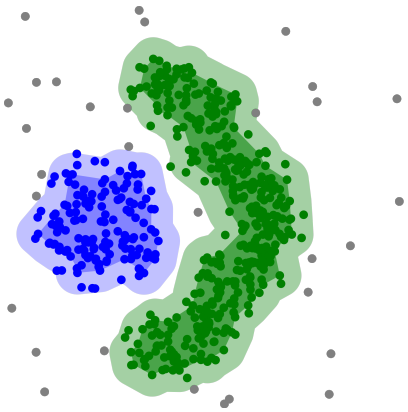


Figure 3-9: DBSCAN can find non-linearly separable clusters, which cannot be done by k-means or Gaussian Mixture EM clustering. [52]

Topic Creation

This is the last stage of transformer-based topic modeling process, where we used a *class-based TF-IDF (c-TF-IDF)* [22] to extract the topic modeling keywords from the generated topic clusters. The equation of c-TF-IDF is described below. TF-IDF is based on the importance of words that each document is composed of, whereas c-TF-IDF treats all documents

clustered in the common topic as a single document – the result can be a long series of documents per topic – and then applies TF-IDF. In this way, c-TF-IDF can demonstrate the important words in a topic rather than a document.

$$\boldsymbol{c\text{-}TF\text{-}IDF_i} = \frac{\boldsymbol{t_i}}{\boldsymbol{w_i}} \times \boldsymbol{log}\frac{\boldsymbol{m}}{\sum_j^n \boldsymbol{t_j}} \tag{3.5}$$

Where,

$\frac{t_i}{w_i}$ is the frequency of **each word** $\boldsymbol{t}$ extracted for **each class** $\boldsymbol{i}$ and divided by the **total number of words** $\boldsymbol{w}$. It is a form of regularization of frequent words in the class,

$\frac{m}{\sum_j^n t_j}$ is the **total number of documents** $\boldsymbol{m}$ divided by the total frequency of **word** $\boldsymbol{t}$ **across all classes** $\boldsymbol{n}$ [22]

3.3.3 Topic Modeling Measurement

The performance of topic modeling is measured by the accuracy of the text clustering; how well the model groups the text inputs into a correct set of topics. In total five models are evaluated, namely LDA, GSDMM, DBSCAN, OPTIC, and HAC. To evaluate any clustering output, you need gold-standard data (a.k.a ground-truth data), which serve as an answer sheet to compare with the output of the model. To create the gold-standard data, 685 responses were randomly sampled from the collected survey responses and manually clustered by humans according to their problem-solving methods. The clustering result from the model is then evaluated by the B-cubed clustering evaluation metrics [4].

B-cubed precision of an item is the proportion of items in its cluster which have the item's category (including itself). The overall B-cubed precision is the averaged precision of all items in the distribution. Since the average is calculated over items, it is not necessary to apply any weighting according to the size of clusters or categories. The B-cubed recall is analogous, replacing "cluster" with "category".

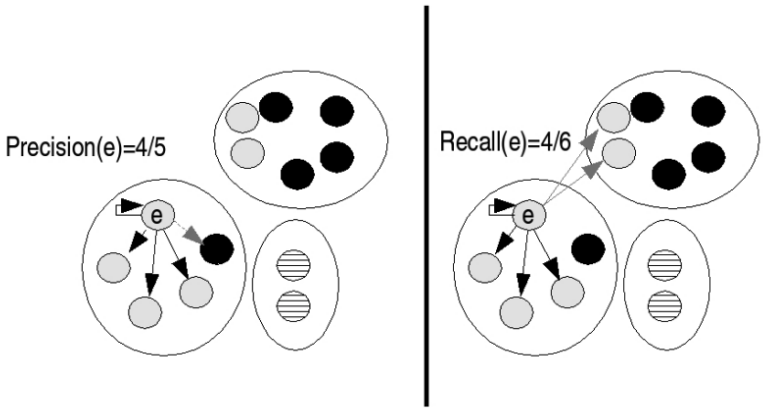


Figure 3-10: Example of computing the B-Cubed precision and recall [4]

3.4 Generating Readable Text Summarization

The text summarization model is integrated in the last phase of the pipeline to generate narrative summaries of the identified topics. By creating the summaries, the end-users is informed with rich contextual information of the identified problem-solving methods. Fig. 3-11 shows the process of text summarization.

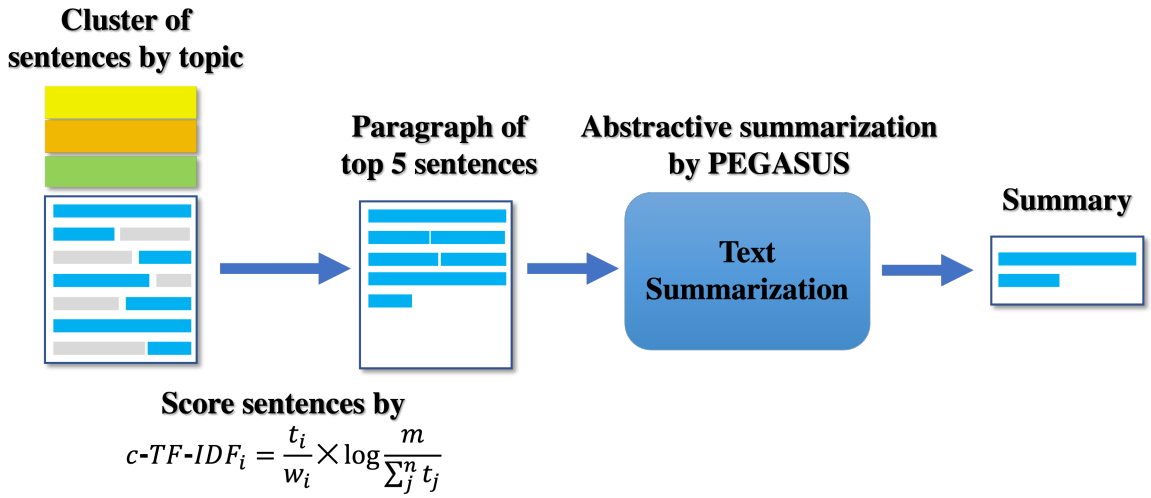


Figure 3-11: Workflow of text summarization

3.4.1 Input Data

The inputs for the summarization are created by merging five most representative sentences from each topic. The representativeness of a sentence is calculated by summing the *c*-

TF-IDF scores of all the words present in the sentence. In this way, we could identify the sentences that hold more important keywords representing the topic. Considering that unimportant words are scored significantly low and the length of a single sentence is short, it was unnecessary to normalize the summing of words. Fig. 3-12 below is an example of an input text created by merging the sentences. The generated summary of this example is *"I try to step through the problem by writing the code on the piece of paper."* Please see all the results in Table 5.2.

```
'i write the code with pen and paper and try to step through the problem. write out code on piece of
paper to try to work it through step by step. i approach these exercise by think through a solution
before i start write and try to get a basic outline of my final code. i first think how to solve it
without write then if i get stuck i image myself walk through the code step by step. i first try to
think through all of the step i will need to solve the problem and the start programming'
```

Figure 3-12: A paragraph of top five sentences from a topic that is fed into the text summarization model for summarization.

3.4.2 Text Summarization Model

The text summarization can be done in two methods; Abstractive and Extractive. Abstractive summarization attempts to generate an entirely novel reconstruction of a summary based on the key topics discussed in the text input. Extractive seeks to select a subset of important words or sentences from the existing document, then combines all of them to create the summary.

Here, we used the Pre-training with Extracted Gap-sentences for Abstractive Summarization (PEGASUS) model [55] to perform the text summarization. It demonstrates a state-of-the-art performance on low-resource summarization, which is suitable for our dataset. Based on the quantitative and qualitative evaluations conducted in Section 4.4.1, *tuner007/pegasus_paraphrase* is selected to perform the summarization as it showed the best performance in summarizing the texts.

3.4.3 Summarization Evaluation

The quality of generated summaries are evaluated in two ways. Firstly, ROUGE metrics are used to evaluate the summaries. ROUGE stands for Recall-Oriented Understudy for Gist-

ing Evaluation. It determines the quality of a summary (candidate summary) by comparing it to an ideal summary created by humans (reference summary) [34]. The score is measured by counting the number of overlapping text units, such as n-gram and word sequences between the human-generated and computer-generated summaries. The ROUGE evaluation package offers different measures such as ROUGE-N and ROUGE-L. ROUGE-N measure the matching of 'n-grams' between the two summaries.

Here, three measures are used. (1) A unigram (ROUGUE-1) measures the match rate of a single word between the summaries. (2) A bigram (ROUGE-2) measures the match rate of two consecutive words. (3) Lastly, ROUGE-L measures the longest common subsequence (LCS), meaning that it counts the longest sequence of units that is shared between the summaries. Below is the equation of this metrics.

$$ROUGE\text{-}N = \frac{\sum_{S \in \{Reference\ Summaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{Reference\ Summaries\}} \sum_{gram_n \in S} Count(gram_n)} \tag{3.6}$$

Where, $Reference\ summaries$ is the human-generated (gold standard) summaries

S is a sentence

N is the length of the n-gram (word sequence)

$Count_{match}(gram_n)$ is the number of common n-grams in candidate (model generated) and reference (human generated) summaries

The second method of evaluation is to measure the length of generated summaries. Some models were producing excessively long summaries. Therefore, I included this evaluation criteria to recognize the summaries of adequate length. The baseline of length (an ideal length) is set by the human-generated summaries, which is 64.1 characters. Based on the heuristic judgement, any summary that exceeds over 100 characters are considered too long. See Table 4.5 for details.

3.5 Educational Theories

Two educational theories are selected to guide the interpretation of findings in the educational perspective; task-structure and active/passive learning frameworks.

3.5.1 Task-structure Framework

The general practice of task-structure formulation is to first decompose a task into subtasks then determine the problem-solving methods for tackling each subtask. However, in this research, the problem-solving methods are first identified, then formulated into subtasks backwards. Considering that students come up with novel ways to approach problems in the online learning setting, constructing the task-structure from bottom to up makes it more sensible to capture the students' approaches.

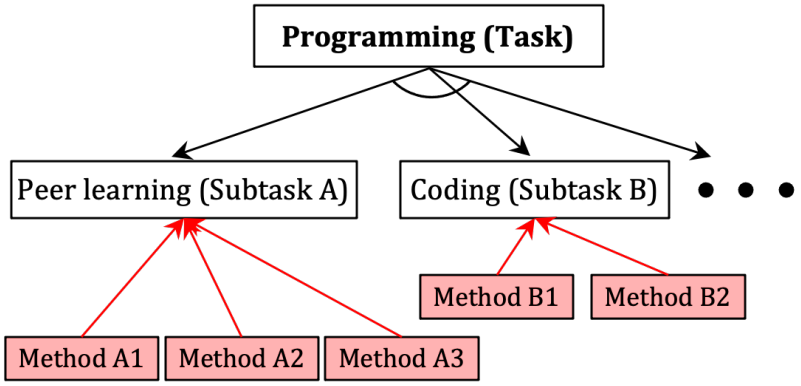


Figure 3-13: An example of bottom-up formulation of a task-structure. Subtasks are formulated based on identified methods

3.5.2 Active/Passive Learning Framework

Active learning is "a method of learning in which students are actively or experientially involved in the learning process" [9]. Examples of *Active Learning* includes online discussion/debates, group projects, and worksheets encouraging the application of new knowledge [18]. *Passive Learning* is a process where students passively receive information from the learning environment and internalize it in a form of memorization [37]. Examples of passive learning includes reading, listening to a lecture, watching a video, and looking at picture or powerpoints. Students learn by merely taking in the given information [18]. We use these definitions to categorize the problem-solving methods into active or passive learning.

Chapter 4

NLP Pipeline Sensitivity Analysis and Tuning

There are a number of NLP techniques and models included in the NLP pipeline. To produce the best output, the pipeline needs to go through a tuning process to find the optimal configuration. This chapter discusses the types and processes of tests conducted to find the best configuration of each NLP component. Fig. 4-1 below shows an overview of the performed tests.

4.1 Data Labeling

The text clustering accuracy is used as one of measurements for the NLP pipeline's performance; how accurately it can separate and group similar texts into corresponding (problem-solving) clusters. Ensuring a high-level of clustering quality is extremely important as both topic modeling and text summarization rely on the clustering output to perform their own tasks.

