Analyzing MOOC Reviews: A Comparative Study of Learner Feedback and Sentiment

by

Israt Zahan Era
ID: 20301442

Fiana Nilhat Faruquee
ID: 20101236

Mohammad Showrab Hossan
ID: 20301081

Sumaiya Ali
ID: 20301405

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
June 2025

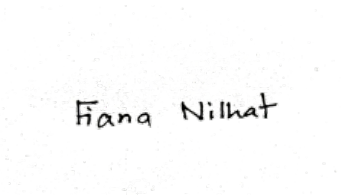
Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Israt Zahan Era
ID: 20301442



Fiana Nilhat Faruquee
ID: 20101236

Mohammad Showrab Hossan
ID: 20301081

Sumaiya Ali
ID: 20301405

Approval

The thesis/project titled "Analyzing MOOC Reviews: A Comparative Study of Learner Feedback and Sentiment" submitted by

1. Israt Zahan Era (ID: 20301442)
2. Fiana Nilhat Faruquee (ID: 20101236)
3. Mohammad Showrab Hossan (ID: 20301081)
4. Sumaiya Ali (ID: 20301405)

Of Spring, 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on June 20, 2025.

Examining Committee:

Supervisor:
(Member)

Najeefa Nikhat Choudhury
Senior Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Program Coordinator and Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

MOOCs (Massive Open Online Courses) offer broad and diverse access to education, but summarizing learner feedback remains difficult because of the large volume and wide-ranging content of the reviews. Extracting meaningful insights from learner feedback saves the time of the learner and helps to make a quick decision about the course. This study focuses on analyzing and summarizing MOOC reviews across multiple courses to understand learner impressions towards the course employing advanced Natural Language Processing (NLP) models including BART, PEGASUS, T5 and DISTILBART with an aim of generating brief yet coherent summaries. Since MOOC reivews lacked human-written summaries, our experiment demonstrate that, even in the absence of supervised data from MOOCs, our method greatly enhances summary quality. Finally, our evaluation framework incorporated semantic similarity, coherence, and human evaluation. T5 achieved the highest semantic similarity score (0.60), while PEGASUS demonstrated the best coherence (0.40). In human evaluations, PEGASUS received the highest ratings in fluency (mean score: 4.75) and maintained strong performance across relevance (4.35) and factual accuracy (4.40). T5 closely followed with the highest relevance (4.40) and factual accuracy (4.45) scores. However, Cohen's Kappa scores revealed low inter-rater agreement, with most values indicating slight or even negative agreement—highlighting subjectivity and inconsistency among raters. Ultimately, this study intended to help prospective learners quickly perceive the overall sentiment and key takeaways of a course that saves their time and effort of reading through thousands of individual comments.

Keywords:
MOOC reviews, summarization, natural language processing (NLP), BART, PEGASUS, T5, DistilBart, semantic similarity, coherence, human judgment, Cohen's Kappa, relevance, fluency, factual accuracy, inter-rater agreement, educational insights, online learning experiences, Word Cloud, Transformer, Encoder, Decoder, fine-tuning.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

MOOC platforms have opened a new horizon for the students and teachers in recent years. The concept referred to as "MOOC" was first introduced in 2008 by David Cormier and Bryan Alexander to describe the course 'Connectivism and Connective Knowledge' [1]. It has brought about a revolution in terms of equal learning opportunity. Coursera, Udacity and edX are three major online platforms that offer courses without regard to geographical barriers. No matter whether a learner lives on a different continent or in the depths of poverty itself, anyone with internet access can learn and improve themselves through the educational experience it offers. It is now extensively applied in higher education, but both the learners and instructors are facing some challenges. As MOOCs continue to grow in popularity, they generate a huge amount of feedback. This can take the form of reviews, message board posts, wrap-up quizzes or even simple clicking on an icon for 'evaluation'. These feedbacks are very essential for understanding the vital details (course strengths, areas of improvement, and specific aspects like teaching quality, assignment effectiveness, etc.) of a course. Learners often feel uncertain about which course to enrol in or whether the course they've selected meets their quality expectations. By reviewing feedback from past students, they can quickly determine if a course aligns with their needs. Since MOOCs are intended to provide educational support to as many learners as possible, the amount of feedback is huge. Reviewing all feedback and identifying key insights is a challenging task for both learners and instructors. Through the use of Natural language Processing, educators and course developers will be able to view data points which lead to the development of better courses with better learner satisfaction and improved completion rates for courses. In addition, it is necessary to summarize such responses for drawing informed conclusions so that important trends and problems may be understood with support from a large volume of data. Even to this date, there is always a negative correlation between the level of summarization and the loss of important information; this is because of the many NLP models that have been developed for summarization. This work tries to fill this gap by describing new paradigms for keeping meaningful information in compact summaries so that a limited loss of understanding occurs for educators and learners to make effective decisions in online learning. Text summarization involves reducing the size of the main text by preserving its key information and meaning. Since manually summarizing text is often time-consuming and challenging, automating this

process is becoming increasingly popular and is a key driver of academic research [2]. This technique has broadly applied in various domains. In this study, we will explore text summarization of learners' feedback collected from coursera.

1.2 Problem Statement

With MOOCs, today it is easy to gain quality education, but this type of learning still has critical challenges on how it will deal with learners' disengagement and dissatisfaction. Involving learners in course development is advantageous, as they provide substantial feedback through reviews, discussion forums, and ratings. To enhance learner satisfaction, it is essential to understand their needs and sentiments. While ratings provide a quantitative measure of feedback, textual reviews offer deeper insights into learners' experiences, concerns, and expectations. Additionally, analyzing MOOC reviews using sentiment analysis may provide a general understanding of learners' satisfaction but opinion summarization can provide a deeper insight into learners' demand and sentiment. In the MOOC domain, much of the work has been done to analyze MOOC reviews using sentiment analysis but summarizing the opinion is very underexplored. However, manually analyzing thousands of reviews across multiple courses is impractical. The problem lies in making appropriate sense of this voluminous and unstructured data for a system. Traditional methods of summarizing cannot minimize information loss and maximize brevity at the same time. In general, large sets of reference summaries are needed for supervised training. Otherwise, the generated summaries often involve a trade-off between the level of compression and the amount of useful information retained. Moreover, they generate summaries that may suffer from poor readability and lack coherence. But the resources, especially in the MOOC domain, are unavailable for fine-tuning the models. To overcome this limitation, we fine-tuned pre-trained models on a related domain that included human-written summaries. In this case, we used Amazon product reviews and their summaries to train models and generate course-wise summaries from unstructured MOOC feedback. To generate appropriate summaries from the complex and diverse languages of educational reviews, the key problem was for the models to adapt the language style from product review summaries. Though the language pattern and style are similar in the sense that they are both human-written judgments, it was nevertheless difficult to retain crucial information in course review summaries. Therefore, our goal is to provide concise, informative summaries for each course by addressing all these identified issues that can assist both prospective learners in making informed enrollment decisions and educators in identifying areas for course improvement. Furthermore, we did an assessment comparing the traditional models to examine their adaptability when they were fine-tuned on a different domain.

1.3 Research Objective

Thus, the objective of the research reported in this paper is to design the method for summarizing the Massive Open Online Course reviews that is as concise as possible but informative at the same time. This includes:

1. **Evaluating Summarization Models**: To evaluate the performance of summarization models (e.g., BART, PEGASUS, T5, and DISTILBART) in generating high-quality summaries, assessed through semantic similarity, coherence, and human evaluation.

2. **Optimizing Summary Quality**: To retain essential information, ensure coherence and relevance in summaries, explore methods involving domain-specific fine-tuning and generate high-quality summaries.

3. **Comparative Analysis of Models Adaptability**: Conduct a comparative analysis of the adaptability of various pre-trained models in generating MOOC review summaries after training on a different domain.

4. **Improving MOOC Effectiveness**: To gain actionable insights and identify strengths and address weaknesses in learners' experiences this work will offer course developers and learners an informative summary.

The paper will hence attain these goals of helping the research in educational technology by demonstrating how the cutting-edge summarization techniques with accessible assessment methods transform learners' feedback. This not only supports MOOC improvement but also enables prospective students to make quick decisions when selecting courses.

Chapter 2

Literature Review

Understanding and analyzing the overall sentiment of customers through summarization is always a great challenge because of its unstructured nature. Moreover, when it is about student feedback, most of the researchers followed the sentiment analysis approach to understand learners' satisfaction and dissatisfaction by classifying the opinion as positive or negative. However, sentiment analysis alone does not provide comprehensive insights into the specific aspects of learner experiences. In this section we reviewed some papers that are intended to analyze MOOC reviews through sentiment analysis. Later we explored some research about opinion summarization and text summarization approaches.

2.1 Related Work

2.1.1 Sentiment analysis on MOOC reviews

Sentiment analysis has become a popular approach to analyzing learners' attitudes and experiences about online education, especially in a context where Massive Open Online Courses (MOOCs) occur. There are several studies carried out using sentiment analysis to analyze learners' opinions from reviews of MOOCs. This literature review will discuss different forms of sentiment analysis in online education.

One interesting investigation by Wen et al. [3] used a collective sentiment analysis method to explore the relationship between students' opinions and dropout rates in a MOOC. Interestingly, they found a significant correlation between the two – the more negative sentiment there was in the forums, the higher the dropout rate tended to be. This suggests that paying attention to student sentiment could be a valuable way to predict and potentially prevent dropouts in MOOCs. Wang [4] took a somewhat different approach, using semantic analysis in an attempt to predict students' graduation probability based on their emotional tendencies. They started by using the NRC Affect Intensity Lexicon to assign intensity scores to different words. Marion et al. [5] in their study, investigated students' perceptions of their experiences during a cloud computing course. They used VADER to produce a lexicon based sentiment score. The result they found from this study is sentiment analysis can measure the emotions of students efficiently, and emotions vary from different student populations, particularly based on gender. Karsten et al. [6] also used the VADER algorithm in their study, but they applied it specifically to partic-

ipant comments in a MOOC designed to teach beginners how to code. They wanted to see if there was a relationship between the overall sentiment expressed in the comments and the reviews that learners left about the MOOC. Manuel et al. [7]in their study, developed a system for evaluating academic institutions by analyzing aspect-level sentiment of online reviews. The system achieved a sentiment analysis accuracy of 72.56. The study aims to analyze MOOC reviews to address biased ratings and improve course selection through sentiment analysis and topic modeling. Using VADER (highest Pearson correlation: 0.49) and LDA, it identified themes like course quality and content relevance. Results showed 63% of ratings were 5-star, highlighting bias, while NLP models provided actionable insights for better course characterization. Shaik et al. [8] give an in-depth review of methods for analyzing emotions in the university setting. The methods reviewed included NB, LR, SVM, Decision Trees, Random Forest, CNN and LSTM. They also reviewed the LDA and LSI approaches, which can also be helpful in aspect-based sentiment analysis, as they help reveal which aspects of the course (that students comment on positively or negatively) are considered. Gottipati et al. [9] also used a combination of topic modeling and sentiment analysis techniques to analyze learner reviews and improve teaching and learning in higher education. Specifically, to classify the sentiment expressed in the reviews, they used TextBlob and LDA for unsupervised extraction of topics. The study [10] aims to analyze MOOC learner perceptions using sentiment analysis and deep learning, focusing on factors influencing satisfaction. Using RCNN for topic classification and four sentiment lexicons (syuzhet, bing, afinn, nrc) for sentiment scoring, it identified seven key factors, including course quality, instructor, and assessment, with humanities-related MOOCs receiving higher satisfaction than STEM courses.

