

# Using Generative Text Models to Create Qualitative Codebooks for Student Evaluations of Teaching

International Journal of Qualitative Methods  
Volume 23: 1–19  
© The Author(s) 2024  
DOI: 10.1177/16094069241293283  
[journals.sagepub.com/home/ijq](https://journals.sagepub.com/home/ijq)



Andrew Katz<sup>1</sup> , Mitch Gerhardt<sup>1</sup> , and Michelle Soledad<sup>1</sup>

## Abstract

Feedback is a critical aspect of improvement. Unfortunately, when there is a lot of feedback from multiple sources, it can be difficult to distill the information into actionable insights. Consider student evaluations of teaching (SETs), which are important sources of feedback for educators. These evaluations can provide instructors with insights into what worked and did not during a semester. A collection of SETs can also be useful to administrators as signals for courses or entire programs. However, on a large scale as in high-enrollment courses or administrative records over several years, the number of SETs can render them difficult to analyze. In this paper, we discuss a novel method for analyzing SETs using natural language processing (NLP) and large language models (LLMs). We demonstrate the method by applying it to a corpus of 5000 SETs from a large public university. We show that the method can extract, embed, cluster, and summarize the SETs to identify the themes they contain. More generally, this work illustrates how to use NLP techniques and LLMs to generate a codebook for SETs. We conclude by discussing the implications of this method for analyzing SETs and other types of student writing in teaching and research settings.

## Keywords

natural language processing, large language models, generative artificial intelligence, student evaluations of teaching, codebook generation, qualitative data analysis

## Introduction

Text and speech are essential elements of qualitative data used across many research domains (Miles et al., 2014; Pope et al., 2000). These data, which include interviews, transcripts, reports, social media posts, written answers to open-ended response questions, and essays are traditionally analyzed using methods such as qualitative thematic analysis (Terry et al., 2017). However, these approaches often require significant research investments in both time and cost due to third-party software and revisions (Basit, 2003). Moreover, as the data grow, so does the time needed for analysis, making large and existing datasets progressively difficult to manage. Despite these challenges, thematic analysis remains prevalent in qualitative research because it can extract detailed insights and themes (Miles et al., 2014; Pope et al., 2000; Terry et al., 2017).

During qualitative thematic analysis, researchers scrutinize data to extract or identify salient themes and concepts based on codes they have identified through initial rounds of coding. Under

the steps outlined by Braun and Clarke (2006) for thematic analysis, code generation happens in phases one and two of their six-phase process whereas theme identification and refinement happens in phases three through five. In the Braun and Clarke framing, codes denote features of the data that are notable to the researcher. Thus, the thematic analysis process involves identifying codes and then themes based on researchers' choices, such as theoretical stances and employing an inductive or deductive coding approach (Terry et al., 2017). Inductive coding, a "bottom-up" approach, derives codes (and resulting themes) from the data, allowing data to guide the identification of meaning and interpretation. Conversely, deductive coding leverages existing

<sup>1</sup>Department of Engineering Education, Virginia Tech, Blacksburg, VA, USA

## Corresponding Author:

Andrew Katz, Department of Engineering Education, Virginia Tech, 635 Prices Fork Road 340 Goodwin Hall, Blacksburg, VA 24061, USA.  
Email: [akatz4@vt.edu](mailto:akatz4@vt.edu)



Creative Commons Non Commercial CC BY-NC: This article is distributed under the terms of the Creative Commons Attribution-NonCommercial 4.0 License (<https://creativecommons.org/licenses/by-nc/4.0/>) which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (<https://us.sagepub.com/en-us/nam/open-access-at-sage>).

theoretical concepts or frameworks as a lens to interpret the data – a “top-down” approach. Inductive and deductive processes differ in the extent to which the data and existing theory are informing the researcher’s interpretation of the data and consequent contents of the codebook (i.e., collection of generated codes and their associated definitions). Nevertheless, both processes are subject to researcher subjectivity and positional bias, leading to the common suggestion that multiple researchers should analyze the same data (Miles et al., 2014; Terry et al., 2017).

Qualitative data hold substantial research potential given the density of the information they contain but entail costs and time-consuming processes (Miles et al., 2014). Fortunately, recent advancements in natural language processing (NLP), a subset of machine learning, integrate large language models (LLMs) to automate language understanding and synthesis (Crowston et al., 2010; Katz, Shakir, et al., 2023). These tools offer significant potential for reducing the cost, while enhancing the accuracy and efficiency of qualitative thematic analysis (Bhaduri, 2017; Garman et al., 2021). Current NLP applications leverage LLMs to analyze sentiment, synthesize text, design feedback systems, develop tutoring systems for computer education, and model stances in student essays (Chiu et al., 2023; Ganesh et al., 2022; Kastrati et al., 2021; Mathew et al., 2021; Persing & Ng, 2016; Shaik et al., 2022; Somers et al., 2021; Sunar & Khalid, 2024). These applications highlight the considerable potential for NLP to complement existing human-based qualitative data analysis approaches.