To conduct the performance tests, a total of 685 responses are randomly sampled from the 7,482 survey responses, as shown in Fig. 4-2 and labeled (clustered) manually in accordance to their problem-solving methods. This labeled dataset serves as a gold standard dataset to measure the accuracy of text clustering. We created 15 problem-solving clusters

Component	Parameters / Experimental setups			Evaluation metrics /methods	Section
Preprocessing pipeline	Original data, Basic preprocessing, Full preprocessing			Text clustering acc.	4.2
Preprocessing > Key phrase extraction	Max N-gram			Text clustering acc.	4.2
Sentence embedding	Pre-trained embedding models (BERT)			Text clustering acc.	4.3
Topic modeling		Algorithm	Parameter	Text clustering acc., Interpretability of keywords	4.4
	BoW	LDA	alpha beta/eta num_topics		
	BoW	GSDMM			
	EMB	DBSCAN	eps		
	EMB	OPTICS	max_eps		
	EMB	HAC	distance_threshold		
Text summarization	Pre-trained summarization models (PEGASUS)			ROUGE scores; Summary lengths	4.5
Full pipeline	Test runs			Method's sensitivity analysis	4.6
	Year 2020 dataset, Year 2021 dataset, Combined dataset				
Task-structure framework	Identified problem-solving methods			Task formulation	4.7

* BoW: Bag-of-Words
* EMB: Embedding

Figure 4-1: Descriptions of test components, parameters, and evaluation approaches

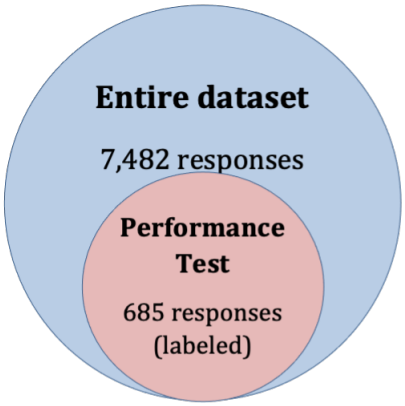


Figure 4-2: Data setup for the performance test. 685 responses were sampled from the entire dataset and labeled for the performance testing

from the sampled responses as shown in Fig. 4-3. *IDE*[1] (119; 17.4%) and *Video lectures* (108; 15.8%) are two most frequently mentioned problem-solving methods, followed by *Pen and paper* (63; 9.2%), *Instructions* (62; 9.1%), and so on. Contrarily, *Lecture slides* (13; 1.9%), *Mental process* (25; 3.6%), and *Problem breakdown* (25; 3.6%) are some of relatively less mentioned methods. To validate the accuracy and consistency of labeling, two individuals cross-reviewed the labels and went through 4 rounds of revision.

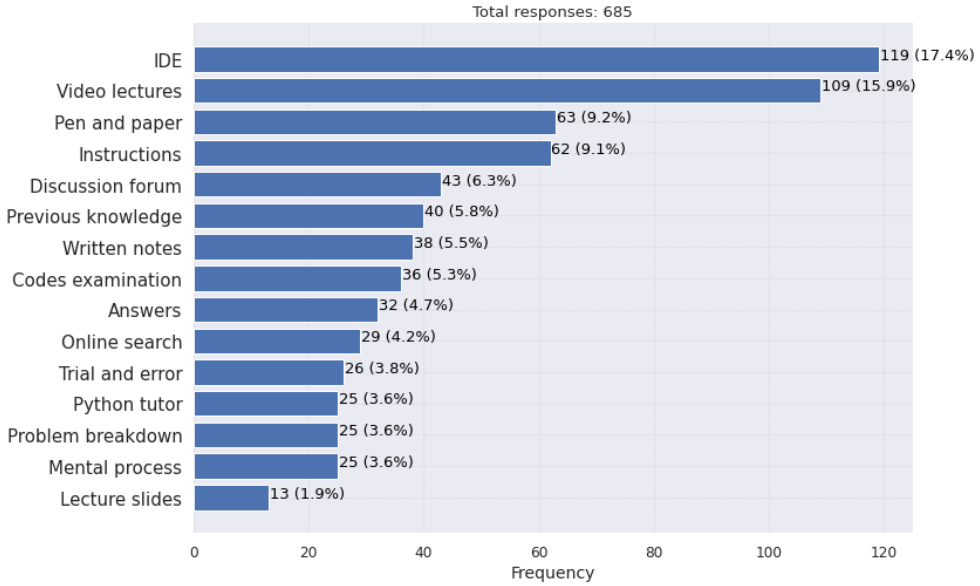


Figure 4-3: Count of clustered responses by problem-solving methods

4.2 Preprocessing

The preprocessing techniques applied in the pipeline can be largely divided into two categories: basic and advanced techniques. The basic ones are popularly used techniques, such as stopwords removal, spelling correction, lemmatization, and so on. The advanced ones, i.e. POS filtering and key phrase extraction, are particularly included for this research, given that the noise level of text data was high. The preprocessing test was designed to measure the impact of different configuration of preprocessing techniques on the clustering accuracy.

[1]IDE: Integrated Development Environment

With that said, the following three setups, as shown in Table 4.1, were used to conduct the performance tests:

1. Original data: No preprocessing is applied except for the sentence tokenization
2. Basic preprocessing: Stopwords removal, expansion of contraction, spellchecking, lemmatization, and collocation detection are applied
3. Full preprocessing: In addition to Basic preprocessing (Setup 2), POS filtering and key phrases extraction are applied

Table 4.1: Preprocessing Performance Test Setup

	Original data	**Basic preprocessing**	**Full preprocessing**
Sentence tokenization	✓	✓	✓
Stopwords removal		✓	✓
Expansion of contraction		✓	✓
Spellchecking		✓	✓
Lemmatization		✓	✓
Collocation detection		✓	✓
POS filtering			✓
Key phrase extraction			✓

Table 4.2: Clustering Accuracy (F1 Score)

	Original data	**Basic preprocessing**	**Full preprocessing**	**Increase**
LDA	0.18	0.19	0.23	5%
GSDMM	0.29	0.29	0.30	1%
OPTICS	0.40	0.38	0.37	-3%
DBSCAN	**0.63**	**0.73**	**0.78**	15%
HAC	0.57	0.68	0.75	18%
Mean	0.414	0.454	0.486	7.2%

Table 4.4 shows the test results. The full preprocessing setup showed the highest clustering accuracy for all algorithms except for OPTICS. In average, the accuracy increased by 7.2 %. HAC shows the biggest increase of 18%. In terms of overall performance, DBSCAN shows the best performance, marking 78% accuracy.

Fine-tuning Key Phrase Extraction

Tuning *max N-gram* of key phrase extraction has a substantial influence on the clustering performance. Fig 4-4 shows the result of the performance tests, which illustrates the

changes in the clustering accuracies based on the different *max N-gram* values. The combination of *Max N-gram = 11* and *DBSCAN* clustering algorithm marked the highest accuracy of 0.76%.

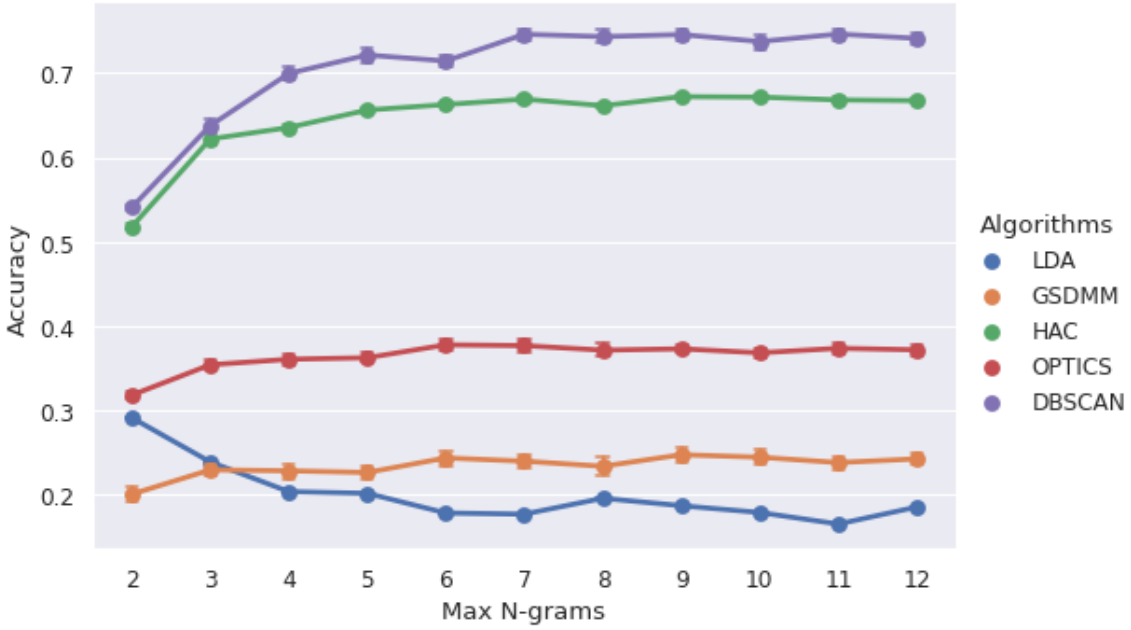


Figure 4-4: Clustering accuracy (F1 Score) by algorithms with different max N-gram values

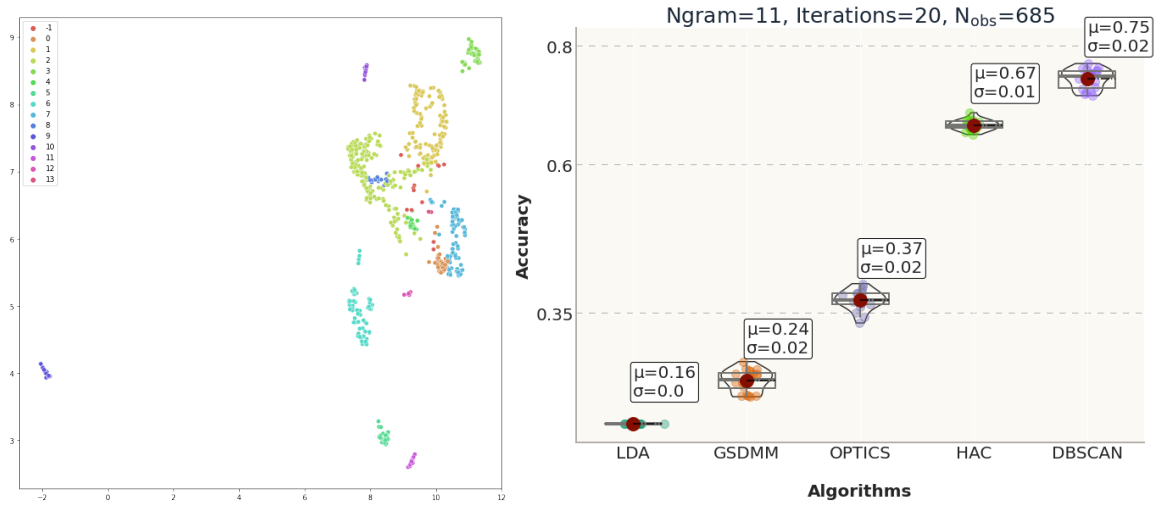
4.3 Topic Modeling

4.3.1 Text Clustering Test

To produce the best topic modeling result, its pipeline has to be equipped with a high-performing clustering algorithm. With the optimal configuration identified in the prior chapters, a series of text clustering tests were conducted to find the best performing clustering algorithm. We configured the setting around the best accuracy performed by DBSCAN.

The test setup is configured as below:

- Test iteration: 20
- Preprocessing: Full preprocessing setup as described in Section 4.2
- Max N-gram of key phrase extraction: 11
- Testing algorithms: LDA, GSDMM, OPTICS, HAC, DBSCAN



(a) Visualization of DBSCAN's result

(b) Clustering accuracy by algorithm

Figure 4-5: (a) Visualization of problem-solving approaches clustered by DBSCAN; (b) Clustering accuracy by algorithms with 20 test iterations and max N-gram of key phrase extraction set to 11

The test result is shown in Fig. 4-5 (b). DBSCAN consistently shows the best performance among the algorithms. Its mean accuracy (μ) is 75%, while LDA shows the lowest mean accuracy (μ) of 16%. The result of DBSCAN is visualized in the vector space in Fig. 4-5 (a).

Fig. 4-6 shows the result of topic modeling conducted with the optimal configuration. Y axis shows the topic clusters and their top 3 keywords. X axis is the number of documents (sentences) assigned to that specific topic, in another word the topic frequency. The model generated 16 topics from the sampled 685 responses, with one topic *{defined, methods, previously}* clustered as an outlier (topic number 16); which means it does not belong to any topic. The result indicates that *{video, lecture, watching}* is the most frequently mentioned problem-solving method by the students, followed by *{ide, code, test}* and *{write, pen_paper, steps}*.