2.1.2 Opinion Summarization

Opinion summarization refers to a process of generating summaries that reflect the subjective opinions expressed across multiple texts, such as product reviews [11]. Given the high dropout rates associated with MOOCs, it becomes relevant to identify what factors influence learners' engagement and satisfaction. By analyzing the opinions expressed in comments, reviews, and forum posts left by learners, researchers can more deeply understand the feelings of students regarding their learning experience and which areas need further attention. The goal of this paper [12] is to improve MOOCs by analyzing the summary of discussion forums. They focused on the extraction of the suggestions for the course improvement. The techniques they used were VADER to detect the sentiment towards specific courses, rule-based techniques focused on detecting modal verbs (e.g., "should", "could") and wishful phrases (e.g., "hope", "recommend"), and the summarized opinions and suggestions are displayed through intuitive visual aids (e.g., pie and bar charts), helping instructors quickly identify areas needing improvement. The VADER method successfully recalled around 65% of sentiment classes. The suggestion mining method identified 50% of posts containing actionable suggestions. This paper [13] focused on generating concise summaries from vast datasets, such as customer reviews or feedback. They used extractive summarization methods. For the unsupervised learning, the algorithms used include K-Means, MiniBatchK-Means, and Graph-based summarization. Graph-based summarization showed the

best performance, improving precision, recall, and F-measure compared to other methods like K-Means and MiniBatchKMeans. The researchers in this paper [14] describe a fully automated process of movie review summarization by natural language processing. It proposes a graph-based ranking algorithm. According to the results of this experiment, the recommended approach outperformed the existing methods (LexRank and TextRank) on the main evaluation measures. For the proposed method, the average precision was 0.48485, recall was 0.47925, and F-measure was 0.482. With LexRank, the average scores were 0.39215, 0.3997 and 0.3959 for precision, recall and F-measure respectively. Similarly, TextRank achieved lower values of 0.24515, 0.25535, 0.25015, respectively. Another work [15] investigates whether multi-document summarization models are able to synthesize 8 conflicting inputs into accurate summaries. The models, including T5, PRIMERA, and GPT-4, were evaluated on movie reviews and biomedical systematic reviews. The results indicated that GPT-4 outperformed others with a high sentiment correlation, Pearson's r = 0.900, in synthesizing movie reviews. This study [16] investigates the automatic detection of suggestions in opinionated texts, for example, feedback and posts on social platforms, by classifying sentences as suggestions or non-suggestions. LSTM demonstrated the highest performance, with F1 scores of 0.727 on the suggestion forum and 0.672 on the electronics reviews, whereas CNN showed slightly lower results. Arthur et al. in their research [11] proposed an unsupervised opinion summarization model named Copycat, which outperformed the summarization model like LexRank. It achieved a Rogue score on Amazon reviews (R1,0.2947 , R2 ,0.0526 , RL,0.1809) and (R1 R2 RL ,0.3197 0.0581 0.2016) on Yelp review. Similarly, using the Amazon and Yelp dataset they generate opinion summaries in another study [17]. They proposed another novel framework named fewSum, which outperforms copycat. Target-orientated opinion summarization aims to extract user opinions on specific targets or fine-grained sub-aspects with minimal supervision, employing phrase-based summarization to enhance coherence and informativeness. FineSum, a novel framework in this domain, clusters fine-grained opinions under specific aspects using aspect and sentiment classifiers [18]. Somnath et al. in their study [19] introduce a semantic autoencoder for unsupervised opinion summarization in extractive approach. The model is designed to pick key information from multiple reviews to generate concise summaries. It does this without needing human-written reference summaries. Stefanos et al. [20] developed a weakly supervised opinion summarization system that extracts aspects and their sentiments from product reviews without requiring huge amounts of supervised training data. The authors in this study [21] presented an opinion summarization framework named OpinionDigest which doesn't require any gold standard summary for training. They used ABSA models to extract review phrases and then applied trained transformer-based summarization models to generate the opinion summary. They achieved a rogue score for R1, R2 and RL, respectively (29.30, 5.77, 18.56). Instead of considering only the text data, the researchers in this study [22] consider the image data also. They used separate encoders for text and image input data. Finally they decode them as text and generate an opinion summary. They achieved an FBERT score for the Amazon and Yelp datasets, respectively, 87.7 and 87.9. To address issues like being too generic or lacking supporting details in opinion summarization the authors in this study [23] introduce a new paradigm of summarizing reviews. The system they proposed is an unsupervised extractive system named RATION. The

system consists of two components: an opinion extractor and a rationale extractor. They got an accuracy result of 0.88% and 0.84% for non-redundancy and coherence. Another unsupervised extractive opinion summarization method named TokenCluster was introduced by Haoyuan et al. in this study [24]. Their findings indicated that TokenCluster enhanced aspect coverage in multi-opinion summarization tasks, benefiting both aspect-based and general summarization approaches.

2.1.3 Text Summarization Methods

An application of the T5 transformer-based model was explored for automated text summarization in this research [25]. The model implements abstractive summarization, leveraging the transformer architecture's encoder-decoder framework and multi-head attention mechanisms. The evaluation metrics ROUGE achieved high precision (e.g., ROUGE-1: 0.88) but relatively lower recall (e.g., ROUGE-1: 0.58). The study underscores the efficiency of transformer-based models in producing clear and concise summaries, helping to manage the problem of information overload in natural language processing. AlArfaj et al. [26] evaluates the effectiveness of automatic summarization methods for simplifying texts for readers with varied literacy levels, focusing on Brazilian Portuguese. Methods like EPC-P, SuPor-2, TextRank, and GistSumm were tested using precision and ROUGE scores. SuPor-2 achieved the highest informativeness (ROUGE: 0.5839), while EPC-P excelled in identifying key sentences (89% precision). In a similar approach this study [27] evaluates the impact of automatic text summarization methods on simplifying texts for readers with limited literacy skills. It tested several extractive summarizers, including EPC-P, GistSumm, SuPor-2, and TextRank, using precision and ROUGE metrics. SuPor-2 achieved the best results in informativeness (ROUGE: 0.5839), while EPC-P excelled in identifying key sentences (89% precision). Another study [28] developed an automatic text summarization tool using the TF-IDF algorithm, and compared its performance with two online summarizers, Tools 4 Noobs and Text Summarization. The TF-IDF-based summarizer achieved the highest average F-measure (0.666), outperforming both Tools 4 Noobs (0.622) and Text Summarization (0.629) in most experiments, demonstrating superior accuracy in matching human-generated. S. Aubakirov et al. [29] propose an improvement in the GreedSum algorithm, a technique for extractive text summarization. The research investigates whether adjusting the minimum document frequency parameter for individual document clusters can lead to better summary quality, measured by ROUGE scores. In this paper [30] , the authors introduce EXABSUM, an innovative framework for automatic text summarization that incorporates a hybrid approach to generate summaries. This study explores using BART as a way to generate abstractive summaries of patient enquiries in the medical context [31]. The conclusion is drawn that a pre-trained BART model was fine-tuned on patient-doctor dialogue data, and the summaries can improve the effectiveness of summarizing medical texts. The results have been highly encouraging, surpassing over 92% of existing studies in this area. The proposed approach was evaluated using the "MedDialogue" dataset, with outcomes compared against prior research to demonstrate its effectiveness in medical summarization tasks. This paper [32] addresses challenges brought about by the plethora of textual data available on the internet, which, although abundant, makes it hard to find relevant information. Conventionally, comprehensions were manually made

by subject experts from lengthy texts, which was time-consuming and inefficient. It advocates for an intelligent "abstractive" text summarization system beyond the generic extractive methods.

This literature review identifies the progression from simple sentiment analysis to sophisticated opinion and summary generation for MOOCs. Although early research was limited to binary sentiment classifications, most recent research studies emphasize fine-grained opinion-level detail and a focus on summarization capabilities with models based on transformers for coherent and informative summaries. These advances provide a foundation for a new paradigm of feedback summarization systems aimed at online education systems.

Chapter 3

Methodology

3.1 Dataset Collection

3.1.1 Overview

The dataset that was used for generating summaries was collected from Kaggle, which is known as a popular platform for data science competitions and datasets. Additionally, we collected learners' feedback from Coursera, a well-known MOOC platform and merged it with the dataset from Kaggle. Since the dataset was unlabelled, which means that there were no reference summaries available for training, the Amasum dataset, which contains Amazon product reviews and summaries, was used to fine-tune abstractive models from the transformer family, namely BART, T5, PEGASUS, and DistilBART. We obtained the Amasum dataset from Papers with Code.

3.1.2 Dataset

The final dataset consist of 18 courses from Coursera with their textual reviews. The dataset contains 17,417 unique reviews and six features including reviews, ratings, reviewers, date , course_id and course_title. This dataset serve as a rich foundation for the analysis of learner preference identification of trends and actionable insights for course creators in order to further enhance their course offerings. Meanwhile, the Amasum dataset is organized into separate folders for training, validation, and testing. It contains 25,203 training samples, 3,114 validation samples, and 3,166 test samples. Each folder stores data in JSON format, where each JSON file consists of multiple user reviews for a specific product along with their corresponding summaries.

3.2 Dataset Preprocessing

The dataset was cleaned to prepare it for processing and to ensure its integrity and suitability for generating summaries using summarization models. Additionally, this process ensured that data remained intact. The steps are outlined as follows:

Key cleaning steps:

- **Dealing with Duplicate Data:** To avoid redundancy, ensuring that the dataset contains unique records, duplicate values in the dataset were identified and removed.
- **Remove null values:** Checked and removed the null rows based on the review column.
- **Normalize Multiple Punctuation Marks:** Standardized redundant punctuation (e.g., `!!!` becomes `!`).
- **Remove Hyperlinks:** Stripped URLs and hyperlinks to avoid irrelevant content.
- **Filter Non-English Words:** It kept only valid English words in order to consistency.
- **Remove Extra Spaces:** It condensed space in order for clean formatting.

3.3 Dataset Visualization

To analyze the presence of positive, negative and neutral feedbacks in the dataset we visualize the feedback patterns in terms of sentiment score and ratings. These visualizations provide an insights into the sentiment patterns of learners' feedback for further analysis.

3.3.1 Feature-Wise Visualizations

1. **Rating Distribution**
    - Based on the learners ratings to visualize the frequency of each rating class we plotted a bar graph.
    - Each rating value was considered a separate class and the corresponding frequency was represented as the number of reviews.
    - **Insights:** It gives us a clear vision of general trends, rating imbalances, and the pattern of student satisfaction and provides a thorough understanding of the rating distribution.
2. **Sentiment Distribution**
    - A bar graph and a pie chart were plotted to visualize the frequency of positive, negative, and neutral reviews depending on their sentiment score.
    - Every sentiment label was treated as a separate category and the corresponding frequency was represented as the number of reviews in the graph.
    - On the other hand, each sentiment category is shown in the pie chart as a different colour.
    - **Insights:** It provides a general understanding regarding the learners' attitude towards the courses.