In this paper, we outline the development and utilization of an NLP-based method for inductive qualitative data analysis. We demonstrate this method on a set of 5000 student evaluations of teaching (SETs) and generate a collection of codes (codebook) that summarizes their recurring patterns and concepts. We show that this generated codebook mirrors the codebook produced from a previously completed traditional thematic analysis on a selection of the same dataset (M. Soledad et al., 2017; M. M. Soledad, 2019). While previous approaches have utilized NLP for deductive qualitative coding (Katz, Wei, et al., 2023; Tai et al., 2024), we present a novel method for inductive coding using a robust, iterative process that extracts, embeds, clusters, and summarizes qualitative data into a codebook akin to traditional inductive coding processes. This automated, data-secure method employs open-source LLMs, avoiding proprietary private models that may utilize user data for training (Gemini Apps Privacy Hub - Gemini Apps Help, n.d; Privacy Policy, 2023). Utilizing such open-source LLMs deployed locally on one’s computer also permits a more automated coding process without the need for user interfaces like chatbots. Consequently, we anticipate researchers can employ our approach to conduct meaningful qualitative analysis securely, efficiently, and expediently. When combined with researcher oversight, there is significant potential for this method to broaden existing qualitative data analysis approaches.

## Literature Review

This section delves into the use of NLP techniques for analyzing textual data. We explore the growing application of NLP in

education research, including for analyzing educational content, generating course materials, and assisting learning processes. Student evaluations of teaching, a form of such educational content, were used to evaluate our approach, which we also describe.

## NLP Use in Education

Machine learning (ML) is a growing area in research and industry, with significant potential to transform education through its pattern recognition and decision-making capabilities (Alpaydin, 2021; El Naqa & Murphy, 2015). The ability of machine learning models to analyze and interpret textual and unstructured datasets, like videos and audio, promises to revolutionize learning experiences, instruction methods, and educational systems (Katz, Shakir, et al., 2023; Shaik et al., 2022; Sunar & Khalid, 2024). Natural language processing, a sub-field of ML, uses algorithms to enable computers to understand, interpret, and generate human language data (Bhaduri, 2017; Chowdhary, 2020; Gillioz et al., 2020; Katz, Shakir, et al., 2023). Capabilities of NLP models expand beyond single words to analyze large swaths of text, allowing computers to better approximate contextual understanding, and making responses more coherent and contextually appropriate (Brown et al., 2020; Devlin et al., 2019; Vuong et al., 2021).

Recent advances have enhanced NLP models’ ability to generate text from input prompts and maintain coherence over longer sequences (Brown et al., 2020; Min et al., 2023), as seen in chatbots like ChatGPT (Ansari et al., 2023; Cooper, 2023). While chatbots represent a major advancement in NLP, they are part of a broader field that includes other techniques and approaches for processing and analyzing human language. Research in this area is diverse, encompassing a wide range of applications that utilize both the analytical and generative capacities of NLP (Bhaduri, 2017; Katz, Shakir, et al., 2023; Shaik et al., 2022; Sunar & Khalid, 2024). These applications include tasks such as feature extraction from text, generating summaries, and creating educational content.

When used for text analysis, NLP-generated features, summaries, and categories can serve as the foundation for further analyses. For example, NLP enables sentiment analysis, where NLP models identify the underlying tone of a text and determines whether it is positive, negative, or neutral (Crossley et al., 2018; Kastrati et al., 2021). Such analyses can then be used to gauge overall impressions and attitudes. While this can be useful for tracking changes in sentiment over time or across different groups, sentiment analysis may miss subtle or complex emotions (Wankhade et al., 2022). Further, sentiment is only a single dimension of a text, raising questions about how the detected sentiment interacts with the text’s greater context and meaning. Other approaches for NLP-based text analysis include topic modeling, like latent Dirichlet allocation (LDA), where topics are identified across text documents using word frequencies (Blei et al., 2003). While topic modeling can quickly process and identify latent

themes within large volumes of text, results may miss nuance and rich detail (Abram et al., 2020; Johri, 2023). Consequently, topic modeling has been used to assess trends in educational research where less-detailed topics are valuable (e.g., Chen et al., 2020; Johri et al., 2011; Ozyurt & Ayaz, 2022; Yun, 2020). Alternatives to traditional LDA topic modeling include transformer-based approaches like BERTopic (Grootendorst, 2022) which are more adept at general language understanding. While BERTopic and similar transformer-based approaches have been used for text analysis in educational settings (e.g., Bala & Mitchell, 2024; Najmani et al., 2023; Wang et al., 2023), they present standardized techniques that may not be optimized for unique datasets or custom applications.