The model was able to capture all the topics that were created during the data labeling. However, there is a slight discrepancy between the human and the model's perception of text information. The model recognizes *'read questions'* and *'read instructions'* as different topics, whereas we (humans) labeled them as a common topic, considering that *'instructions'* was given as part of *'questions'*. You can see the model created two separate topics

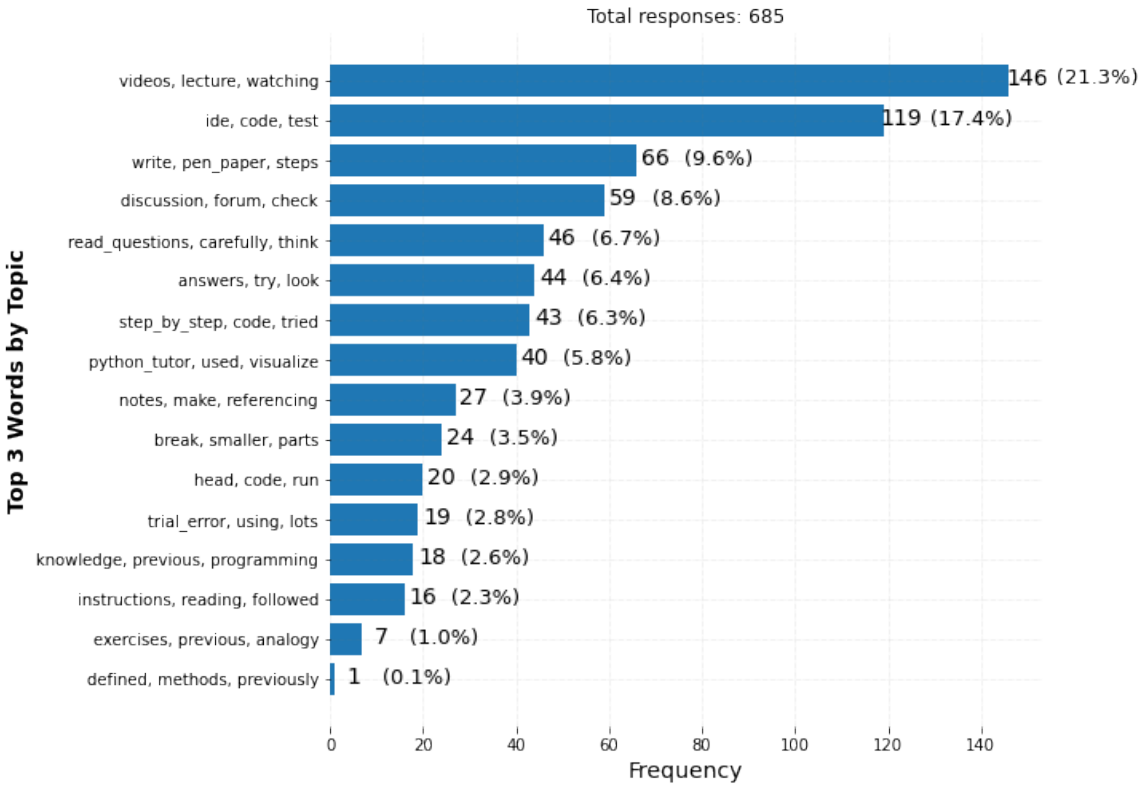


Figure 4-6: Topic clusters and keywords generated by Topic Modeling

{read_question, carefully, think} and *{instructions, reading, followed}*.

Furthermore, the model sometimes pick up popular (frequently appeared) words more sensitively than the important keywords. For example, the model grouped *'Youtube videos'* and *'lecture videos'* as a common topic *{video, lecture, watching}*. Because they are two different behaviors exploiting different learning resources, they need to be differentiated. But this could be challenging for the computer to understand. Such cases are some limitations of topic modeling observed during the tests.

4.3.2 Keywords Interpretations

Table 4.3 shows how problem-solving methods are interpreted based on the keywords. We were able to interpret 14 topics, but the rest 2 topics (Topic 15 and 16) were difficult to understand. Nevertheless, the overall quality of keywords is satisfactory with high accuracy and interpretability.

Table 4.3: Interpretation of Topic Modeling Keywords

Topic	Top 3 words	Interpretation
1	videos, lecture, watching	Watch lecture video
2	ide, code, test	Test code on IDE
3	write, pen_paper, steps	Write on paper
4	discussion, forum, check	Check discussion forum
5	read_questions, carefully, think	Re-read questions
6	answers, try, look	Look at answers
7	step_by_step, code, tried	Try code step by step
8	python_tutor, used, visualize	Use python tutor
9	notes, make, referencing	Make notes for reference
10	break, smaller, parts	Breakdown parts
11	head, code, run	Run code mentally
12	trial_error, using, lots	Trial and error
13	knowledge, previous, programming	Use previous knowledge
14	instructions, reading, followed	Follow instructions
15	exercise, previous, analogy	N/A
16	defined, methods, previously	N/A

4.3.3 Method's Sensitivity Analysis

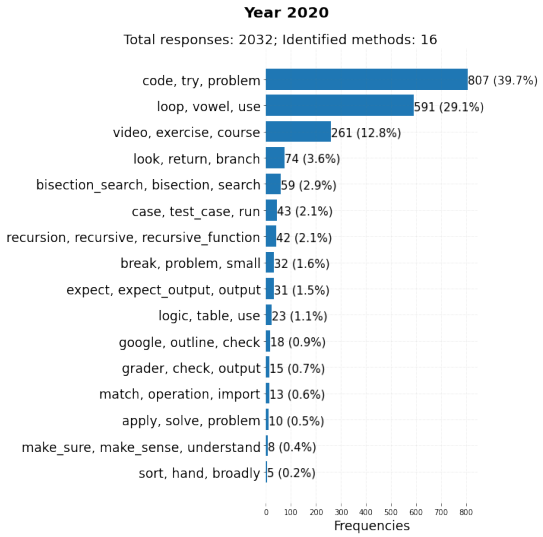
This sensitivity analysis is conducted to identify how much variations in the input data will impact the results for the proposed NLP pipeline. This is to test the robustness of the method's results and increase the understanding of the relationships between input and output. In this analysis, we mainly looked into three areas:

1. **Robustness of the NLP method:** We looked into whether the method produces reasonable results with different input data. We wanted to reduce the uncertainty of the method by examining how the method behaves on different datasets

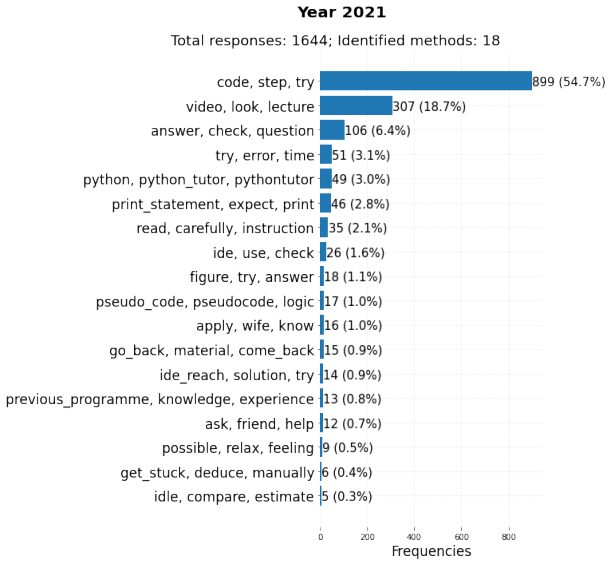
2. **Types of problem-solving methods:** We took this opportunity to observe the types of problem-solving methods used by students in different years

3. **Comparison of problem-solving methods:** We compared the identified methods for 2020 and 2021 to find their commonalities. We compared by counting the common keywords occurrences between the methods of 2020 and 2021

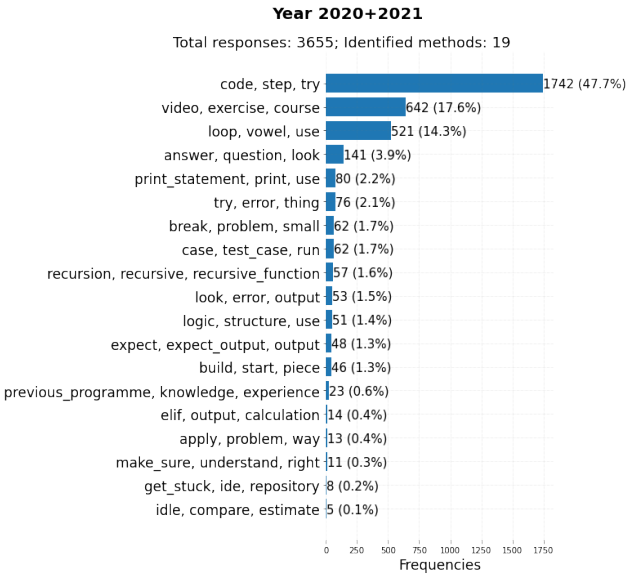
First, we ran the topic modeling model using the survey responses from year 2020 and a combined responses of 2020 and 2021. As for the combined dataset, we got 3,655



(a) Survey responses of 2020



(b) Survey responses of 2021



(c) Combined responses of 2020 and 2021

Figure 4-7: Topic modeling results of survey responses from year 2020, 2021, and combined dataset

data points after the preprocessing, down from originally 11,096 points. The results of topic modeling are shown in Fig. 4-7. The topic modeling model performed very well on both datasets. It produced a reasonable number of problem-solving methods with sensible keywords and cluster groups.

Table 4.4: The properties of the datasets

Year	Number of responses	After preprocessing	Avg. number of sentences per response	Avg. number of words per response
2020	3,614	2,032	1.52	4.61
2021	7,482	1,644	1.16	4.80
Combined	11,096	3,655	1.33	4.71

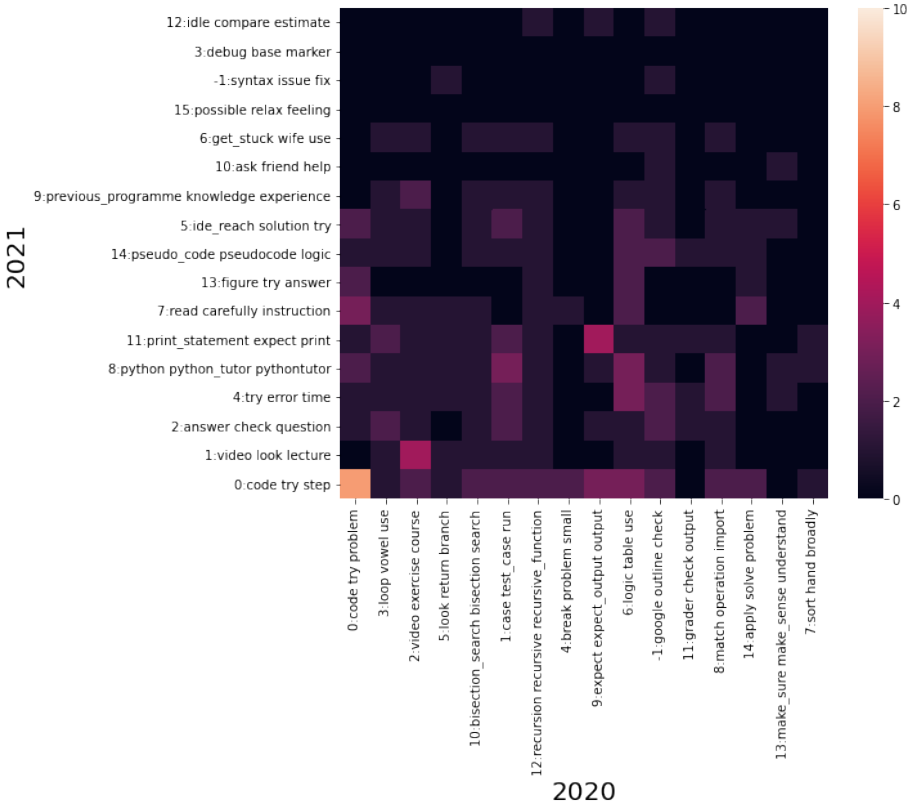
Next, we used heatmaps to visualize the similarity between the identified problem-solving methods. The similarity is measured by the occurrences of common keywords. We used top 10 keywords from each method to compare. For this analysis, we created three versions of heatmap: (1) 2020 vs 2021, (2) Combined vs 2020, (3) Combined vs 2121.