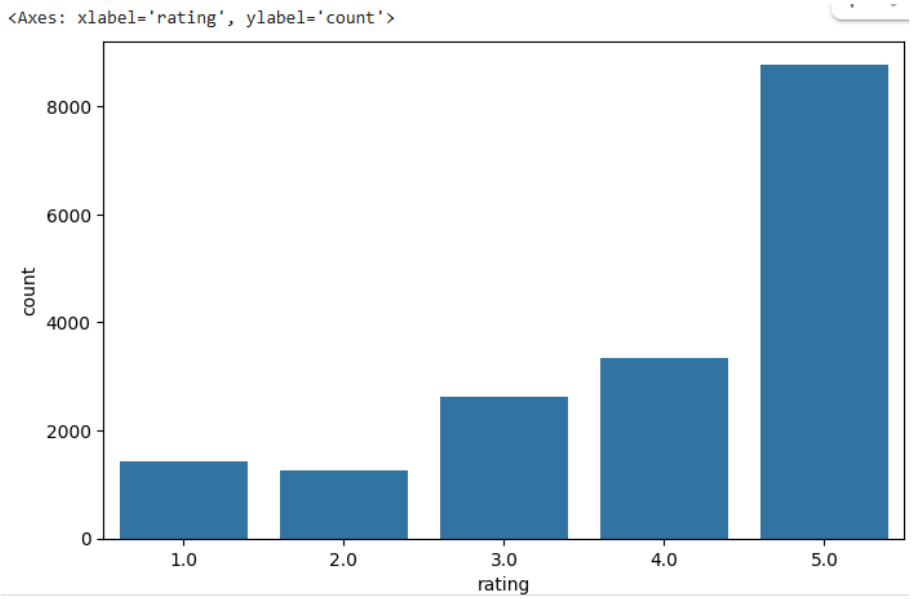


Figure 3.1: Rating Distribution of all Course Reviews

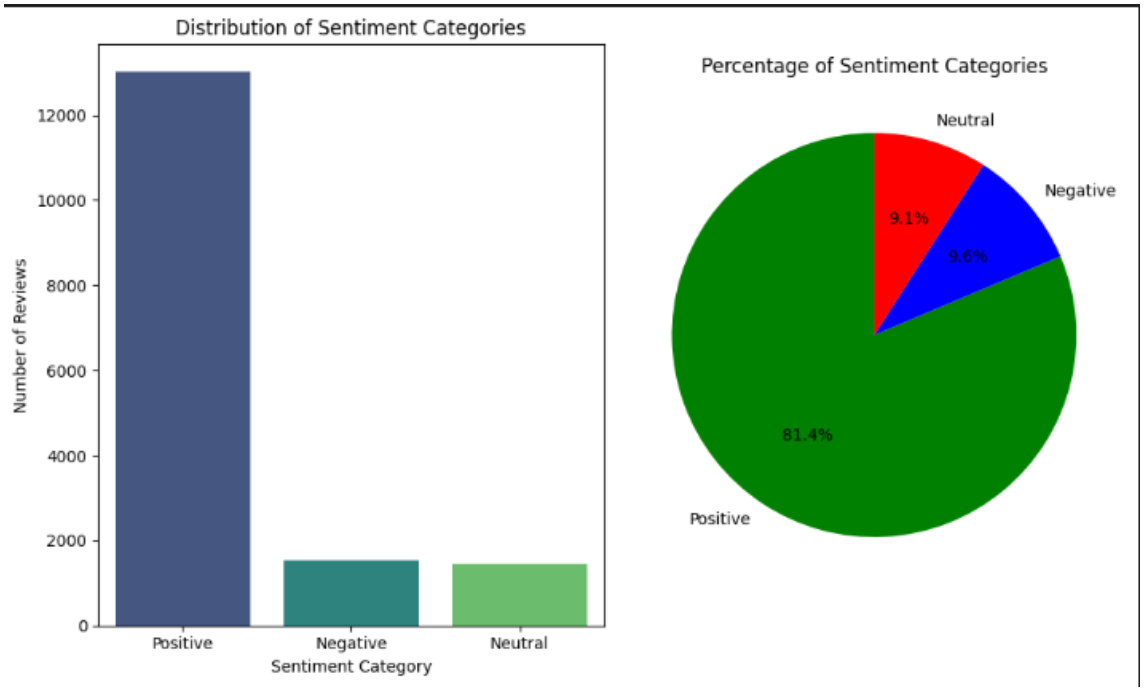


Figure 3.2: Sentiment Category Distribution

3.3.2 Frequency Analysis

For categorical and binned numerical columns, frequency plots were created to examine the frequency of each category or range. The visualizations above helped to understand the composition of the entire data set and gave insight into potential data imbalances.

3.3.3 Visualization Tools and Techniques

The visualizations were constructed using Python's `Matplotlib` and `Seaborn` libraries. Visualizations used appropriate titles, axis labels, and color schemes to aid clarity and interpretability. The `sns.countplot()` function counts the frequency of values in a specific column, grouped by categories, and **Matplotlib** is used to plot the resulting graph.

3.3.4 Calculation of Sentiment Score

To generate the sentiment score we used the python TextBlob library. The Textblob's sentiment polarity function outputs a float ranging from -1 to +1. Here -1 corresponds to extreme negative sentiment, 0 to a neutral tone, and +1 means highly positive sentiment. TextBlob is a sentiment analyzer that relies on a predefined lexical dictionary containing words with their corresponding sentiment scores. Lexicon contains positive, negative and neutral words with assigned values. TextBlob first tokenizes the text into words and phrases. Then it looks up sentiment scores in the lexicon for each word and adjusts scores based on negation (e.g., not ,never, no) and modifiers (e.g.,very, extremely, a little). For sentence level, once identifying the sentiment of each word TextBlob calculates the average polarity score across the entire sentence [33].

We passed every single comment to TextBlob to generate a sentiment score and created another column in our dataset which stores the sentiment score. This gave us a sentiment score of all distinct reviews. After that we grouped the dataset by course title and computed the mean sentiment score for each course using the `mean()` function.

```
from textblob import TextBlob

# Function to calculate sentiment polarity
def get_sentiment_score(text):
    blob = TextBlob(text)
    return blob.sentiment.polarity
```

Figure 3.3: TextBlob Function

3.3.5 Word Cloud of all Reviews

To highlight the terms used most frequently in the feedback a word cloud was created. The visual has highlighted several common aspects and keywords like 'Course', 'Assignment', 'Material', 'Content', 'Lecture', 'Video', 'Exercise', 'Time', 'Example', 'Practice', 'Instructor', 'Lab' etc. This word cloud was generated from combined reviews of all courses. This gives an intuitive summary of the dominant aspects in a pretty form that learners are most concerned about.

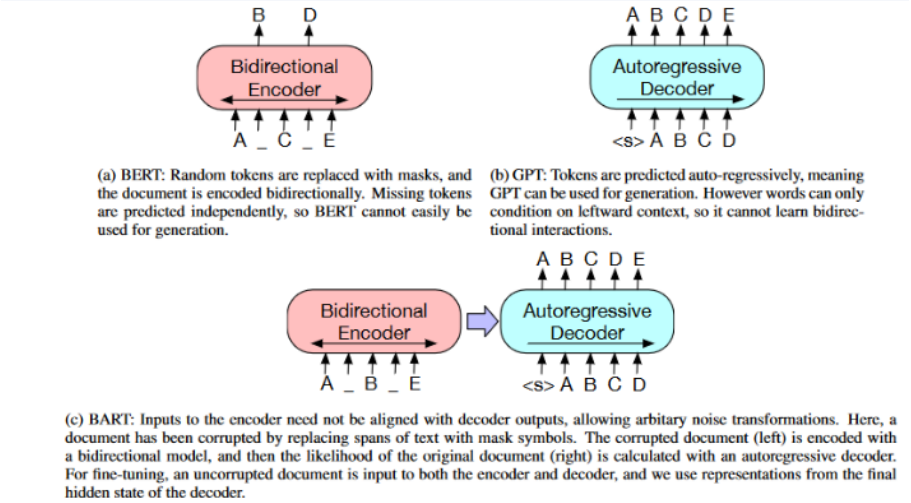
Figure 3.4: Word Cloud of Reviews

3.4 Summarization Models and Selection Rationale

3.4.1 BART (facebook/bart-large-cnn)

Model Overview: The Bart (Bidirectional and Auto-Regressive Transformers) is a transformation-based model. It has a bidirectional autoencoder which is involved in considering both the previous and next tokens. The encoder is inspired by BERT and in the pretraining stage it trained with denoising activity while, its decoder has an autoregressive nature and this is like the decoder of GPT. This model is explicitly developed by the Facebook AI for text generation task and because of its denoising property it can generate coherent, concise abstractive summaries.

Architecture: BART has been implemented with a sequence-to-sequence (seq2seq) architecture consisting of:
Encoder: It has a bidirectional transformer that is able to capture contextual information from both the previous and the next tokens, and it processes the texts as well.
Decoder: The decoder consists of an autoregressive transformer that generates each token one by one depending on the token generated previously and the output generated by the encoder.



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.

(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.

(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Figure 3.5: BART Architecture (with Comparison to BERT and GPT) [34]

Working Process: The working mechanism of BART's bidirectional encoder and autoregressive decoder is explained below:

- **Encoder:** The encoder has been designed as bidirectional so that it is able to process input text and capture the context of both the previous and next token. Bart achieved this bidirectional nature through several layers of self-attention mechanisms alongside feed-forward neural networks. At the pretraining stage its encoder is introduced with a corrupted input text, meaning some tokens or spans are masked, deleted or permuted. After that the encoder generate a comprehensive representation (internal vector embeddings) from this corrupted input sequence.

- **Decoder:** The decoder follows the autoregressive manner that means it generate the output as just one token at a time. In every step or sequence it generate a token based on encoder's embeddings and previously generated token. Until reach at the end token it continues like that. Similarly to the encoder, it comprises multi-head attention mechanisms. And finally, it generates coherent and appropriate contextual text [35].

Pre-training and Fine-tuning: BART proceeds through two training phases:

- **Pre-training:** During the pre-training phase BART goes through various noising strategies. The autoencoder gets an intentionally corrupted input text, like token masking, deletion and permutation. The decoder's function is to restoring the original text from this noisy data. This denoising activity makes BART strong to understand the language structure and semantics.

- **Fine-tuning:** After pre-training, BART goes through various labeled dataset to fine-tuned on certain tasks. Fine-tuning for summarization was conducted on the CNN/Daily Mail dataset that consisted of news articles with their associated summaries. BART was pre-trained on a diverse set of datasets, including Books Corpus, English Wikipedia, and other datasets used in the Common Crawl. [36]

Performance Metrics: This model achieved a very impressive Rouge scores on CNN/DailyMail dataset.

- **ROUGE-1:** 42.95
- **ROUGE-2:** 20.81
- **ROUGE-L:** 30.62

These metrics indicates the models effectiveness and how closely it can generate human-written summary.

Why Selected: Because of its pre-training strategies and internal architecture, it is very suitable for generating human-like, fluent summaries.

3.4.2 T5 (Text-to-Text Transfer Transformer - Base)

Model Overview: In our study we used the T5-base model. The Google research team developed this model. In the field of NLP it portrays a significant advancement because of its multitasking ability with one transformer. This model formulates all the NLP operations as a text-to-text problem, thereby allowing diverse tasks to share the same objective and model architecture. This approach allows this model to handle multiple tasks, like translation, summarizing, and question answering. The base model is an efficient version but effective in abstractive summarization.