In addition to text analysis, the text generation capabilities of NLP models have also found applications in education, such as in the design of tutoring systems, the development of educational content, and the creation of personalized learning experiences (Katz, Shakir, et al., 2023; Shaik et al., 2022). These adaptive systems generate personalized recommendations and feedback for users, dynamically adjusting the difficulty, level of support, and content based on individual needs. For example, NLP models have been used to generate feedback for student writing (Madnani et al., 2018; McNamara et al., 2013) and team performance (Sajadi et al., 2023). Systems analyze student responses and generate constructive feedback on various aspects, such as grammar, content, tone, and argumentation. NLP models can also simulate conversations, providing tutoring information about course material (Mathew et al., 2021; Paladines & Ramirez, 2020; Troussas et al., 2023). These systems can provide 24/7 support to students and be tailored to their individual needs and education level.

Given the existing uses of NLP for text analysis and generation, there is potential to leverage these techniques for highly time- and resource-intensive processes, like qualitative thematic analysis. Thematic analysis is a widely used qualitative research method for identifying, analyzing, and reporting patterns or themes within data (Miles et al., 2014). This flexible approach can be applied to various data sources, including interview transcripts, focus group discussions, open-ended survey responses, and other forms of textual data. Notably, many processes inherent to thematic analysis align with the capabilities of NLP (Berdanier et al., 2018; Katz, Shakir, et al., 2023). Previous studies have used NLP for deductive code application (Katz et al., 2021; Tai et al., 2024; Xiao et al., 2023), where models determine whether an existing code applies to a piece of text. However, the use of NLP in inductive qualitative analysis, where the model generates a codebook without relying on a pre-existing framework, has been limited (Katz et al., 2021).

The purpose of this paper is to introduce a method for generating a set of qualitative codes by using NLP techniques, showcased through its application on a dataset comprising written feedback from undergraduate students at a research-

focused university. In the next section, we detail these responses, known as student evaluations of teaching, and their significance in educational settings.

## Student Evaluations of Teaching

Many institutions facilitate SETs at the end of semesters (Marsh, 1984; Wachtel, 1998). These surveys usually consist of Likert-scale items that are meant to provide students with the opportunity to “rate” an instructor’s teaching effectiveness, as well as open-ended prompts meant to elicit students’ perspectives on their learning experiences, generating quantitative and qualitative data (Marsh, 1984; Sproule, 2000). Quantitative ratings, particularly those meant to measure overall teaching effectiveness, are commonly used by institutions as input into such administrative decisions as salary, promotion, and tenure (Abrami et al., 2007; Hornstein, 2017). Students, however, are more likely to engage in the evaluation process to share their perceptions about their learning experience in the class rather than how their ratings may influence administrative decisions, and they share these perspectives primarily through their responses to open-ended prompts (Hoel & Dahl, 2019). These prompts generate qualitative data that are challenging to comprehensively distill for instructors teaching large class sizes, which results in the underutilization of these data. That underutilization results in a disconnect between how SETs are used in practice and students’ motivation to engage in the process (Abrami et al., 2007; Hornstein, 2017). Evaluations of teaching present one use case for NLP. The following section describes the development of an NLP-based methodology for inductive codebook generation that can be used to analyze SET data more effectively, a process mirroring the initial code generation phases of traditional thematic analysis.

## Methods

In this paper, we present a workflow that involves extracting information, embedding the extracted information in a high-dimensional vector space, clustering those embeddings, and summarizing the clusters. The goal of our extract, embed, cluster, and summarize (EECS) workflow is to provide an easy-to-use and effective way to extract information from a large corpus of documents and distill it into a collection of codes. The process mimics common steps for codebook generation in traditional qualitative data analysis (Reyes et al., 2021) but on a larger scale. In this case, those documents were SETs from introductory science and engineering courses at a large R1 university. The data were acquired and used per VT IRB #17-432. Data were not anonymized further because we were using local, open-source language models and therefore adhered to standard protocols when handling data only within a research team. This setup meant that potentially sensitive data never left our local computers (and approved cloud-based storage platforms), which may be different from arrangements

other researchers use that involve sending data to third-party language model providers.