The heatmaps are shown in Fig. 4-8. For both X and Y axises of the heatmap, the most frequent topics (problem-solving methods) are laid in sequence from the lower-left. In the image (a), as the bright space indicates, popular topics in both 2020 and 2021 seem to be talking about similar methods, contrarily, while least popular topics hardly have any similarity (dark space). The image (b) and (c) also shows a tendency of high frequency, high similarity, but there are random bright dots scattered along the diagonal. We believe these dots indicate the topics that are preserved through the topic modeling process of the combined dataset.

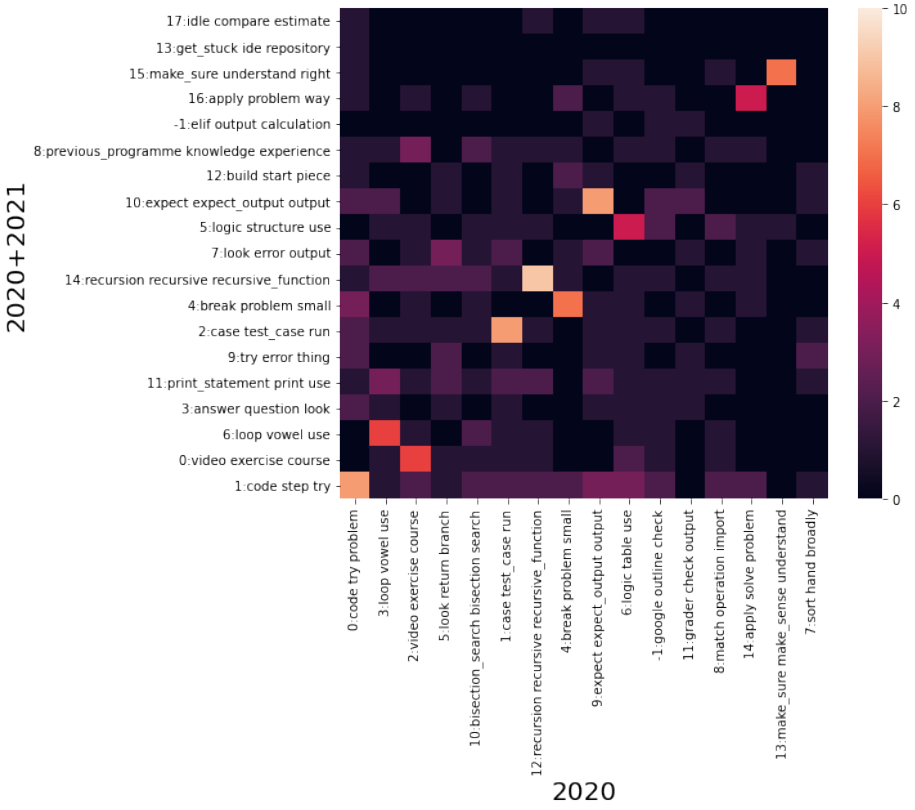
4.4 Text Summarization

4.4.1 Quantitative and Qualitative Evaluations

We used ROUGE metrics to evaluate the quality of the generated summaries. As shown in Table 4.5, three models from each Extractive and Abstractive summarization models from *Huggingface* are selected to perform the tests. The selection was made based on the



(a) Year 2020 vs. Year 2021



(b) Combined dataset vs. Year 2020

(a) Combined dataset vs. Year 2021

Figure 4-8: Heatmap of common keywords occurrences between problem-solving methods

most download counts and the highest user rates. The human-generated baseline is used to calculate the ROUGE scores of model-generated summaries.

One of ROUGE score's weaknesses is that it tends give higher scores for longer summaries as the score is calculated based on the overlapping words and the length of the longest common subsequences. Based on the human baseline measured by the human generated summaries, it is recommended that the length of summaries is around 65 characters. We selected *pegasus_paraphrase* model as it showed the best performance and an adequate length of summaries.

Table 4.5: ROUGE F1 results

Model	R-1	R-2	R-L	Avg. length (char)
Extractive				
cnn_dailymail	0.20	0.06	0.18	169.5
pegasus-large	0.23	0.10	0.22	78.7
pegasus-wikihow	0.18	0.05	0.16	319.8
Abstractive				
pegasus-xsum	0.23	0.07	0.20	85.3
pegasus_paraphrase	**0.29**	**0.12**	**0.27**	**61.9**
pegasus-reddit_taifu	0.69	0.59	0.69	322.9
Baseline				
Human	-	-	-	64.1

Table 4.6 is the result of text summarization performed by *pegasus_paraphrase*. The generated summaries provide contextual information compared to the keywords from the topic modeling, helping the end-users to better understand the context of students' problem-solving methods. The quality of the summaries is satisfying except for the last two summaries, which are grammatically sound but hard to understand the meaning.

4.5 Task-Structure Formulation

Fig 4-9 illustrates the task-structure of programming formulated based on the identified problem-solving methods. Putting together problem-solving methods that work towards the same goal helps reverse-engineer the types of tasks that students are working on.

The diagram shows that students are using diverse methods to tackle tasks like 'Strengthen

Table 4.6: Topic summarization generated by PEGASUS model

Keywords	Generated summaries
video, lecture, watching	The lecture videos gave me most of the information I needed for the exercises.
ide, code, test	I use reverse engineering to try to understand the correct answers by running the code in the ide.
write, pen_paper, steps	I need to write down the steps I need to take in order to get the desired result.
discussion, forum, check	I read the discussion board to understand why I missed the question.
read_questions, carefully, think	The way I used to correct my mistake was to read more carefully the question.
answers, try, look	When I was wrong, I looked at the answers if I didn't get it right.
step_by_step, code, tried	I tried to find errors with a step by step approach.
python_tutor, used, visualize	I used python tutor to understand the code.
notes, make, referencing	I look at my notes and make a program from them.
break, smaller, parts	To see how python calculates, I broke the problem down into the most basic parts.
head, code, run	I think about how the code will run and do it in my head.
trial_error, using, lots	It was difficult to find the problems in my code during the trial and error process.
knowledge, previous, programming	I have used previous programming knowledge to create programs.
instructions, reading, followed	I just followed the instructions.
exercise, previous, analogy	I have run into a lot of errors while writing solutions to previous exercises.
defined, methods, previously	I looked at methods before they were defined.

knowledge' and 'Clarify code logic' in order to solve programming problems. We can also learn that students engage in other task activities such as 'Coding & degugging', 'Understand problems', and 'Social/peer learning'.

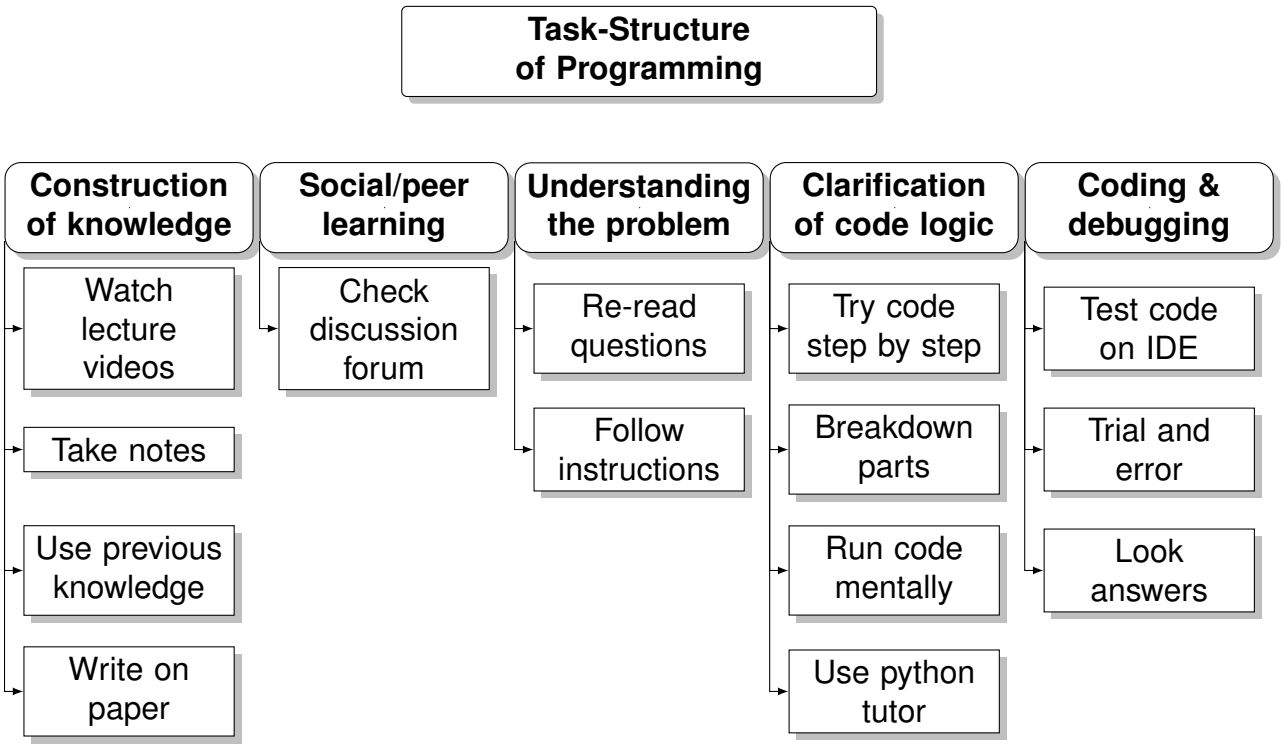


Figure 4-9: Task-structure of programming formulated based on identified problem-solving methods

Chapter 5

Experiments

This chapter describes the research setup and presents the results of each research component. Firstly, the experimental setup is described, specifically on how the NLP libraries and models are configured based on the performance test conducted in Chapter 5. Then it discusses and evaluates the results obtained, followed by their educational implications.

Each of the section in this chapter aims to address the overarching and sub research questions as follows:

- Overall research results: *RQ 1. (Overarching question) Can we capture the students' problem-solving methods carried out both inside and outside the MOOC platform?*
- Section 5.2, 5.3, and 5.4: *RQ 2. (Technical question) Do NLP-driven methods identify accurate problem-solving information?*
    1. Section 5.2 elaborates the process of data cleaning and the impact of NLP techniques on the accuracy of text clustering. It addresses the question *"Can they maintain high accuracy with extremely noisy, large-scale text data?"*
    2. Section 5.3 and 5.4 present the problem-solving methods extracted from the text data. They address the question *"Can topic modeling accurately discover the problem-solving methods?"*
- Section 5.5: *RQ 3. (Educational question) What are the educational implications of identified problem-solving methods?*

1. Section 5.5 discusses the interpretation of findings using the educational theories, such as task-structure and active/passive learning frameworks.

Lastly, Section 5.6 discusses important discussion points and limitations of the research.

5.1 Experimental Setup

The entire data of 7,482 survey responses from 2021 is used to run the optimized NLP pipeline. Results from Chapter 5 served as a foundation for configuring the NLP pipeline. In total, about 16 NLP libraries and models are used to build the pipeline. The hyperparameter tuning of models is also conducted during the optimization process in Chapter 5, which are already reflected in the experiment setup. You can see the specification of optimal NLP configuration in Table 5.1 below.

Phases	Components	Library / Model	Func / Param
Preprocessing	Sentence tokenization	nltk.tokenize	sent_tokenize()
	Spelling correction	symspellpy	SymSpell()
	Stopwords removal	gensim.utils	simple_preprocess()
		nltk.corpus	stopwords.words()
	Contraction expansion	nltk.tokenize.treebank	TreebankWordDetokenizer()
	Lemmatization	nltk.stem	WordNetLemmatizer()
	Collocation detection	gensim.models	Phrases()
	POS filtering	SpaCy	spacy.load('en')
	Key phrase extraction	keybert	KeyBERT()
Topic Modeling	Sentence embedding	Sentence transformers	all-MiniLM-L12-v1
	Normalization	torch.nn	BatchNorm1d()
	Dimension reduction	umap	UMAP()
	Clustering algorithm	sklearn.cluster	DBSCAN()
	Topic creation	sklearn.feature_extraction	CountVectorizer()
Text summarization	Summarization	transformers	PegasusTokenizer
			PegasusForConditionalGeneration

Table 5.1: Types of libraries and models used to perform NLP tasks in this research. Each of the setting is optimized based on the performance test results described in Chapter 4

5.2 Preprocessing

Fig. 5-1 shows the preprocessing steps and the number of remaining responses after each step. In total, 7,482 responses were fed into the preprocessing pipeline. As a result, merely 1,644 responses were selected; approximately 78% of data is discarded from the original dataset, which are considered poorly structured or irrelevant to work with. Fig. 5-1 below shows the preprocessing steps and the number of remaining responses.