Architecture: T5 incorporates transformer's encoder-decoder design for its architecture. The encoder of T5 converts the inputs into a sequence of contextualized embeddings after processes. Like the BART, the decoder of T5 also follows an autoregressive manner, meaning processes the tokens one by one.

- **Encoder:**
  - **Structure:** The structure of the encoder includes multiple layers each integrating a multi-head self-attention mechanism alongside a feed-forward neural network.
  - **Function:** It simultaneously handles the entire input sequence while contextually capturing the meaning from all the tokens.
- **Decoder:**
  - **Structure:** The decoder consists of multiple layers as well. Those layers contains several attention layers and a feedforward neural network.
  - **Function:** It generates the output tokens one by one taking care of both the prior generated token and the output from encoder.

Working Mechanism: The working mechanism of T5 includes several stages: tokenization, encoding, decoding, and output generation.

- **Input Tokenization:** Tokenization of input text is performed using a SentencePiece tokenizer. That means the text is segmented into subword units. As a result the model can easily handle the vocabulary efficiently and manage out-of-vocabulary words. The same tokenizer is used across the encoder as well as the decoder.

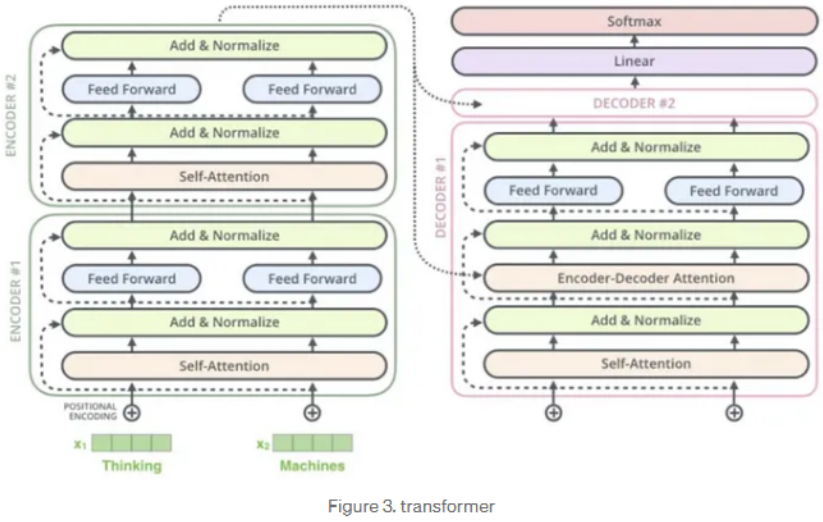


Figure 3.6: T5 Transformer Architecture
[37]

- **Encoding the Input:** Here, two crucial mechanisms take place: feedforward neural networks and self-attention. The self-attention mechanism basically weighs different words based on how they related to each other. After that, using that score it can easily understand which words are contextually more related. Next, the data are sent to a feedforward neural network to capture complex patterns in text using a non-linear transformations.

- **Decoding the Output:** In the output decoding section, the masked multi-head self-attention makes sure of the autoregressive property by predicting a particular position depending on only the earlier position from the output sequence, which is already known. Following that, encoder-decoder attention works with the encoder's output to ensure contextually relevant outputs. Finally, the feed-forward neural network works similarly to the encoder's one. Similarly, it captures necessary complex patterns using non-linear transformations for accurate text generation [37].

Pretraining and Fine-tuning:

- **Pretraining:** Like the BART model, T5 also has a denoising objective during its pretraining phase on a large corpus. But unlike BART here the corrupted version of text not only mask a single token; it masks a span. And the decoder is trained to regenerate the original text.

- **Fine-tuning:** Once finished with pretraining it has to going through fine-tuning for specific tasks with labelled data. But every task here is done by text-to-text generation and and task specific prefix is added. For example, for summarizing task the prefix would be ("summarize"). The model then understand which task it has to do and gives output based on that. [38]

Why selected: In our thesis, the T5 base model is utilized to abstractive summarization from learner feedback. The model is able to distill key information from extensive reviews and generate concise summaries.

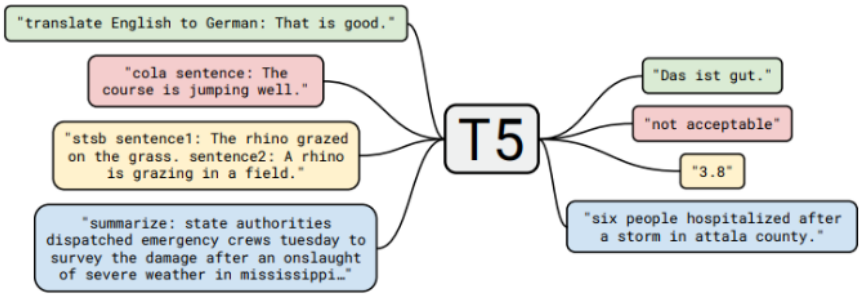


Figure 3.7: T5 text-to-text framework
[37]

3.4.3 PEGASUS

Model Overview: PEGASUS was proposed by Jingqing Zhang, Yao Zhao, Mohammad Saleh and Peter J. Liu on Dec 18, 2019. PEGASUS is specially developed for abstractive summarization by masking complete sentences while doing its pre-training so that coherent and contextually rich summaries. Some variations of this model showed not only just human-level summary also it showed human beating performance in generating abstractive summary.

Architecture: Similar to the Bart and T5, Pegasus also a transformer-based model so it includes encoder-decoder. Pegasus consists of 16 layers of transformer encoder-decoder in each component. The encoder-decoder architecture of Pegasus is similar to the BART model with multiple layers of transformer block. Again, each block with multi-head-self-attention and feedforward neural network[39].

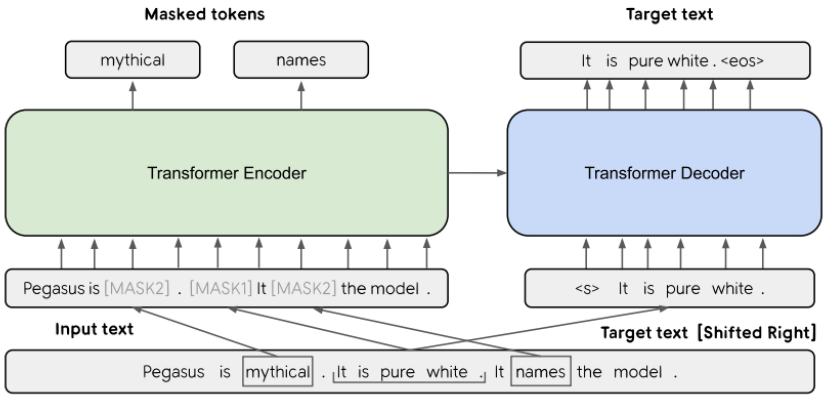


Figure 3.8: The base architecture of Pegasus
[39]

Pre-Training Mechanism: It occupies a self-supervised objective, Gap Sentence Generation (GSG) for summarization. GSG helps Pegasus learn to reconstruct a gap sentence. This part is very crucial to generate a quality abstractive summary. Along with GSG, an MLM was applied to the encoder. The steps of this mechanism are below:

GSG:

- **Selecting Sentences:** Some sentences are removed from a document.
- **Creating a Summary:** The removed sentences are combined to create a pseudo-summary.
- **Masking:** The spots in the source document where the sentences were removed are marked with special tokens like [MASK1].
- **Training:** The training objective of the model involves generating the omitted sentences based on the remaining text.

  To choose which sentences to remove, three strategies are used. Pick any random sentences, choose the first few sentences or select sentences that are most important, determined by comparing each sentence to the rest of the document using a metric called ROUGE-F1.

MLM (Masked Language modeling):

- **Masking Token**: A selection is made of 15% of the total words in the text.
- **Replacing Tokens**: A portion of these selected tokens were exchanged for a mask token or a random token, and sometimes remianed as it was.
- **Training**: The model learns to infer the intended words depending on contextual information.

This approach allowed PEGASUS to effectively learn summarization tasks by focusing on generating important sentences from the remaining text.

Pre-training Dataset:

- **Corpus:** PEGASUS was pre-trained on the C4 and HugeNews corpora.
- **Fine-tuning Dataset:** Fine-tuned on CNN/Daily Mail, XSum and arXiv datasets [39].

Why we selected: In our study we have selected Pegasus because of its robustness across domains. It has been evaluated on diverse datasets. It has an innovative sentence gap pre-training objective. And it outperforms many existing models. Its specialized training methodology ensures high-quality output for complicated texts. For instance, it achieved a ROUGE-1 score of 44.17 on the CNN/DailyMail dataset and 47.21 on the XSum dataset [39].

3.4.4 DistilBART

Model Overview: Distilbart is a distilled version of Facebook AI's BART model. This model is tailored for tasks like abstractive summarization. DistilBart retainss much of BART's performance while being more efficient in terms of speed and size through a process called knowledge distillation. DistilBART is very efficient in terms of real-time applications without sacrificing performance. It is capable to handle a large volume of text data at once. [40]

Architecture: DistilBART follows the encoder-decoder structure like most of the transformer-based models. This model has two core structural parts, an encoder and a decoder. The input text is encoded by the encoder and then it transforms them into a sequence of hidden states that capture its contextual meaning. Subsequently, the hidden representations fed to the decoder to produce a final sequence of output tokens. A sequence-to-sequence task like summarization, where it's essential to thoroughly understand the entire input before generating the output, this architecture is particularly effective. The DistilBART components are the BARTEncoder and the BARTDecoder. The model consists of 12 layers of encoders, which is similar to BART. But since it is a distilled version of BART, it consists of only 6 layers of decoder rather than 12 layers like BART. The embedding model output meaning the hidden size is 1024 and utilizes 16 attention heads. [41]

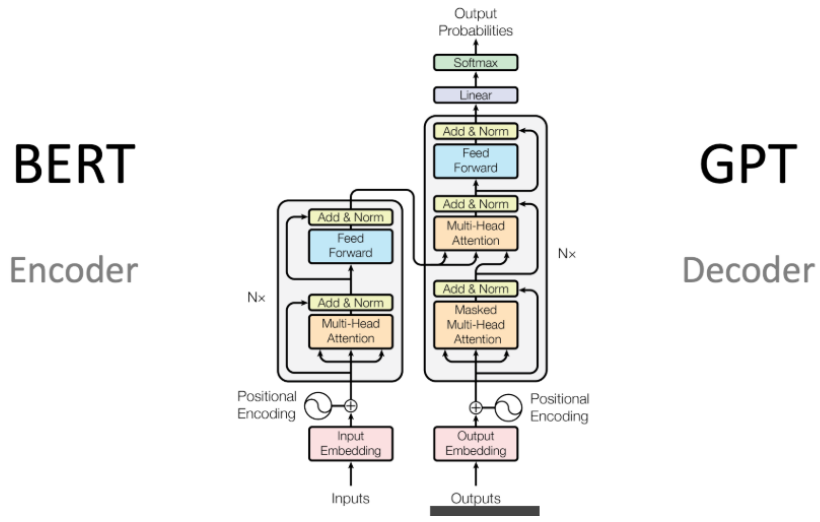


Figure 3.9: Transformer architecture
[41]

Why Selected: It is lightweight and efficient in nature, hence it has been selected for large-scale summarization tasks where the computational resources will be at stake.
These models represent a wide variety of architectures that are capable of doing abstractive summarization effectively at varying input lengths and complexity levels. The selection balances a high performance, for example, BART and PEGASUS, along with computational efficiency, such as DistilBART and T5. This comparative evaluation enables a comprehensive understanding of summarization performance across various model families.