In certain research settings, it is common to have tens of thousands of SETs to analyze. The EECS workflow is designed to be scalable so it can be applied to a large corpus of documents beyond education settings. For demonstration purposes here we use a relatively small corpus of 5000 SETs, but in theory, the workflow can be applied to a much larger corpus, depending on computing resources and the nature of the textual units of analysis. To provide readers a sense of the length of the data, Figure 1 illustrates the distribution of the word count for each SET. The average length of a response was 22 words, and the median length was 14 words.

Figure 2 shows a flowchart of the EECS workflow. The workflow begins with the extraction of ideas from the original documents. The extracted ideas are embedded into a vector space and then clustered into groups of similar ideas. Next, those clusters are summarized to create a codebook, which is simplified to remove redundancies and improve usefulness. The overall workflow is designed to be modular so that each step can be modified or replaced with a different technique or language model as needed. Indeed, there is reason to believe that optimizing each step along the workflow would yield better results – a fruitful direction for future work. Finally, we have designed this workflow to use open-source models and tools for the broader research community’s use. These features allow researchers to run the EECS process on their own machine, thereby eliminating the need to send data to third-party providers, such as OpenAI or Google. The following sections describe each step in the EECS workflow in more detail.

### Step 1: Extracting Information

The first step in the EECS workflow is to extract the information from the documents we want to analyze. This step is intended to overcome the varied language found in respondents’ free form SET responses because excessive variation could cause larger problems in the downstream analysis. This extraction essentially is a standardization step and a way to isolate each of the ideas expressed in the SETs. In other words, we want to extract the main ideas from the SETs by disentangling them into simpler, discrete ideas.

For the extraction step, we used a lightweight generative text model with the prompt provided in the Appendix. The specific model was the dolphin fine-tuned version of Mistral-7b (Jiang et al., 2023). We used this language model because it is an open-source language model with a permissive license that is sufficiently small to run on common computer hardware at a fast generation rate. The size tradeoff is between larger models that may be more accurate but also may not fit on the average computer or may take too long to generate words. Since this task of summarization is relatively simple for the language models to complete, our initial tests suggested there was no advantage to using a larger generative text model (i.e., one with more billions of parameters). Regardless of the model used, the extraction allows us to analyze each idea separately in subsequent iterations. For example, if a student said, “I liked the in-class demonstrations and availability during office hours,” we would want to extract the two ideas “in-class demonstrations” and “availability during office hours.” The prompt is designed to be a general prompt that can be used for any type of document so that the same workflow can be used for many other contexts.

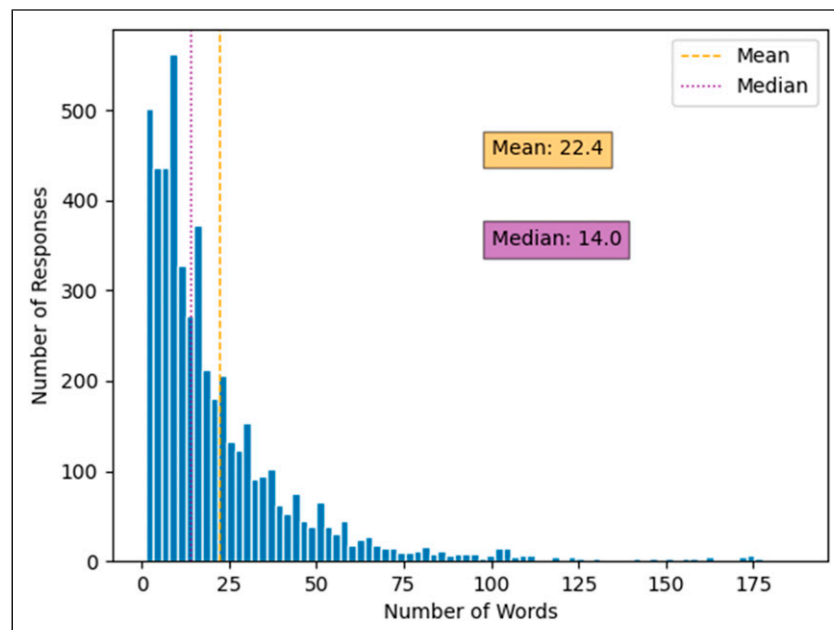


Figure 1. Word count histogram for raw responses.