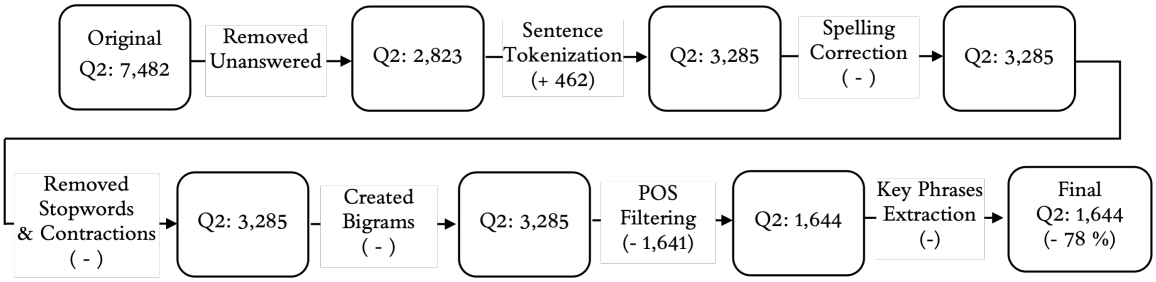


Figure 5-1: Flowchart of the preprocessing steps and the number of remaining responses

5.3 Topic Modeling

Table 5.2 shows the result of topic modeling in "Freq." and "Keywords" columns. The model generated 18 topics from 1,644 responses, with one topic showing irrelevant information; ID-16 {possible, relax, feeling}. The result indicates that ID-01 {code, step, try} is the most frequently mentioned problem-solving approach by the students, followed by ID-02 {video, look, lecture} and ID-03 {answer, check, question}.

5.4 Text Summarization

Table 5.2 shows the summaries of topics generated by the text summarization model in the "Generated summaries" column. Generated summaries give more contextual information than the keywords, supplementing the limited readability of the keyword outputs of topic modeling.

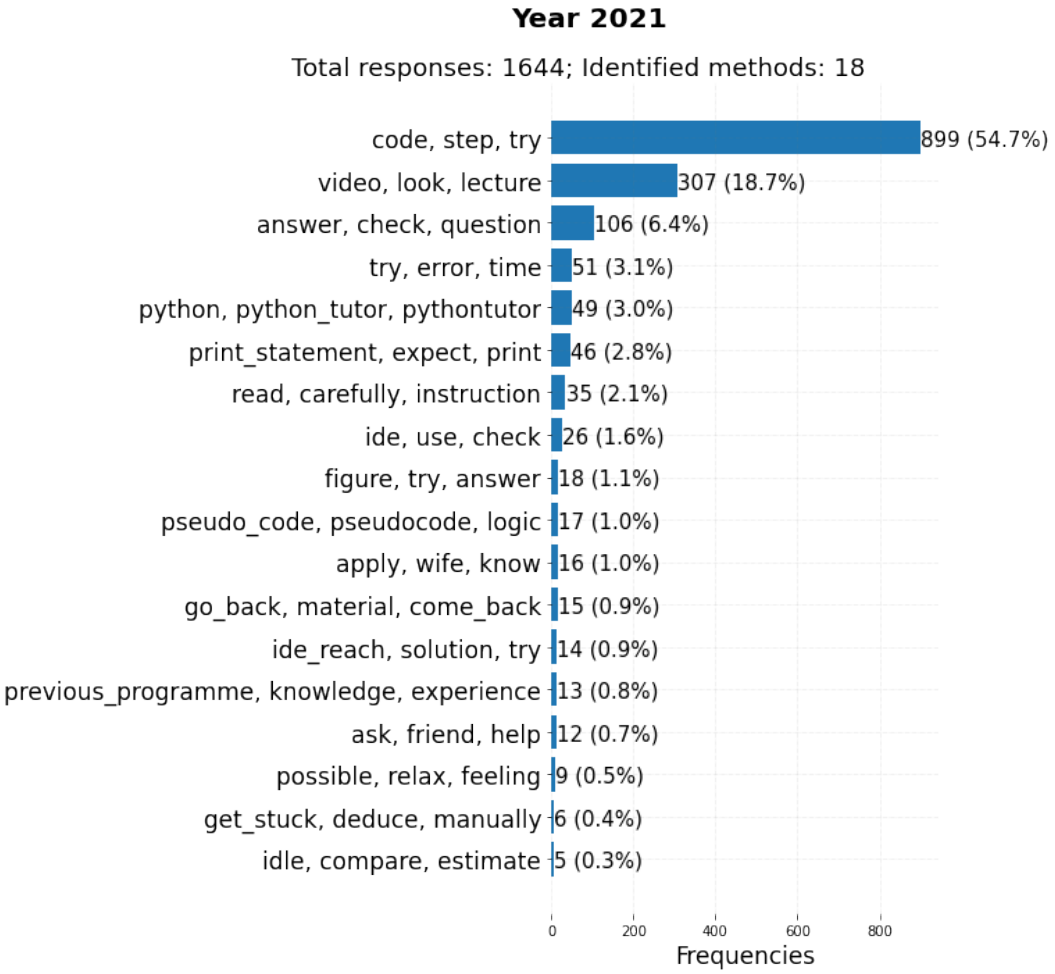


Figure 5-2: Topic Modeling result of 1,644 responses

5.5 Educational Implications

5.5.1 Task-structure Framework

Five tasks are formulated based on the identified 18 problem-solving approaches, which are *Knowledge construction, Social/peer learning, Understanding the problem, Clarifying code logic,* and *Coding and debugging*. One approach below is excluded from the task-structure due to its irrelevance.

- ID-16: I am moving this week across the country so I need to be calm and methodical

5.5.2 Active/Passive Learning Framework

We created a matrix to categorize the problem-solving approaches in the two-dimensional space; Task-structure vs. Active/Passive Learning. We placed the approaches in the matrix

ID	Freq.	Keywords	Generated summaries	Examples (highest c-TF-IDF)
01	899	code, step, try	I try to step through the problem by writing the code on the price of paper.	i write the code with pen and paper and try to step through the problem.
02	307	video, look, lecture	Use the knowledge from the video to answer the question.	make good note from the video and look at discussion thread.
03	106	answer, check, question	If I get the question wrong, I will use my ide and the discussion thread to correct it.	for true false question i answer with the inverse of my wrong answer after identify why i be incorrect.
04	51	try, error, time	I try to understand the error by using try and error method.	i try to understand the error for example with google stack overflow.
05	49	python, python_tutor, pythontutor	I use python tutor to help me understand my code.	in that case i run the code with python tutor and read the discussion to understand the try except else finally flow.
06	46	print_ statement, expect, print	I use auxiliary print statement to check if the program works as I expect.	mainly i use auxiliary print statement to check if the different iteration work as i expect.
07	35	read, carefully, instruction	Carefully read the instruction.	read the instruction very carefully.
08	26	ide, use, check	I would use the ide shell if I didn't know an answer.	if i do not know an answer i would try different thing use the ide shell.
09	18	figure, try, answer	To figure out a graceful solution to the cube root video, use stack overflow.	use stack overflow to figure out a graceful solution to the cube root video cliffhanger negative fraction.
10	17	pseudo_code, pseudocode, logic	Prepare pseudocode to understand the logic flow and then code.	pseudocode and reading on blog to see the thinking that be involve.
11	16	apply, wife, know	I use what I know about programming and trial and error to fine tune my solution.	i mostly apply what i know about programming and then fine tune the result through trial and error.
12	15	go_back, material, come_back	I would go back to the lecture and try to find a solution to the problem.	i would go back through the example problem give in the handout in the lecture and try to adopt that code for these situation.
13	14	ide_reach, solution, try	I try on ide until I reach the solution.	try on ide until reach the solution.
14	13	previous_ programme, knowledge, experience	I have used previous programming knowledge.	by use previous knowledge skill from another programming language.
15	12	ask, friend, help	I have to ask my friend for help.	i have to ask for help from my friend at these exercise.
16	9	possible, relax, feeling	I am moving this week across the country so I need to be calm and methodical.	i be move this week across country so honestly it be just panic and desperation.
17	6	get_stuck, deduce, manually	If I am stuck and can't see any way out, I check the resource online.	first i try to deduce with what i know and if i be stuck really hard and cannot see any way out i check the resource online.
18	5	idle, compare, estimate	If it doesn't work, I try it in my idle and compare it to what I'm suppose to get.	i do it in my idle and if it do not work i debug it and compare it with be correct.

Table 5.2: Results of topic modeling and text summarization. Column (1) ID is used as an identifier in Table 5.3, (2) frequency of topic, (3) topic keywords from topic modeling, (4) generated summaries by the summarization model, and (5) examples of input sentences to the summarization model

	Knowledge construction	Social/peer learning	Understanding the problem	Clarifying code logic	Coding and debugging
Active Learning	• Check online resources (ID-17)	• Ask a friend (ID-15) • Use discussion forum (ID-03)	• Write down steps (ID-01)	• Use python tutor (ID-05) • Write pseudo code (ID-10)	• Use print statement (ID-06) • Trial and error (ID-11) • Try on IDE (ID-8, ID-13, ID-18) • Use stack overflow (ID-09)
Passive Learning	• Watch lecture videos (ID-02) • Look lecture material (ID-12) • Use previous knowledge (ID-14)		• Read instructions (ID-07)		• Understand error (ID-04)

Table 5.3: Categorization of methods by Active/Passive Learning (row) and Subtasks (column). ID is the identifier that helps locate the topic in Table 5.2. Underlines indicate the methods that took place outside the platform.

according to their corresponding tasks and activeness of learning. Table 5.3 shows how approaches are placed across the matrix. In *Knowledge construction*, we can see that students engage more with the passive learning methods, focusing on consuming the learning resources from the course or relying on the previous knowledge. Contrarily, in *Coding & Debugging*, students engage more with the active learning methods. It seems like a natural outcome as the coding and debugging requires students to actually write codes and fix errors. Students also use Python tutor and pseudo-code to *Clarify code logic*. To *Understand the problem*, students write down the steps of problem and carefully read the instructions. Lastly, students also try to *Learn from their Peers* by asking friends and using discussion forum.

5.6 Discussion and Limitation

While this methodology successfully identified students' problem-solving methods from the large-scale text data, there are some important discussion points and limitations to discuss:

5.6.1 Preprocessing

(1) Improved Accuracy vs. Data Loss

Preprocessing plays a significant role in improving the quality of topic modeling; we observed accuracy improvements in all clustering algorithms, except for OPTICS. The best performing algorithm, DBSCAN, improved its accuracy from 63% to 78%. However, we also had to suffer approx. 78% of data being discarded during the preprocessing process. This can play as a constraint for data of small size with a high level of noise.

(2) Multi-methods Problem Solving Approach

Another point to discuss is how we can capture students who using multiple problem-solving methods. This research tokenizes all the responses by sentences to avoid conflicting topics in the same data entry. However, the shortcoming of this approach is that it can only capture one topic per data entry (single sentence). This approach cannot detect how many methods are used by a single students.

5.6.2 Topic Modeling

(1) Enhanced Performance through Transformer

The transformer-based topic modeling model significantly outperforms the "bag-of-words" models, such as LDA and GSDMM. The topic modeling performed most accurately with the combination of BERT sentence-embedding representations and DBSCAN clustering algorithm, which achieved the highest clustering accuracy of 78% compared to 16% by LDA and 24% by GSDMM.

(2) Considering FP and FN in the Performance Measurement

In this research, we only used F1 Score to measure the accuracy of clustering as shown in Table 4.4. It is because we thought the impact of FP (False Positive) and FN (False Negative) to the topic modeling result is not significant. However, in the future research, we can explore using FP and FN to evaluate the clustering result. It can provide us with

insights on what types of similar documents are assigned to different clusters (FN) and, vice versa, dissimilar documents are assigned to the same cluster (FP).

5.6.3 Text Summarization

(1) Improved Readability and Importance of Pre-trained Model Selection

Integrating the text summarization model into the topic modeling pipeline provided a novel solution to improving the limited readability of keyword-based results of topic modeling. However, as shown in Table 4.5, the quality of summaries highly depends on the pre-trained model you chose to use. If you are not fine-tuning the model with your data, it is highly recommended that you test different models before selecting one.