3.5 Evaluation Metrics: Semantic Similarity, Coherence, Human Evaluation

3.5.1 Semantic Similarity

Semantic Similarity Overview
Semantic similarity is also a natural language processing (NLP) task like summarization. It involves, in quantitative measure, how closely two texts are similar with each other. It eases various NLP tasks to measure the similarity of texts, such as text classification, information retrieval, text summarization, essay evaluation, machine translation, and question answering [42].

Used Model for Embedding
We utilized all-MiniLM-L6-v2 model for generating the embeddings of the MOOC reviews and the generated summaries to compute the semantic similarity. The all-MiniLM-L6-v2 model is designed specially to generate high semantic sentence embeddings. This model is a part of the Sentence-Transformers framework. Sentence-Transformers is not optimized for token-level tasks like traditional BERT models; rather, it utilizes an architecture which enable direct extraction of sentence-level embeddings via a pooling strategy [43]. Moreover, this model is a compact version of BERT that uses deep self-attention distillation with significantly lesser parameters but retain the performance. It uses only 6 transformer layers and a 384-dimensional output vector but still gives a good trade-off between speed and accuracy. So this is an efficient version built upon the MiniLM architecture [44].

Evaluation methodology: It contains the following steps:

- **Embedding Generation:** We passed both the concatenated course reviews (original text) for a specific course and the generated summary for that course's reviews through the all-MiniLM-L6-v2 model to acquire sentence embeddings. The semantic meaning of the encoded input text is conveyed by the embeddings in a vector format.
- **Computation of Cosine Similarity:** From the Scikit-learn library, we used the `cosine_similarity()` function to compute the cosine similarity between the original text and the generated summary. This method produces a score between -1 to 1 which represents how well the summary corresponds to the original text. Higher value indicates greater semantic similarity.
- **Score Aggregation:** We computed the semantic similarity for all the courses present in our dataset. So we averaged all the resulting scores to provide an overall semantic similarity score.

3.5.2 Coherence

Overview Of Coherence metric for Summarization
The coherence metric for evaluating abstractive summaries basically measures the logical flow of a text and structural integrity. It represents the logical and semantic

```python
def evaluate_semantic_similarity(original_text, generated_summary):
    # Load pre-trained sentence transformer model
    model = SentenceTransformer('all-MiniLM-L6-v2')

    # Generate embeddings for original text and summary
    embeddings = model.encode([original_text, generated_summary])

    # Compute cosine similarity
    similarity_score = cosine_similarity([embeddings[0]], [embeddings[1]])[0][0]
    return similarity_score
```

Figure 3.10: Implementation of semantic similarity metric

connectivity between sentences in a text [45]. For multi-sentence summaries, it is very important because it should examine whether the adjacent sentences can disrupt readability for the reader. There are many ways to compute a coherence score. But we followed the cosine similarity based approach where, at first, an embedding is created, and then the similarity between two adjacent sentences is calculated. We used the same model, "all-MiniLM-L6-v2", for creating the embedding.

Implementation Steps:
First, we passed all the summaries one by one in a coherence measurement function, and that function worked in the following way:

- We splitted the whole text into sentences using the split() function.
- After that we converted the split sentences into numerical vectors (embeddings) using the loaded pre-trained "all-MiniLM-L6-v2" model.
- We also handled the single-sentence case by checking if the len(embedding) is less than two. If it is less than two, that means there is no need to calculate smooth transition between sentences so we returned the score as 1.0, assuming the coherence is perfect.
- Next, using the cosine_similarity function, we calculated each adjacent sentence pair to calculate the cosine similarity score, which measures how similar those sentence pair. And stored each similarity score in a list. The scores are between 0 and 1. A higher value suggests higher coherence in the text.
- Finally, returned the mean similarity score to get an overall coherence score of the summary.
- We did this whole thing in a loop for all the course summaries to measure their coherence score. And finally, averaged all the summary scores to produce an overall coherence score of all the courses.

3.5.3 Human Evaluation:

For our study, human evaluation of the generated summary is a very important matter, as our dataset does not contain any human-written summaries. Moreover, we trained the models on a different dataset which contains product review summaries instead of student feedback summaries. Therefore, we need human judgment for our generated summaries to check whether they are good enough in terms of fluency,

relevance and faithfulness or not. We asked two human evaluators to evaluate our generated summaries from MOOC feedback. Since we employed four abstractive models for generating summaries and we had 18 courses as a whole, we selected 10 courses. So, a total of 40 summaries from four models (1 course summary for four models) were evaluated by humans to measure the Cohen's kappa score.

Cohen's Kappa: Cohen's Kappa κ is a statistical measure that measures the agreement between two distinct persons who independently rate the same set of items [46].

Formula:

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)} \tag{3.1}$$

Where:

- $\Pr(a)$ represents the level of agreement actually observed between two raters.
- $\Pr(e)$ it represents the amount of expected agreement by chance.

Interpretation of κ Values
According to McHugh [46], the κ value, which indicates the level of agreement, is evaluated following the interpretation as table 3.1.

Table 3.1: Interpretation of Cohen's Kappa value

Kappa Value	**Strength of Agreement**
< 0.00	Poor
0.00–0.20	Slight
0.21–0.40	Fair
0.41–0.60	Moderate
0.61–0.80	Substantial
0.81–1.00	Almost Perfect

In our study, since we evaluated our summaries by humans, it was important to assess how consistently the raters agreed; hence, we utilized Cohen's Kappa score. We asked the evaluators to rate each summary on fluency, relevance and factual accuracy based on the original reviews.

Relevance: It measures how relevant the generated summary is to the original content.
Fluency: It measures the grammatical correctness, smoothness of phrasing and readability.
Factual Accuracy: It measures the faithfulness, meaning the generated summary actually contains true information from the original text or not [47].

Mean Score Computation

In addition to Cohen's Kappa, we also computed the **mean scores** from the human ratings for each dimension: *relevance*, *fluency*, and *factual accuracy*, to provide an aggregated quality assessment of the summaries.
The scoring process was as follows:

- Each rater rated every summary (from each of the four models) on a scale of 1 to 5 for each metric.
- We grouped the scores based on their evaluation dimension (i.e., relevance, fluency, factual accuracy).
- The mean score for each metric was computed separately for each rater.
- Finally, the average of the two raters' scores for each metric was taken to compute the **final mean score per dimension**.

This computation was performed using a Python script with the `NumPy` library, ensuring systematic and consistent averaging of scores across the 10 evaluated summaries per model. These final mean scores provided a numeric indicator of how each model performed according to human judgment.

Chapter 4

Model Adaptation and Domain Application

4.1 Dataset for Fine-Tuning: "AmaSum"

4.1.1 overview:

The AmaSum dataset was in JSON format. It contains reviews of Amazon products and website summaries. The website summary data fields contain entries including 'verdict': Summary Statement, 'pros': pros of the product, 'cons': cons of the product, 'aspects': empty dictionary, 'publication date': empty string, 'rating': empty string and 'source'.

4.1.2 Preprocessing the dataset

We were intended to train our pre-trained models on reference summaries, so all the data entries were not needed.

- At first we loaded the train, test, and validation split from the dataset.
- Extracted all the customer reviews for each file and merged them.
- Then extracted the pros and cons from the website summaries data field.
- The customer reviews and summary are stored in a dictionary. After that, all the reviews and summary dictionaries were stored in a list.
- Later, we converted the list to a pandas dataframe and did this for all training, test, and validation splits.
- We introduced two new columns, `"input text"` and `"target text"`. The combined reviews stored in the input text and the target text consist of pros and cons (concatenated pros and cons for training).
- We tokenized the training data using the specific tokenizer for each specific model.

Finally, we fine-tuned the pre-trained BART, DistilBart, Pegasus, and T5 models.

4.2 Model Fine-Tuning Setup

We fine-tuned four transformer-based models on the preprocessed AmaSum dataset using the Hugging Face `Transformers` library. The input texts were aggregated product reviews, and the target outputs were the corresponding human-written summaries (pros and cons). All training scripts were implemented in a Kaggle notebook and executed on GPU-accelerated environments using CUDA, where available. Model weights and logs were saved during the training process for further evaluation. Model fine-tuning was performed through the Hugging Face `Trainer` API, which provides built-in support for gradient accumulation, checkpointing, evaluation, and logging. Due to different sizes and architectural complexities, the training parameters varied across the models. The training configurations for the models were:

- **Output Directory:** `./[model-name]-finetuned` (custom directory to save the trained model and checkpoints).
- **Evaluation Strategy:** `epoch` (when to run evaluation, here after completing an epoch).
- **Save Strategy:** `epoch` (model checkpoints were saved after every epoch).
- **Save Total Limit:** 2 (maximum checkpoints to be kept).
- **Load Best Model At End:** `True` (the best checkpoint would be loaded after training is complete).
- **Metric for Best Model:** `eval_loss` (metric which is used to select the best checkpoint).
- **Greater is better:** false.
- **Per Device Batch Size:**
  - BART: 4
  - Pegasus: 2
  - T5: 4
  - DistilBart: 8
- **Epochs:** (complete iterations over the training dataset).
  - BART: 4
  - Pegasus: 3
  - T5: 7
  - DistilBart: 3
- **Weight Decay:** 0.01 (penalizes large weights to reduce overfitting; regularization).
- **Logging Directory:** `./logs` (location to save logs and other metrics).

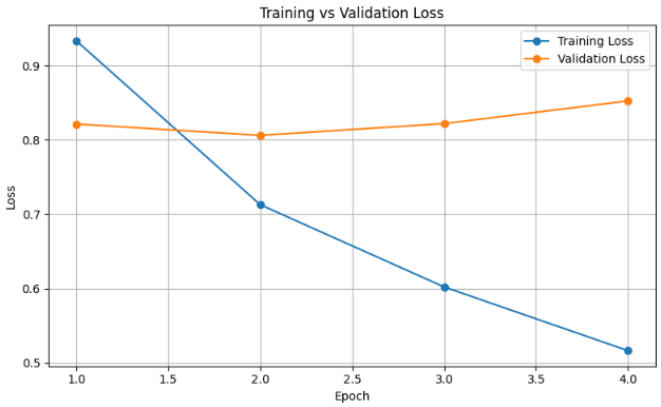


Figure 4.1: Training and validation loss of BART

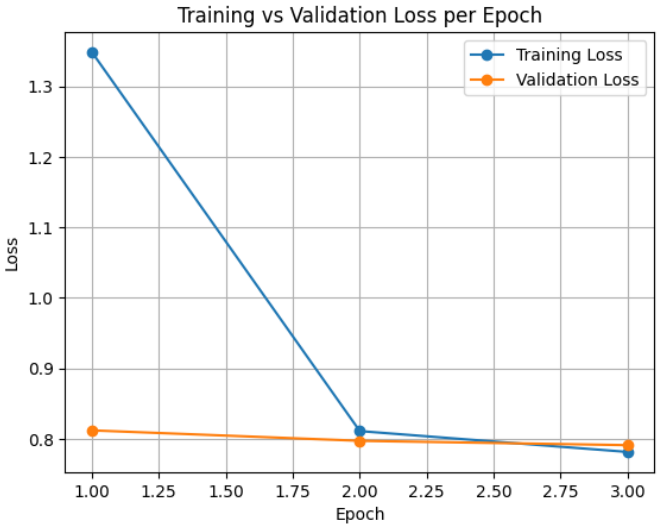


Figure 4.2: Training and Validation loss of Pegasus

From the Figure 4.1, we can see the training loss has decreased smoothly up to epoch 4. But the validation loss after epoch 2 started to increase. So, we got our best model at epoch 2. Pegasus in Figure 4.2 demonstrates stable learning with decreasing validation loss across all the epochs. Due to its large size, only 3 epochs were completed. Next, as shown in Figure 4.3, we applied 7 epochs to T5. However, there were no significant improvements in validation loss after epoch 5. So we saved the best model at epoch 5. Lastly, Figure 4.4 shows the training and validation loss graph of DistilBart. The validation loss started to increase slightly after epoch 2, from 0.8571 to 0.8672. Therefore, we saved the best model at epoch 2. After finishing the training using the trainer.save_model() the fine-tuned model was explicitly saved to ensure that the best model checkpoint with low validation loss was preserved for further inference and evaluation.

4.3 Summary Generation on Mooc Reviews

After saving the best models based on their evaluation loss, we loaded each model to apply them to the combined MOOC feedback. The following steps we followed

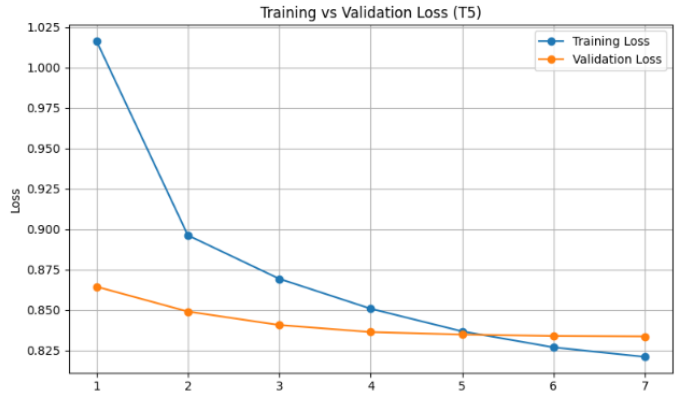


Figure 4.3: Training and validation loss of T5

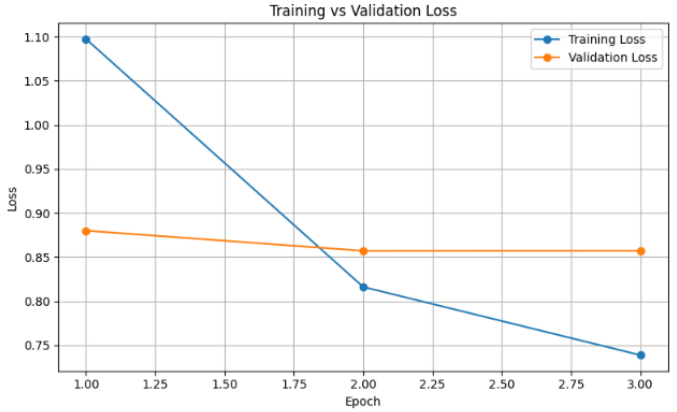


Figure 4.4: Training and validation loss of DistilBart

to generate course-wise summaries:

- At first, we read the MOOC review dataset using `pd.read_csv()`.
- Cleaned the data using the necessary text cleaning steps.
- Based on the course_title we made a dictionary. Where we combined all the feedback of a specific course in a list and assigned the course title as their key.
- Then applied the fine-tuned models on the dictionary items.

To perform abstractive summarization with each fine-tuned model, we set up a custom summary generation method. The method in question simplifies the reviews in bulk by first tokenizing each review with the tokenizer made for the model, creating tensors the model can work with. Once the tensors are generated, the model is used to create the concise summaries.
To enhance the quality of the output generated, the function was designed with several important decoding parameters which impact the generated output quality.

- **Beam Search**: We used beam search with `num_beams=8` to explore multiple decoding paths and select the most coherent summary.
- **Maximum Length**: We established the max length=256 in the algorithm to manage the length of the output summary.

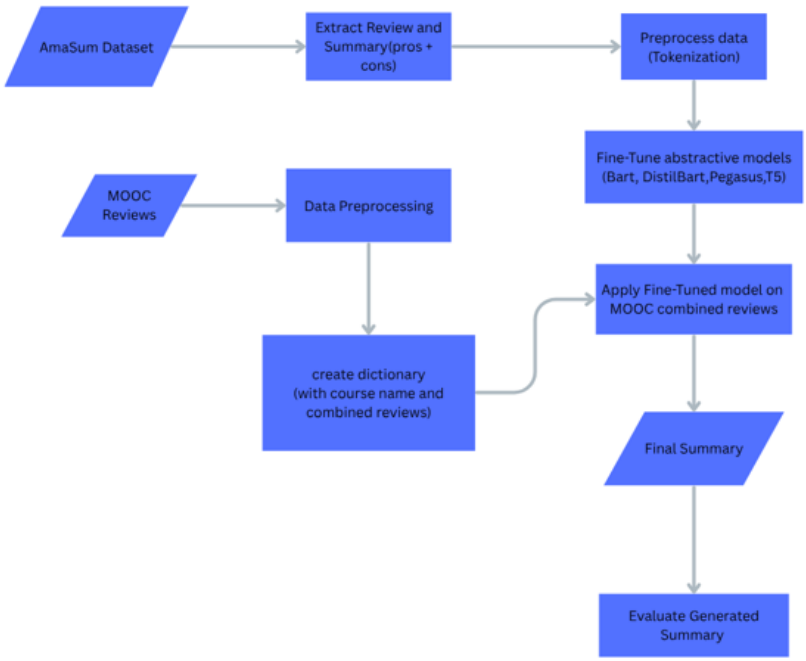


Figure 4.5: Workflow Diagram

Original Text:

One of the best, well organised and convenient coursed that I have participated in!VG, started with setup and included all basics needed. A good overview of using and configuring JIRA for a variety of Agile methodologies. Complicated topic but very interesting to have a nice overview of Jira's functions. tasks could be more exciting, but all in all a good intro to JIRA in agile context. Very easy to follow.Very basic course, most of the stuff you can learn with a quick google search. The videos itself are very low pace and focus mainly on how to use Jira on a very basic level. I had to listen to the videos to 1,5x speed for it to be at a normal listening level. The content of this course is valid and helps to understand the basic funcionalities of Jira and its agile possibilities.The videos are poorly done. The person who is explaning everything is talking with such a monton voice that it is hard to follow. The videos are too long. It would be better to split them in more videos. There are no parts of repetition, so you hear everything just one time in the video. They only teach you how to use Jira but nothing about agile. A course as long as this is not needed to learn Jira. Real life examples would make it much clear and enable faster learning. Also, please consider implementing real life software examples in lab-tasks, not just "add item 1".....

Figure 4.6: original text 1

- **Early Stopping**: We had early stopping enabled to halt the decoding when an adequate summary is generated.
- **Repetition Penalties**: Length penalty=1.5 and repetition penalty=2.0 were in place to manage overly long and repetitive outputs.
- **No Repeat N-Gram**: no_repeat_ngram size=3 was applied to manage repeated phrases in the output summary.

Inference was run on GPU if possible, due to the size of the model, also helped speed up inference time. Combined course reviews for each course were stored in a variable as a single input string. A summarization function was then repeatedly applied to the entire dataframe. The summary was output to a dictionary with the course title as the key.
This automated approach to summarization enabled us to produce concise and

Generated Summaries

Facebook Bart: This online course is designed to help you learn how to use Jira in a variety of ways, It focuses on the basics of using Jira as well as how to set up and use it effectively, Most of the content is free, so you don't have to pay for a subscription. Some of the topics are more technical than others, but most of the material is useful for those who want to get started with Jira right away.

Pegasus : This course is designed for those who want to learn how to use Jira, but don't want to spend a lot of time learning the ins and outs of the software. The course doesn't provide much in the way of real-world examples, so you may not be able to use the software immediately.

T5 : Designed to help you learn how to use Jira in agile environments, this course is designed to give you the tools you need to get started quickly and efficiently, It's easy to understand how to set up your own site functions. The videos aren't as engaging as some of the other courses on the market.

DistilBart : A comprehensive overview of how to use Jira on a very basic level, Coursera's free JIRA-based course is well-organized and easy to follow, A good value for the money, as you get a free online certificate. Some of the content isn't worth the cost, and some of the topics are fairly basic, The videos are short, so you'll need to listen to them at 1,5x speed for it to be at a good pace.

Figure 4.7: sample generated summary 1

Original Text:

The curriculum was poorly organized. A two-hour Youtube course can teach much more about html/css/js than all this 4-week long course. At least you should expect to be able to set up a simple page with some decent (beginner) layout and some interactvity after finishing the course. But no! This much time spent for so little stuff. The most annoying course ever. I can't imagine why this course is included to Java specialization.HTML AND CSS is fine. Introduction to java is hell. I bet it is not as complicated as in the video. I signed up for "Java Programming and Software Engineering Fundamentals". Instead of learning Java, this class goes over Javascript/HTML/CSS which I don't need at all. yet, you are required to work in BlueJ, (google it) with custom Javascript functions that you are not allowed to view or use outside the course. I acknowledge the high level of the syllabus, it was really designed to bring some advanced contents although advertised as "for beginners".All core concepts such as data types, data structures are explained very sketchy. OOP paradigm isn;t explained at all. I don't know if I just had issues with my english or it was not the best course structure. The JAVASCRIPT part of the course was to difficult for me I didn't understand enough. I would like to see more tutorial.....

Figure 4.8: original text 2

course-relevant summaries, reflecting the collective feedback from students. These summaries may be helpful for course intended audiences and for course designers.

Generated Summaries:

Facebook Bart: A comprehensive course that covers a wide range of programming concepts, including the basics of HTML, CSS, JavaScript, and more, It's also easy to learn how to use with BlueJ, which is an open-source version of the Java programming language. Some of the topics covered are difficult to understand for non-beginners, especially when it comes to data types and data structures, Less than half of the material is in English.

Pegasus : This course is designed for those who want to learn the basics of JavaScript, but don't want to spend a lot of time learning how to use the language, The course covers a wide range of topics, from basic concepts to more advanced techniques. Some students find the course difficult to follow due to the lack of video material.

T5 : a good introduction to the programming language, but it's also a great course for those who want to learn more about HTML and CSS. This is a beginner's course, as it doesn't cover all of the fundamentals of JavaScript.

DistilBart : The Simple Image Programming Program is a free online program that teaches basic programming concepts in the language of HTML and CSS, It's also a free course for beginners who want to learn more about how to use the language. Some of the materials are hard to understand, and some of the content isn't as detailed as others on our list.

Figure 4.9: sample generated summary 2

Chapter 5

Result and Analysis

In this section, we evaluate the performance of our fine-tuned abstractive summarization models – BART, DistilBART, PEGASUS, and T5 using automatic metrics and human ratings. The assessment was done across three primary areas: semantic similarity, coherence, and human ratings with Cohen's kappa.

5.1 Semantic Similarity

Semantic similarity measures the extent to which the generated summary preserves the original meaning of the input reviews. As shown in Table 5.1 T5 achieved the highest semantic similarity score, followed by BART and PEGASUS.