5.6.4 Educational Implications

(1) Identified Tasks through Task-structure Framework

Based on the formulated task-structure, we learned that students focus on five learning tasks to solve programming problems: (1) Knowledge construction, (2) Social/peer learning, (3) Understanding the problem, (4) Clarifying code logic, and (5) Coding & debugging. Many problem-solving approaches focused on tackling *Coding & debugging*. This shows that the selection of problem-solving approach is highly task-dependent (programming) and demonstrates the "tool" importance as the problem-solving is very reliant on using the tools.

(2) Active/Passive Learning by Tasks

The Active/Passive Learning framework also provided insights on the types of learning student use to tackle certain tasks. For example, *Knowledge construction* mainly relied on passive learning, such as *watching lecture videos, look lecture material,* etc. Contrarily, *Coding & debugging* relied on active learning, such as *Try on IDE, Trial and error, Use print statement,* etc. A previous study suggests that there is a strong correlation between doing and learning outcome, called "doer effect" [31]. With the method we proposed, we

can further analyze whether "doer effect" exist in the learning behaviors carried out outside the platform, such as *use stack overflow, use pythontutor,* and so on.

(3) MOOC's Possible Constraints for Some Methods

Another thing to notice is the empty or less-populated cells in Table 5.3. It can be an indication of learning constraints in the MOOC environment, which hinder students from using the methods in those cells. For example, as an Active Learning method for *Knowledge Construction*, students can carry out a group project to apply knowledge in different contexts to earn a new one. However, due to the nature of MOOC, such coordination is difficult for instructors to accommodate.

5.7 Use Case Diagrams

This section describes an overview of how the proposed NLP system fits into the existing MOOCs learning environment and interacts with the stakeholders involved in the operation of MOOCs.

5.7.1 Boundary and Architecture of the System

Object Process Methodology (OPM) is a conceptual modeling methodology for designing systems. We used this system design method to describe and specify the functions, structures, and values offered by the system. Fig. 5-3 shows the OPM of MOOCs and proposed NLP system's architecture. The system's function, what it does, consists of an *operand* and a *process*. The *process* is an action or transformation taken on the *operand* to create values.

The system has five operands.

- External operands: *Students* and *Enhanced LXD (Learning Experience Design)*
- Internal operands: *User data, Survey responses,* and *Insights*

These operands are transformed by *Value Processes*, which changes the states of operands to create the system values. For example, the operand of *Students* is transformed into

Learning Experience as it goes through the process of *Learning*, and again *Learning Experience* is transformed into *Survey Responses* through the process of *Surveying*. This process repeats until it reaches the ultimate goal of *Enhanced LXD*.

For each of processes, there is a single or multiple *Value Instruments* that enables the process. For example, the process of *Learning* is enabled by *Course material* and *Learning features*. Likewise, Fig. 5-3 shows the relationships of other *Value Processes* and *Value Instruments*. The last two layers (from the right) of *Supporting Processes* and *Supporting Instruments* work towards supporting *Value Instruments*.

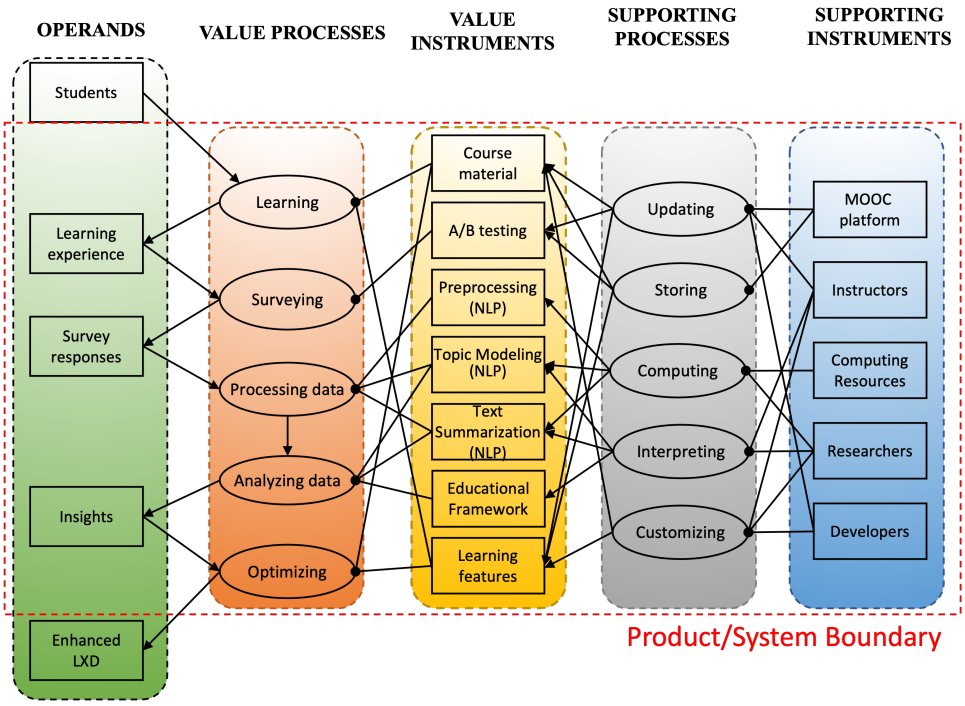


Figure 5-3: OPM of MOOCs and NLP system's architecture

5.7.2 Stakeholder Needs and Relationships

Here, we used a stakeholder map to illustrate the relevant stakeholders and their exchanges of mutual needs. We identified four stakeholders involved in the operation of MOOCs and NLP system:

1. **Students**: Students are anyone enrolled in MOOCs to use the learning content and features.

2. **Instructors**: Instructors deliver courses and facilitate the learning of students based on the instructional design developed by educators.

3. **Educators**: Educators are responsible for contributing to students' learning by creating learning modules and developing instructional designs, while an instructor is the one who teaches; a teacher. Often educators act as an instructor as well.

4. **Admins**: Admins provide administrative and technical supports to instructors and students. Their job includes the customization of online survey questionnaires. For some courses, this task is done by the instructors.

The priority of needs is indicated by three types of arrows: (1) *Must meet*, (2) *Should meet*, and (3) *Might meet*. Please refer to the *Needs priority* box located in the upper-right side of Fig. 5-4.

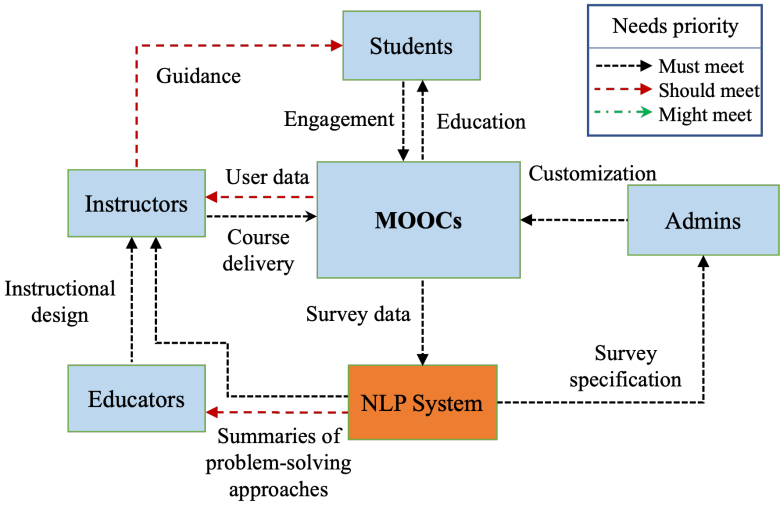


Figure 5-4: Stakeholder map for MOOCs and suggested NLP system with characterization of the needs illustrated

There are several important needs in the loop that are key to our research. First of all, the loop among MOOCs, NLP System, and Admins serves as a foundation of this research, which enables data collection, data processing, and customization of survey based on the data needs. Therefore, their needs must be met, hence, indicated as *Must meet*. The needs in the loop among MOOCs, NLP System, and Instructors are also essential and must be met to achieve the objective of this research. It enables Instructors to better understand

the students' problem-solving approaches and reflect that in their course delivery to improve the learning quality. The instructional design must be provided from Educators to Instructors for them to deliver courses accordingly, therefore, the need priority is indicated as *Might meet*. The summaries of problem-solving approaches from the NLP system can help Educators to enhance the instructional design, but not necessarily have to happen during the course as that could cause disruption and confusion to students. It could happen after the course only if it is considered necessary. Given that MOOCs are delivered with or without live instructors, the needs of user data for Instructors and guidance for Students are indicated as *Should meet*, necessary but not essential. Lastly, but perhaps most importantly, the mutual needs between Students and MOOCs (Engagement and Education) must be met to have the fundamental ground to initiate this research.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In the present work, we suggested a NLP-driven methodology around utilizing topic modeling and text summarization, which successfully captured the students' problem-solving methods carried out in the MOOC's learning setting. By applying an effective set of preprocessing techniques, we were able to secure a high-quality data with a cost of approximately 78% loss of data; in return of 15% increase in the DBSCAN's clustering accuracy. The transformer-based clustering method using BERT's sentence-level embedding and DBSCAN clustering algorithm contributed significantly to improving the topic modeling's performance; using this approach the clustering accuracy increased to a mean accuracy of 75% from LDA's 16%. Furthermore, we integrated the text summarization model into the final step of pipeline to transform the keyword-based results into well-structured narrative summaries, which contributed to improving the readability of the final outputs.

As much as we focused on the computer science aspect of the research, we wanted to stretch the realm of this research to the education domain as well. Two existing educational frameworks were used to draw educational implications from the findings. The task-structure framework provided a guidance in formulating types of tasks that students were tackling, inferring from the identified problem-solving methods. We observed that students heavily rely on "tools", such as IDE, Pythontutor, etc., for solving programming problems and can expect that such selection of methods can vary noticeably depending on

the given task.

The current approach allows the capability to process large-scale text data to gain rich conclusions of students' behaviors while studying on the MOOC platform. This information can provide a good guidance to MOOCs instructors for improving their instructional designs and teaching strategies. Hopefully this research can open new opportunities for future research. Below is a recap of key conclusions of this research:

1. Preprocessing played a significant role in improving the quality of topic modeling; DBSCAN's clustering accuracy improved from 63% to 78% (Table 4.4)

2. Topic modeling performed most accurately with the combination of BERT sentence-level embedding and DBSCAN clustering algorithm, achieving a mean accuracy of 75% compared to 16% by LDA (Figure 4-5)

3. Integrating text summarization with topic modeling suggested a solution to improving the poor readability of keyword-based output of topic modeling by generating narrative summaries (Table 5.2)

4. Based on the formulated task-structure, we learn that students focus on five tasks to solve programming problems: (1) Knowledge Construction, (2) Social/peer learning, (3) Understanding the problem, (4) Clarification of code logic, and (5) Coding debugging (Table 5.3)

5. Based on the active/passive learning framework, among the identified 18 problem-solving methods, 12 are active learning methods, 5 are passive ones, and one outlier. Furthermore, for *Knowledge Construction*, students mainly relied on the passive learning methods, whereas for *Clarifying code logic* and *Coding and debugging*, they more relied on active learning methods (See Table 5.3)

6.2 Future Work

6.2.1 Addressing Research Limitations

There are areas to improve in this research. Firstly, in terms of computer science domain, the current preprocessing pipeline is highly selective considering that only 22% of the data is preserved after the preprocessing. Although it is understandable as user-generated data are often very noisy, it would be interesting to further delve into what are the characteristics of data that are causing such a huge data loss and explore other preprocessing techniques that can reduce the data loss while maintaining a high-level of data quality. Furthermore, examining whether different survey questionnaire design can help minimize the noise in the responses can be an interesting area to look into.

Secondly, in terms of educational domain, although two educational frameworks used in this research provided useful information and an interesting guidance to the interpretation of findings, we can explore more available educational frameworks that can potentially help us draw interesting insights from the discovered problem-solving methods. In particular, it would be interesting to find an educational framework that can help improve the instructional design of online courses based on the students' problem-solving strategies.

6.2.2 Building on the Findings of Research

We can use the NLP method proposed in this research to find answers to other interesting questions in online education. For example, we can examine whether the students' problem-solving methods are time-sensitive, differ every year, whether students have different strategies for solving problems of different course subjects, whether there is a demographic pattern in the use of methods, and so on. Below I elaborated some of potential areas for the future research.

(1) Year-over-Year Analysis

Yearly observation of students' problem-solving methods can common or time-specific methods used over the years can be an interesting research topic. For example, *watching*

video lectures and *try on IDE* are some of popularly used methods in both 2020 and 2021, please see Section 4.3.3. We can observe for a longer time period to see how consistently they appear through the years and what are the newly adopted methods among the students. Furthermore, considering that the pandemic had a significant impact on the people's lives, looking into pre and post pandemic periods can provide an insight on whether the pandemic has altered the types of problem-solving methods students use.