Table 5.1: Semantic Similarity Scores

Model	**Semantic Similarity**
BART	0.57
DistilBART	0.56
PEGASUS	0.57
T5	**0.60**

The high semantic similarity score achieved by T5. It uses a single unified text-to-text framework and the fact that it modeled every task (even summarization) as a transformation of text. This helps capture and maintain the core ideas of the original content. On the other hand, the DistilBART model is the smallest version of BART, and its semantic similarity scores are slightly diminished potentially due to some of the model components being stronger in BART, which has a greater number of attention layers.

5.2 Coherence

Coherence measures the logical flow, grammatical correctness, and structural consistency of the generated summaries. PEGASUS has the highest coherence score, followed by DistilBART and BART, as shown in Table 5.2
PEGASUS's dominance is due to its pretraining objective: gap-sentence generation which explicitly induces learning of inter-sentence dependencies and hence coher-

Table 5.2: Coherence Scores

Model	Coherence
BART	0.33
DistilBART	0.35
PEGASUS	**0.40**
T5	0.28

ence. After Pegasus, Bart and DistilBart performed moderately on the coherence metric. Even though T5 is strong semantically, it suffers in terms of coherence. It demonstrates a trade-off between fact alignment and surface fluency in our domain.

5.3 Human Evaluation via Cohen's Kappa

To validate the automatic metrics, human evaluators rated each model's summaries on three qualitative criteria: relevance, fluency, and factual accuracy. **Cohen's Kappa** was used to assess inter-rater agreement—that is, the degree of consistency between the two annotators' evaluations. It does not measure the quality of the summaries themselves but rather how much the two raters agreed on their assessments. The updated Kappa values are shown in Table 5.3, where most scores fall within the *"slight"* or *"less than chance"* agreement range. Notably, several values are near or below zero, indicating a high degree of subjectivity and inconsistency among raters.

Table 5.3: Cohen's Kappa for Inter-Rater Agreement

Model	Relevance	Fluency	Factual Accuracy
BART	0.19	0.00	0.19
DistilBART	0.09	0.00	-0.04
PEGASUS	-0.19	0.29	0.09
T5	0.04	0.00	-0.01

Despite using a clear rubric for evaluation, these low Kappa scores suggest that annotators interpreted the evaluation criteria differently or found it difficult to assess dimensions such as fluency and factual accuracy consistently. The fact that multiple models received zero or negative scores, particularly in fluency and factual accuracy, highlights the subjectivity involved in evaluating abstractive summaries. This further reinforces the need to combine human evaluation with automatic metrics to form a more balanced assessment.

5.3.1 Mean Human Rating score

We computed the **mean human ratings** using responses on a 1–5 Likert scale across the criteria of relevance, fluency, and factual accuracy. The scores are presented in Table 5.4.

Table 5.4: Mean Human Evaluation Scores (1–5 Scale)

Model	Relevance	Fluency	Factual Accuracy
BART	3.55	4.55	3.55
DistilBART	3.80	4.40	3.60
PEGASUS	4.35	**4.75**	4.40
T5	**4.40**	4.65	**4.45**

5.3.2 Interpretation and Key Findings

The average score of human ratings gives a more comprehensive view of the perceived standard of the summaries produced by each of the models:

- **T5** received the highest rating for factual accuracy (4.45) and relevance (4.40) thus making it the best at generating summaries that were at once informative (high information retention) and faithful (accurate to source text). With a fluency score of 4.65 overall, T5 excelled on all three indices.
- Overall, **PEGASUS** received the highest scores for fluency with a 4.75 as the highest, generally being the most fluent and natural of models across all three human evaluations. PEGASUS also demonstrated good reliability in factual accuracy (4.40) and relevance (4.35), suggesting that PEGASUS generated the most fluid and interesting summaries with good information retention.
- **BART** is a competent method throughout all three indexes with a mean score of 4.55 and comparable scores for relevance and accuracy (3.55). Although BART did not lead in any of the three aspects, it is simply exhibiting necessary performance.
- **DistilBART**, performed above what would be expected for a light model. It received 3.80 for relevance, 4.40 for fluency and 3.60 for factual accuracy. It is slightly behind the larger models, in terms of accuracy, but still offers readable and reasonably relevant summaries.

To conclude, The findings, taken together, highlight the significance of mean ratings and inter-rater agreement in evaluating the suitability of summarization models. The mean scores numerically reflect the subjective assessment of the calibre of the generated summaries, whilst Cohen's Kappa shows the consistency of assessments. Most significantly, this analysis leads us to the conclusion that the best models capable of producing high-quality summaries of MOOC reviews are T5 and PEGASUS.

Chapter 6

Conclusion

6.1 Limitations

Despite achieving promising outcomes with the fine-tuned summarization models, this research has certain limitations to consider when placing findings in context and looking forward in future research. First, the models were fine-tuned on a product review dataset but were applied to and evaluated on student feedback on courses. Though both are human-written feedback and their language style are the same, they belong to different domains. Secondly, the human evaluation was limited to a small sample size, only 10 summaries per model. While a sample size of 10 was adequate for preliminary qualitative analysis, the low number of samples does carry marked statistical uncertainty. Cohen's Kappa, for instance, which was utilized for inter-rater reliability measures, can be very unreliable and erroneous when calculated from a small dataset. In light of this, we must caution that the agreement scores should be interpreted carefully. Thirdly, this research lacked gold-standard, human-written reference summaries. As a result, we had to depend upon proxy evaluation metrics such as semantic similarity and coherence. Although semantic similarity and coherence are well-established measures in summarization literature, they do have limitations. For example, measures of semantic similarity can overlook fluency and factuality, while coherence measures do not always correspond to human readability assessments. Our omission of utilizing reference summaries also meant we were unable to use the more comprehensive automatic metrics (i.e., ROUGE, BLEU or METEOR) that are typically used to assess summarization quality. Finally, while the demonstrated models were only from fine-tuned forms based on pretrained checkpoints, no architectural alterations, decoding strategies, or reinforcement learning were used. All of which could have led to untapped performance potential.

6.2 Future Plan

Based on the contributions made by this research, a number of directions are possible for future work on the topic of abstractive summarization of MOOC reviews.

6.2.1 Scalable and Robust Human Evaluation

Future work extends the human evaluation dimension of previous documentation work both with respect to the scale of the human evaluation (in terms of the number of summaries evaluated) and the diversity of annotators. A more expansive evaluation set (e.g., 50-100 summaries per model) would provide more statistically robust agreement scores and allow for better analysis of inter-rater variability. The inclusion of domain experts (e.g., MOOC instructors in many cases) in the annotation process could provide greater depth to evaluations in terms of factual accuracy and relevance.

6.2.2 Increase the Number of Evaluators

In this study, there were only two evaluators, which may have produced some error based on the level of English each evaluator possessed. By increasing the number of evaluators, we could have diminished the variability error and we could have selected ratings when the rating would have more agreement.

6.2.3 Incorporation of Reference Summaries

Developing a human-written, manually curated data set of summaries of MOOC reviews would vastly improve the evaluation framework. A data set of this kind would allow serious evaluation, using established automatic evaluation metrics, such as ROUGE and BLEU. Both of which would provide better comparative benchmarks and allow for comparisons of models across studies.

6.2.4 Aspect-Based and Multi-View Summarization

Considering the multi-dimensionality of MOOC reviews, which cover framing dimensions of course content, instructor quality, assessment and platform experience, another avenue is possible for aspect-based summarization. An aspect-based summarization would mean sorting or tagging sentences by aspect and then summarizing each aspect in the paragraph or summary. The resulting summary would be more informative and organized by structure. This type of summary would better help prospective learners looking for detailed disclosures while matching the dimensions of MOOC reviews.

6.2.5 Model Ensembling and Controlled Generation

The findings from this study demonstrated complementary strengths across various models. For example, T5 had strong performance in terms of semantic fidelity whereas PEGASUS produced more fluent output. Future work could examine a number of ensemble techniques that model the outputs of a number of models, or methodologies for controlled generation with specific summary features (e.g., factuality vs. concision).

6.2.6 Cross-Platform and Cross-Domain Validation

For the sake of external validity, future work should test the models on reviews from various MOOC platforms, such as edX, Udacity, and FutureLearn. Moreover, applying the process to other domains—like product reviews, health care evaluations, or abstracts of research papers—would provide generalizability and the identification of unique challenges characteristic to a domain.

To sum up, our study examined the application of large transformer based abstractive summarization models to extract key insights from MOOC learner reviews and generate brief, coherent summaries. The meeting feedback we examined is subjective and unstructured, so we fine-tuned BART, PEGASUS, T5 and DistilBART using the AmaSum dataset which includes a large corpus of summaries and reviews of Amazon products, as there were no available reference summaries for MOOCs. We use a combination of automatic measures and human assessments as our evaluation. T5 was the strongest model in terms of semantic similarity and factual accuracy while PEGASUS was the best model for fluency. Cohen's Kappa scores indicated fair to moderate agreement amongst human reviewers, which illustrates that summary evaluations are somewhat subjective, but can be deemed relatively consistent. The calculated mean scores provided a clearer representation of summary quality overall, with T5 and PEGASUS producing the most well-rounded outputs. There were a few limitations to be aware of, including human evaluation being based on a small sample size, as well as the domain shift of the training data. Overall, the findings confirm that it is feasible to fine-tune pre-trained summarization models on different feedback domains and apply them to the educational feedback collected in this study. This research offers a foundational technique for summarizing MOOC reviews at scale, thereby helping prospective learners make timely choices and helping instructors identify an opportunity for improvement in their courses. Future work could look at larger-scale evaluations, summarization data that is domain-specific, and create interactive summarization frameworks for individualized learning.

Bibliography

[1] Y. Li, "Moocs in higher education: Opportunities and challenges," in *Proceedings of the 2019 5th International Conference on Humanities and Social Science Research (ICHSSR 2019)*. Atlantis Press, 2019/05, pp. 48–55. [Online]. Available: https://doi.org/10.2991/ichssr-19.2019.10

[2] B. Khan, Z. A. Shah, M. Usman, I. Khan, and B. Niazi, "Exploring the landscape of automatic text summarization: A comprehensive survey," *IEEE Access*, vol. 11, pp. 109 819–109 825, 2023.

[3] M. Wen *et al.*, "Sentiment analysis in mooc discussion forums: What does it tell us?" 2016.

[4] L. Wang, G. Hu, and T. Zhou, "Semantic analysis of learners' emotional tendencies on online mooc education," *Sustainability*, vol. 10, no. 6, p. 1921, 2018.

[5] M. Neumann and R. Linzmayer, "Capturing student feedback and emotions in large computing courses: A sentiment analysis approach," in *Proc. SIGCSE '21*, 2021.

[6] K. Lundqvist, T. Liyanagunawardena, and L. Starkey, "Evaluation of student feedback within a mooc using sentiment analysis and target groups," *International Review of Research in Open and Distributed Learning*, vol. 21, no. 3, pp. 140–156, 2020.

[7] M. J. Gomez, M. Calderón, V. Sánchez, F. J. G. Clemente, and J. A. Ruipérez-Valiente, "Large scale analysis of open mooc reviews to support learners' course selection," *Expert Systems with Applications*, vol. 210, p. 118400, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422015081

[8] T. Shaik, X. Tao, C. Dann, H. Xie, Y. Li, and L. Galligan, "Sentiment analysis and opinion mining on educational data: A survey," *Natural Language Processing Journal*, vol. 2, p. 100003, 2023.