(2) Comparison with other courses

Observing how students' learning behaviors and methods vary across different course subjects can provide interesting insights to educators. We learned from this research that students rely on tool-based methods to solve programming problems. But students may adopt different problem-solving strategies for problems of different course subjects, e.g. art, liniguistics, etc.

(3) Academic Performance

The correlation between the level of knowledge and the student's choice of problem-solving strategies can be an interesting area to explore. For example, students proficient in coding might focus more on methods like coding and debugging, whereas students of beginner's level might focus more on the construction of knowledge. We can use students' performance information, such as grades, participation level, etc., to conduct the correlations analysis.

(4) Demographic Analysis

The learning engagement and use of problem-solving methods may differ based on the students' demographic information. Depending on the countries or the communities they are part of, students may experience connectivity or social barriers for learning. These factors could influence the types of tools or methods students use to solve problems. For example, students with limited connectivity may prefer text-based or document-based learning methods, which are less data-consuming, over methods like watching videos. It would be

interesting to research on this topic to see whether there are demographic patterns in the problem-solving methods.

Appendix A

List of Generated Bigrams

Table A.1: Examples of created bigrams

Bigram Words
use_ide, sketch_step, lecture_video, videos_previously, read_slide, google_help, keep_trying, read_book, python_tutor, trial_error, step_step, discussion_point

Appendix B

Survey Analysis: Multiple-choice Questions

B.1 Background

An online survey was conducted on students enrolled in a MOOC for the spring term of 2021. Survey participants were asked about their learning experiences, considering difficulties, motivation levels, and self-reflections. The questions were consisted of 5 free-text questions and 6 multiple-choice questions.

- **Course name:** Introduction to Computer Science and Programming Using Python
- **Survey location:** The survey was conducted during the period of Week 1, 2 and 4. It was placed at the end of each section

B.1.1 Survey questions

The survey questions were consisted of 5 free-text questions and 6 multiple-choice questions. Survey participants were asked mainly about their learning experiences, particularly related to the course difficulties and the self-reflections. The list of questions asked in the survey is shown in Table B.2.

Table B.1: Time and locations of survey conducted on students in a MOOC. The course has total 9 weeks of units.

Time	Unit	Section
Week 1	Unit 1: Python Basics	1. Introduction to Python
		2. Core Elements of Programs
		Problem Set 1
	Unit 2: Simple Program	3. Simple Algorithms
		4. Functions
		Problem Set 2
Week 2	Unit 3: Structured Types	5. Tuples and Lists
		6. Dictionaries
		Problem Set 3
	Unit 4: Good Programming Practices	7. Testing and Debugging
		8. Exceptions
		Problem Set 4
Week 4	Unit 5: Object Oriented Programming	9. Classes and Inheritance
		10. An Extended Example
		Problem Set 5
	Unit 6: Algorithmic Complexity	11. Computational Complexity
		12. Searching and Sorting Algorithms
		Problem Set 6

B.2 Multiple-choice Questions Analysis

B.2.1 Responses - Rating

Fig. B-1 shows the area charts of Q3, Q4, Q7, and Q8. In each question, students are asked to answer the question in the scale of *Extremely, Very, Moderately, Slightly,* and *Not at all*. These questions were asked from Unit 1 to Unit 6, so we can observe the changing trend over the time.

According to the Q3 graph, the students thought that Unit 2 and 5 were particularly challenging as indicated by the peaks in both Unit 2 and 5. These peaks correlate with the drops in the Q4 graph which asks about how prepared the students feel for the exercises. However, the increase in the difficulty level does not seem to have a significant impact on the students' motivation levels and perception of usefulness. As shown in the graphs of Q7 and Q8, both graphs maintain steady trends throughout the units, although there are slight increases in the negative responses ('Slightly' and 'Not at all') in Unit 5 and 6.

Table B.2: Multiple-choice questions in the survey

No.	Questions	Format
Q3	How challenging were these exercises?	MCQ
Q4	How prepared did you feel for these exercises?	MCQ
Q5	What previous/external resources were helpful for these exercises? Select all that apply	MCQ
Q7	How motivated are you to continue this course?	MCQ
Q8	Do you find these exercises useful?	MCQ
Q10	Please choose 2 or 3 values that are most important to you	MCQ

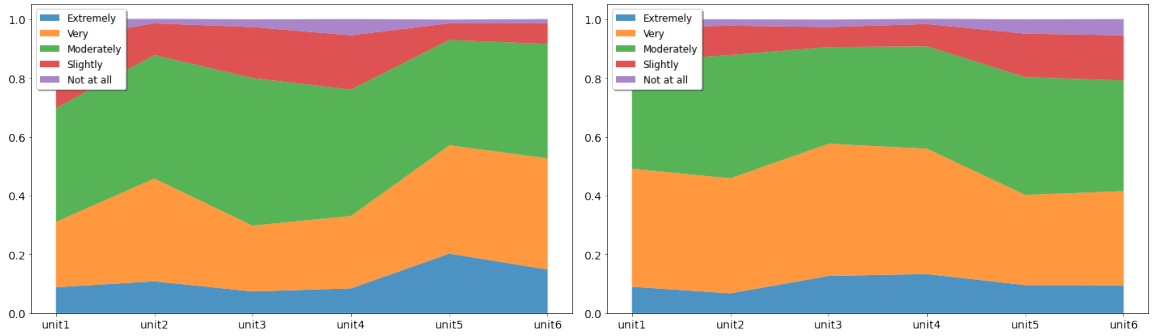
B.2.2 Responses - Multi-selection

As shown in Fig. B-5, students selected the video lectures to be the most helpful learning resources among the given options. It leads the second place by a huge margin, which is followed by previous programming knowledge / skills, discussion threads, and so on.

Here, we used a Bump Chart to explore the changes in Rank of a value over a time dimension (Unit 1 Unit 6). The overall rank maintains the same throughout the progress of the course, except there was a single switch between the last two resource types, 'discussion thread from outside this course' and 'exercises from outside this course'.

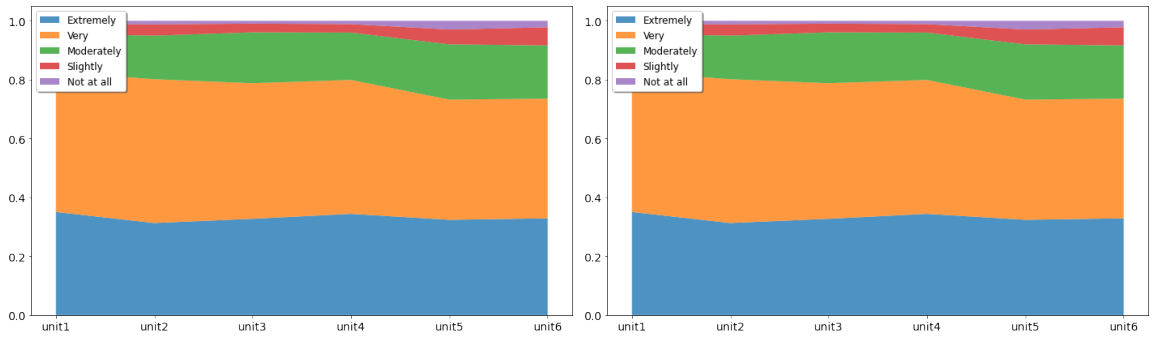
'Personal and intellectual growth', 'Graining broad skills and knowledge', and 'Relationships with family or friends' are among the most valued items by the students, followed by 'Health and well-being' and 'Compassion and kindness'. Contrarily, 'Religion and spirituality' and 'Athletics and sports' are among the least valued items by the students.

The importance of values did not change substantially during the progress of the course, except for minor switches of order that occurred within the top-tier and low-tier groups.



(a) Q3. How challenging were these exercises?

(b) Q4. How prepared did you feel for these exercises?



(c) Q7. How motivated are you to continue this course?

(d) Q8. Do you find these exercises useful?

Figure B-1: Area graphs of multiple-choice questions

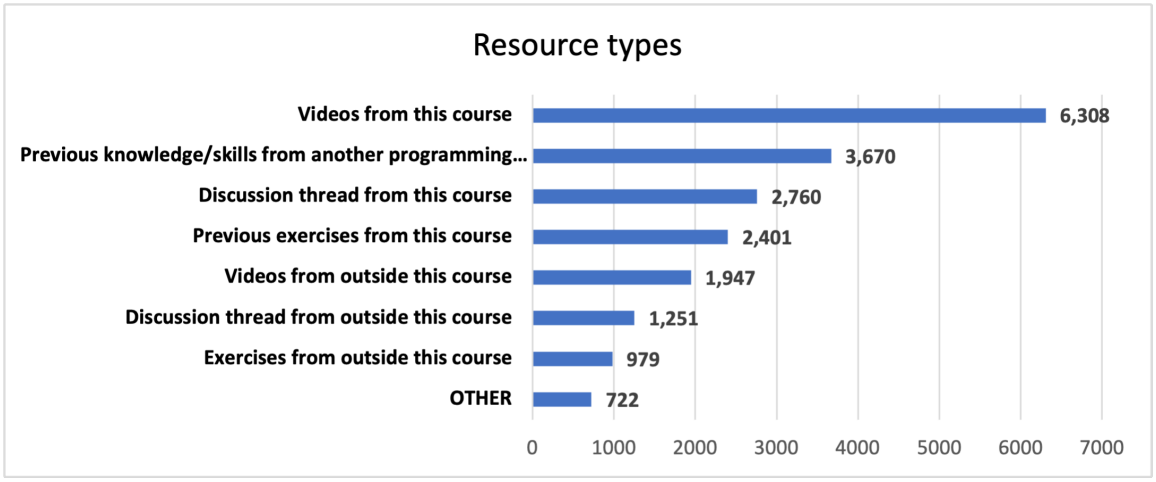


Figure B-2: Q5. What previous/external resources were helpful for these exercises? Select all that apply

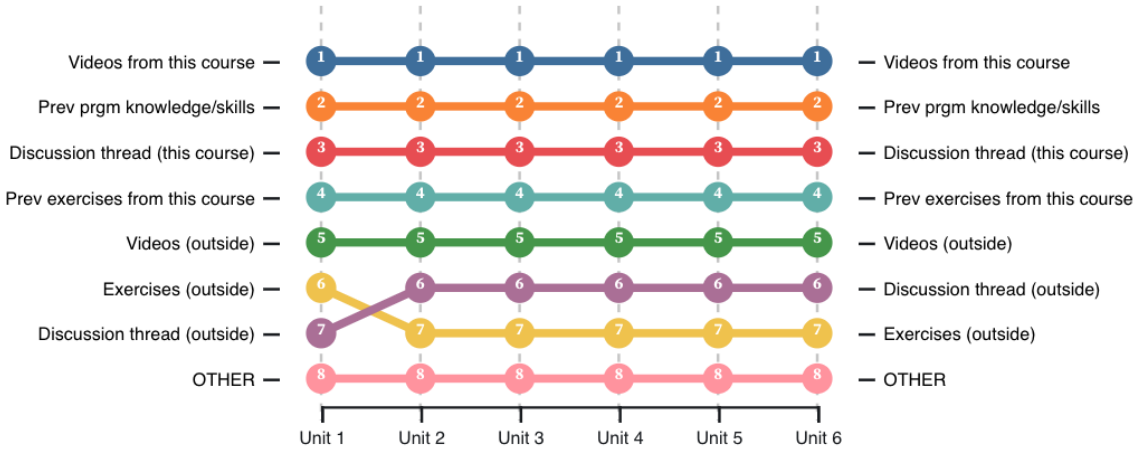


Figure B-3: Bump chart showing the rank of Helpful Resources by Unit

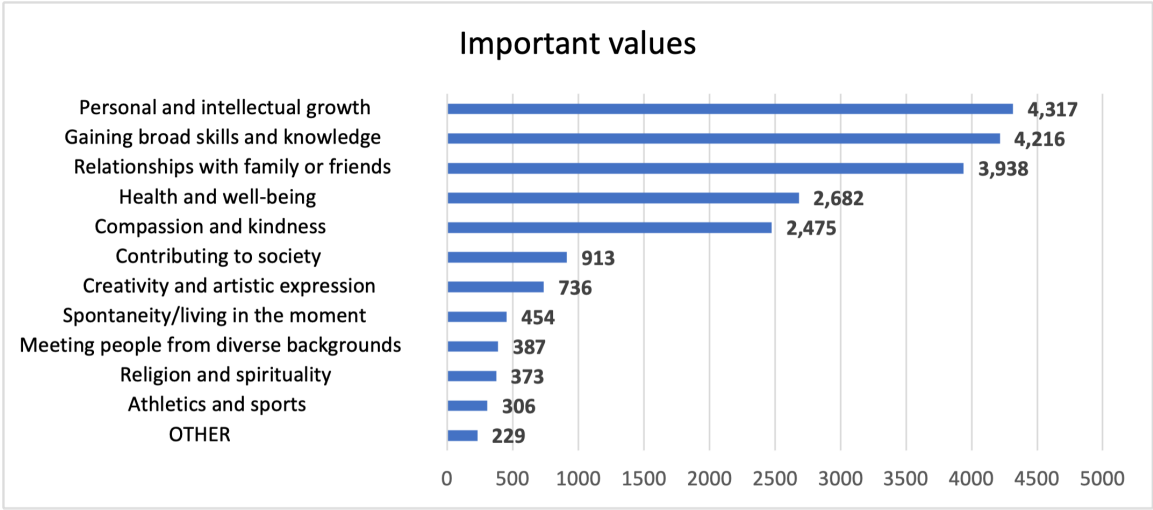


Figure B-4: Q10. Please choose 2 or 3 values that are most important to you (there is no right answer)