[9] S. Gottipati, V. Shankararaman, and J. Lin, "Latent dirichlet allocation for textual student feedback analysis," in *Proceedings of the 26th International Conference on Computers in Education ICCE 2018: Manila, Philippines, November 28-30*, 2018, pp. 220–227.

[10] X. Chen, F. Wang, G. Cheng, M.-K. Chow, and H. Xie, "Understanding learners' perception of moocs based on review data analysis using deep learning and sentiment analysis," *Future Internet*, vol. 14, no. 218, 2022.

[11] A. Bražinskas, M. Lapata, and I. Titov, "Unsupervised opinion summarization as copycat-review generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 5151–5169. [Online]. Available: https://aclanthology.org/2020.acl-main.461/

[12] O. Almatrafi and A. Johri, "Improving moocs using information from discussion forums: An opinion summarization and suggestion mining approach," *IEEE Access*, vol. 10, pp. 15 565–15 573, 2022.

[13] J. Verma and A. Patel, "Evaluation of unsupervised learning based extractive text summarization technique for large scale review and feedback data," *Indian Journal of Science and Technology*, vol. 10, pp. 1–6, 05 2017.

[14] A. Khan, M. A. Gul, M. Zareei, R. R. Biswal, A. Zeb, M. Naeem, Y. Saeed, and N. Salim, "Movie review summarization using supervised learning and graph-based ranking algorithm," *Computational Intelligence and Neuroscience*, vol. 2020, no. 1, p. 7526580. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2020/7526580

[15] K. Ganesan, C. Zhai, and J. Han, "Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions," in *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, Beijing, 2010, pp. 340–348. [Online]. Available: http://timan.cs.uiuc.edu/downloads.html

[16] S. Negi, K. Asooja, S. Mehrotra, and P. Buitelaar, "A study of suggestions in opinionated texts and their automatic detection," in *Proc. 5th Joint Conf. Lexical Comput. Semantics*, 2016, pp. 170–178.

[17] A. Bražinskas, M. Lapata, and I. Titov, "Few-shot learning for opinion summarization," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 4119–4135. [Online]. Available: https://aclanthology.org/2020.emnlp-main.337/

[18] S. Ge, J. Huang, Y. Meng, and J. Han, "Finesum: Target-oriented, fine-grained opinion summarization," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, ser. WSDM '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 1093–1101. [Online]. Available: https://doi.org/10.1145/3539597.3570397

[19] S. B. R. Chowdhury, C. Zhao, and S. Chaturvedi, "Unsupervised extractive opinion summarization using sparse coding," 2022. [Online]. Available: https://arxiv.org/abs/2203.07921

[20] S. Angelidis and M. Lapata, "Summarizing opinions: Aspect extraction meets sentiment prediction and they are both weakly supervised," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*,

E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3675–3686. [Online]. Available: https://aclanthology.org/D18-1403/

[21] Y. Suhara, X. Wang, S. Angelidis, and W.-C. Tan, "OpinionDigest: A simple framework for opinion summarization," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 5789–5798. [Online]. Available: https://aclanthology.org/2020.acl-main.513/

[22] J. Im, M. Kim, H. Lee, H. Cho, and S. Chung, "Self-supervised multimodal opinion summarization," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 388–403. [Online]. Available: https://aclanthology.org/2021.acl-long.33/

[23] H. Li and S. Chaturvedi, "Rationale-based opinion summarization," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, K. Duh, H. Gomez, and S. Bethard, Eds. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 8274–8292. [Online]. Available: https://aclanthology.org/2024.naacl-long.458/

[24] H. Li, S. Basu Roy Chowdhury, and S. Chaturvedi, "Aspect-aware unsupervised extractive opinion summarization," in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 12 662–12 678. [Online]. Available: https://aclanthology.org/2023.findings-acl.802/

[25] N. Balaji, M. N, S. P, N. Bhavatarini, and S. A, "Text summarization using nlp technique," 10 2022, pp. 30–35.

[26] A. AlArfaj and H. Mahmoud, "An intelligent tree extractive text summarization deep learning," *Computers, Materials Continua*, vol. 73, pp. 4231–4244, 06 2022.

[27] R. Margarido, T. Pardo, G. Antonio, V. Fuentes, R. Aires, S. Aluisio, and R. Fortes, "Automatic summarization for text simplification: Evaluating text understanding by poor readers," *Proceedings of the XIV Brazilian Symposium on Multimedia and the Web*, 10 2008.

[28] H. Christian, M. Agus, and D. Suhartono, "Single document automatic text summarization using term frequency-inverse document frequency (tf-idf)," *ComTech: Computer, Mathematics and Engineering Applications*, vol. 7, p. 285, 12 2016.

[29] S. Aubakirov and I. Akhmetov, "Dynamic optimization of minimum document frequency for extractive text summarization using greedsum algorithm," in *2024 20th International Asian School-Seminar on Optimization Problems of Complex Systems (OPCS)*, 2024, pp. 33–37.

[30] K. Prashanth Kumar, G. Gandhimathi, S. Prabha, P. K. Naik, H. H. Abbas, and H. Shareef, "An empirical detailed analysis of text summarization using hybrid and integrated technology," in *2024 4th International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, 2024, pp. 162–166.

[31] T. G. Altundogan, M. Karakose, and O. Tokel, "Bart fine tuning based abstractive summarization of patients medical questions texts," in *2023 4th International Conference on Data Analytics for Business and Industry (ICDABI)*, 2023, pp. 174–178.

[32] S. Dhapola, S. Goel, D. Rawat, S. Vats, and V. Sharma, "Abstractive text summarization using transformer architecture," in *2024 IEEE 3rd World Conference on Applied Intelligence and Computing (AIC)*, 2024, pp. 13–17.

[33] V. Lendave, "How to obtain a sentiment score for a sentence using textblob?" *AI Trends*, October 28 2021.

[34] A. Payong and S. Mukherjee, "Understanding the bart model for accurate text summarization," https://www.digitalocean.com/community/tutorials/bart-model-for-text-summarization-part1, 2025, accessed: 2025-05-06.

[35] GeeksforGeeks, "BART Model for Text Auto Completion in NLP," https://www.geeksforgeeks.org/bart-model-for-text-auto-completion-in-nlp/, 2023, last updated: June 8, 2023. Accessed: 2025-06-15.

[36] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019. [Online]. Available: https://arxiv.org/abs/1910.13461

[37] G. Gupta, "Understanding the t5 model: A comprehensive guide," https://medium.com/@gagangupta_82781/understanding-the-t5-model-a-comprehensive-guide-b4d5c02c234b, 2024, accessed: 2025-05-06.

[38] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2023. [Online]. Available: https://arxiv.org/abs/1910.10683

[39] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," 2020. [Online]. Available: https://arxiv.org/abs/1912.08777

[40] Dataloop, "Distilbart tag – model library," n.d., accessed: 2025-06-14. [Online]. Available: https://dataloop.ai/library/model/tag/distilbart/

[41] M. A. I. Khan, "Understanding the distilbart model and rouge metric," Mar. 2025, published on MachineLearningMastery.com. [Online]. Available: https://machinelearningmastery.com/understanding-the-distilbart-model-and-rouge-metric/

[42] A. Upadhyay, A. Bhatnagar, N. Bhavsar, M. Singh, and P. Motlicek, "An empirical comparison of semantic similarity methods for analyzing downstream automatic minuting tasks," in *Proceedings of the 36th Pacific Asia Conference on Language, Information and Computation (PACLIC 36)*, 2022, pp. 563–573. [Online]. Available: https://aclanthology.org/2022.paclic-1.63.pdf

[43] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2019, pp. 3982–3992. [Online]. Available: https://www.aclweb.org/anthology/D19-1410

[44] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," *arXiv preprint arXiv:2002.10957*, 2020. [Online]. Available: https://arxiv.org/abs/2002.10957

[45] I. Tham, "How to evaluate llm summarization," *Medium (Data Science)*, Jan. 22 2025, 14 min read; accessed 2025-06-21; URL https://medium.com/data-science/how-to-evaluate-llm-summarization-18a040c3905d.

[46] M. L. McHugh, "Interrater reliability: The kappa statistic," *Biochemia Medica*, vol. 22, no. 3, pp. 276–282, 2012. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900052/

[47] Microsoft Learn, "A list of metrics for evaluating llm-generated content," June 2024, accessed: 2025-05-14. [Online]. Available: https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/generative-ai/working-with-llms/evaluation/list-of-eval-metrics

Human Evaluation Form

This appendix contains all the questions and evaluation criteria used in human judgement of automatically generated summaries using the four models. Participants were asked to read the original texts and the generated summaries carefully. Then rate the summaries on a 5-point linear scale in response to the following questions.

1. **Relevance**: How well does the summary reflect the key points of the original text?
2. **Fluency**: Is the summary grammatically correct and easy to understand?
3. **Factual Accuracy**: Does the summary stay true to the facts presented in the original text?

Later, the responses were used to assess the inter-rater agreement using Cohen's kappa and a mean rating score.

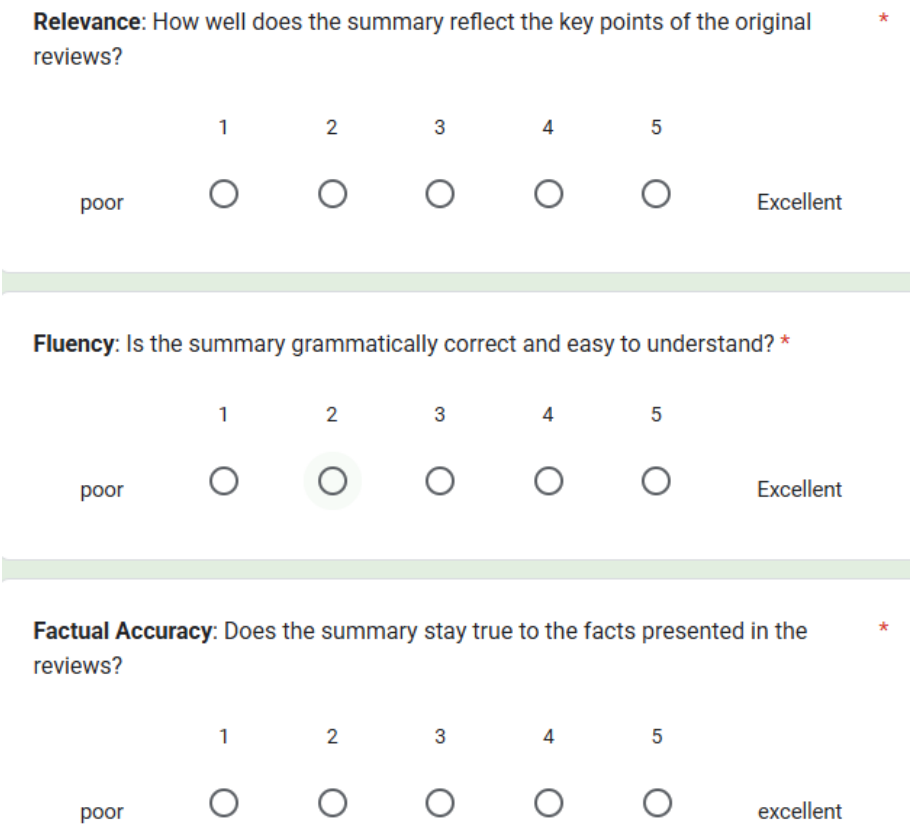


Figure 6.1: Human evaluation from