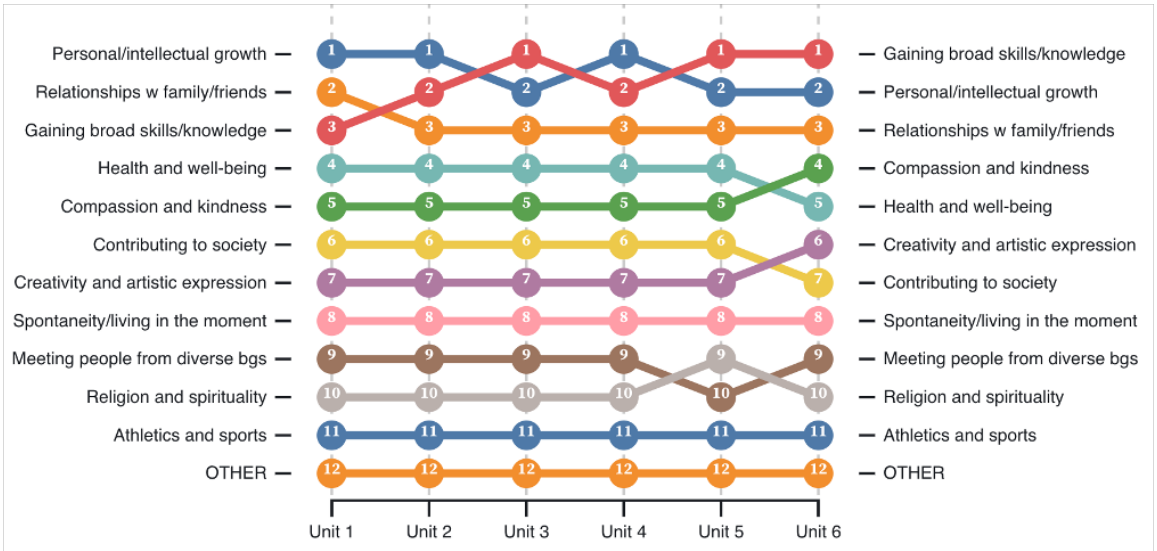


Figure B-5: Bump chart showing the rank of Important Values by Unit

Bibliography

[1] Ashwaq Al-Musharraf and Mona Alkhattabi. An educational data mining approach to explore the effect of using interactive supporting features in an lms for overall performance within an online learning environment. *International Journal of Computer Science and Network Security*, 16(3):1–2.

[2] Valentina Alto. Understanding pointwise mutual information in nlp. *medium*, 2020.

[3] Amina Amara, Mohamed Ali Hadj Taieb, and Mohamed Ben Aouicha. Multilingual topic modeling for tracking covid-19 trends based on facebook data analysis. *Springer Science+Business Media*, 2021.

[4] Enrique Amigó, Julio Gonzalo, Javier Artiles, and M. Felisa Verdejo. Comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(4):14–16, 2009.

[5] Ercan Atagün, Bengisu Hartoka, and Ahmet Albayrak. Topic modeling using lda and bert techniques: Teknofest example. In *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pages 660–664, 2021.

[6] David Paul Ausubel. The psychology of meaningful verbal learning. *New York, Grune Stratton*, 1963.

[7] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[8] Fernanda Cesar Bonafini. The effects of participants' engagement with videos and forums in a mooc for teachers' professional development. *Open Praxis*, vol.9(issue.4):p.433–447, 2017.

[9] J. Bonwell, C.; Eison. Active learning: Creating excitement in the classroom. *EHE-ERIC Higher Education Report*, A(1), 1991.

[10] B. Chandrasekaran. Task-structures, knowledge acquisition and learning. *Machine Learning*, 4:p.339–345, 1989.

[11] B. Chandrasekaran. Design problem solving: A task analysis. *AI Magazine Volume*, (Number 4), 1990.

[12] Xieling Chen, Gary Cheng, Haoran Xie, Guanliang Chen, and Di Zou. Understanding MOOC reviews: Text mining using structural topic model. *Human-Centric Intelligent Systems*, vol.1(Issue 3-4):p.55–65, 2021.

[13] Matthieu Cisel. Interactions in MOOCs: The hidden part of the iceberg. *International Review of Research in Open and Distributed Learning*, vol. 19(5), 2018.

[14] Wikipedia contributors. List of english contractions, 2020. Retrieved July 30, 2020 from https://en.wikipedia.org/wiki/Wikipedia:List_of_English_contractions.

[15] Pratap Dangeti. *Statisticss for Machine Learning*. O'Reilly, 2017. https://www.oreilly.com/library/view/statistics-for-machine/9781788295758/.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[17] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[18] Center for Instructional Technology and University of Florida. Training. Adopting active learning approaches.

[19] Galeopsi. Keyword extraction with bert. *medium*, 2020.

[20] Wolf Garbe. Spelling correction fuzzy search: 1 million times faster through symmetric delete spelling correction algorithm.

[21] Anna Glazkova. Identifying topics of scientific articles with bert-based approaches and topic modeling. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Trends and Applications in Knowledge Discovery and Data Mining:pp 98–105.

[22] Maarten Grootendorst. c-tf-idf. 2020.

[23] Maarten Grootendorst. Keybert, 2021. https://github.com/MaartenGr/KeyBERT.

[24] Philip J. Guo, Juho Kim, and Rob Rubin. How video production affects student engagement: An empirical study of mooc videos. *Conference: Proceedings of the first ACM conference on Learning @ scale conference*, 2014.

[25] Elisa Grande Elena Colomina Clara hamizo Gonzalez, Julian Cano. Educational data mining for improving learning outcomes in teaching accounting within higher education. *International Journal of Information and Learning Technology*, 32:272–285.

[26] Robert Han, Feifei Ellis. Predicting students' academic performance by their online learning patterns in a blended course: To what extent is a theory-driven approach and a data-driven approach consistent? *Educational Technology Society*, 24(1):191–204.

[27] Khe Foon Hew. Promoting engagement in online courses: What strategies can we learn from three highly rated moocs. *British Journal of Educational Technology*, vol.47(no.2):p.320–341, 2016.

[28] Roya Hosseini, Peter Brusilovsky, Michael Yudelson, and Arto Hellas. Stereotype modeling for problem-solving performance predictions in moocs and traditional courses. *Conference: User Modeling Adaptation and PersonalizationAt: Bratislava, Slovakia*, 2017.

[29] Sonia Pamplona Javier Bravo-Agapito, Sonia J. Romero. Early prediction of undergraduate student's academic performance in completely online learning: A five-year study. *Computers in Human Behavior*, 115, 2021.

[30] Aviva Aiden Adrian Veres Matthew Gray Joseph Pickett Dale Hoiberg Dan Clancy Peter Norvig Jon Orwant Steven Pinker Martin Nowak Jean-Baptiste Michel, Yuan Shen and Erez Aiden. *Quantitative Analysis of Culture Using Millions of Digitized Books*. Science (New York, N.Y.), 331 (01 2011), 176–82. edition, 2011. https://doi.org/10.1126/science.1199644.

[31] Ana Bell Erik Hemberg Jitesh Maiyuran1, Ayesha Bajwa1 and Una-May O'Reilly. How student background and topic impact the doer effect in computational thinking moocs. 2019.

[32] Youngjin Lee. Using self-organizing map and clustering to investigate problemsolving patterns in the massive open online course: An exploratory study. *Journal of Educational Computing Research*, vol.57(2):p.471–490, 2019.

[33] James Melville Leland McInnes, John Healy. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. arXiv, 1802.03426v3 edition, 2020. https://arxiv.org/pdf/1802.03426.pdf.

[34] Chin-Yew Lin. *ROUGE: A Package for Automatic Evaluation of Summaries*. ACL, post-conference workshop of acl 2004 edition, 2004. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/07/was2004.pdf.

[35] Sannyuya Liu, Xian Peng, Hercy N. H. Cheng, Zhi Liu, Jianwen Sun, and Chongyang Yang. Unfolding sentimental and behavioral tendencies of learners' concerned topics from course reviews in a mooc. *Journal of Educational Computing Research*, vol.57(Issue 3):p.670–696, 2018.

[36] Lara J. A. Romero C. López-Zambrano, J. Improving the portability of predicting students' performance models by using ontologies. *Advance online publication*, pages 1–19.

[37] Cater III J.J. Varela O. Michel, N. Active versus passive teaching styles: an empirical study of student outcomes. *Human Resource Development Quarterly*, 20(4):397–418.

[38] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *Curran Associates Inc.*, 2013.

[39] Vladimir Molchanov and Lars Linsen. Overcoming the curse of dimensionality when clustering multivariate volume data. *Scitepress*, page 1, 2018.

[40] Jasmine Paul and Felicia Jefferson. A comparative analysis of student performance in an online vs. face-to-face environmental science course from 2009 to 2016. *Frontiers in Computer Science*, vol. 1, 2019.

[41] Xian Peng and Qinmei Xu. Investigating learners' behaviors and discourse content in mooc course reviews. *ELSEVIER Computers Education*, vol.143, 2020.

[42] Zhang G. Sheng X. et al. Qiu, F. Predicting students' performance in e-learning using learning process and behaviour data. *Sci Rep*, 12(453), 2022.

[43] Radhika Santhanam, Sharath Sasidharan, and Jane Webster. Using self-regulatory learning to enhance e-learning-based information technology training. *Information Systems Research*, vol.19(no.1):p.26–47, 2008.

[44] Dhawal Shah. By the numbers: Moocs in 2021. *Class Central*, 2021.

[45] Alexander Shashkov, Robert Gold, Erik Hemberg, ByeongJo Kong, Ana Bell, and Una-May O'Reilly. Analyzing student reflection sentiments and problem-solving procedures in moocs. *L@S '21: Proceedings of the Eighth ACM Conference on Learning @ Scale*, pages p.247–250, 2021.

[46] Stephen Silverman, Raj Subramaniam, and Amelia Mays Woods. Task structures, student practice, and skill in physical education. *The Journal of Educational Research*, vol.91:p.298–307, 1998.

[47] Eleni Stroulia and Ashok K. Goel. Task structures: What to learn? *AAAI Technical Report*, 4:p.339–345, 1994.

[48] Vicki Trowler. Student engagement literature review. *The higher education academy*, pages p.1–15, 2010.

[49] George Veletsianos, Amy Collier, and Emily Schneider. Digging deeper into learners' experiences in MOOCs: Participation in social networks outside of MOOCs, notetaking and contexts surrounding content consumption. *British journal of educational technology*, Vol. 46(3):p.570–587, 2015.

[50] Jovita M. Vytasek, Alyssa Friend Wise, and Sonya Woloshen. Topic models to support instructors in MOOC forums. *Conference: the Seventh International Learning Analytics Knowledge Conference*, page p.610–611, 2017.

[51] Miaomiao Wen, Diyi Yang, and C.P. Rosé. Sentiment analysis in MOOC discussion forums: What does it tell us? *Proceedings of the 7th International Conference on Educational Data Mining*, EDM:p.1–8, 2014.

[52] Wikipedia. Dbscan. 2022.

[53] Xiao Yang-cai and Wang Rui. A study of mooc course review topics mining based on lda topic model. *3rd Africa-Asia Dialogue Network International Conference on Advances in Business Management and Electronic Commerce Research*, 2021.

[54] Jianhua Yin and Jianyong Wang. A dirichlet multinomial mixture model-based approach for short text clustering. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014.

[55] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization, 2020.