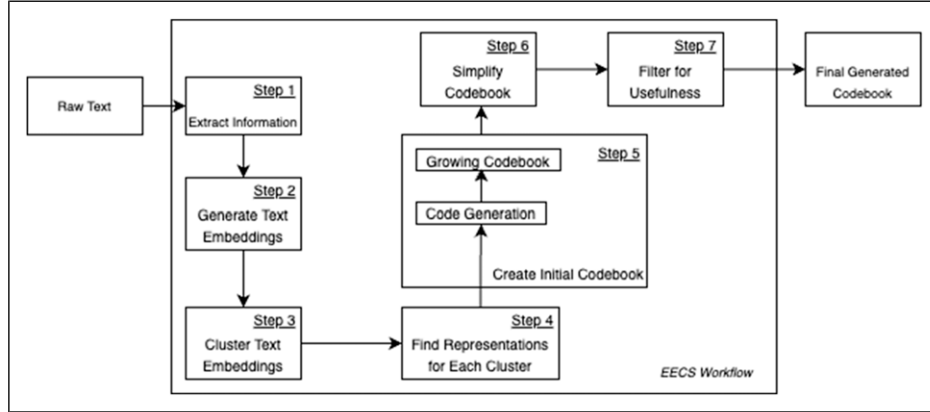


Figure 2. EECS workflow overview.

### Step 2: Generate Text Embeddings

The next step in the EECS workflow is to embed the extracted ideas into a vector space. By doing so, we aim to represent each idea as a vector so that ideas can be compared using mathematical operations, such as cosine similarity, or be clustered together to find semantically similar ideas. We used a pre-trained text embedding model available from Hugging-Face (Wolf et al., 2020) to embed the ideas. The specific model used here was the UAE-Angle model (Li & Li, 2023), but other models would also work in this step. We used that model because it had the highest performance across a range of performance benchmarks on the Massive Text Embedding Benchmark leaderboard (Muennighoff et al., 2022) at the time among open-source models that could be run locally on one’s machine. This ability to run the model on common computer hardware was important to make the workflow more feasible for researchers without needing special equipment or sending data to third-party vendors.

The embedding model uses the text for each extracted idea as inputs and outputs a vector. The vectors from this UAE-Angle model are 1024-dimensional, representing the text for an idea as a vector (i.e., collection of numbers) of length 1024. The goal behind embedding models is to leverage the theory of distributional semantics, which asserts that words that appear in similar contexts have similar meanings (Boleda, 2020; Erk, 2016). Using distributive semantics, we can represent the semantic meaning of words and phrases as vectors. If two different phrases are semantically similar, as in “used realistic examples” and “gave real-world scenarios” then the vector representations of these should be close to each other in that high-dimensional vector space even though the two strings share no words in common. Working in this vector space allows us to compare the meaning of words and phrases using mathematical operations such as cosine similarity. Cosine similarity measures the angle between two vectors in a high-dimensional vector space (Kenter & De Rijke, 2015). In text analysis, we can use cosine similarity to compare the meaning of two phrases. If the cosine similarity between two phrases is

high, then the meaning of the two phrases is similar. This contrasts with older text analysis methods, such as bag-of-words, which do not capture the semantic meaning of words and phrases but rely on the presence or absence of words (Khurana et al., 2023). In that setting, “used realistic examples” and “gave real-world scenarios” would be treated as completely different phrases because they have different words, even though they are semantically similar.

In addition to the ability to calculate semantic similarity between pairs of texts, the embedding model also allows us to calculate the similarity between a single text and a set of texts, as one does when using clustering algorithms. Indeed, clustering is the next step in the EECS workflow because once we have the vector representations of the extracted ideas, we have a new problem: how do we find similar ideas so that we can reduce the information required to represent the SETs into something far smaller, that is, on the order of 50–100 pieces of information/ideas? An alternative way to frame this problem is similar to principal component analysis and the notion of finding the dimensions that explain the most variance in a dataset. In this case, we want to find the dimensions that explain the most variance in the text of extracted ideas. Clustering can help us move in that direction by finding groups of similar ideas.

### Step 3: Clustering Text Embeddings

The next step in the EECS workflow is to cluster the extracted ideas. This is important because there are many instances of students commenting on similar ideas, for example, “great notes” or “the notes added online were very helpful.” To find the unique themes within the SETs, the extracted ideas must be collapsed into a more manageable number of observations. This can be accomplished using a clustering algorithm, which groups similar ideas. We used the hierarchical density-based spatial for applications with noise (HDBSCAN) algorithm, which was implemented using the hdbscan Python library (McInnes et al., 2017). We chose HDBSCAN because it is capable of labeling data points as noise if they do not fit within

a cluster. This ability is in contrast with other hard clustering algorithms that will force data points into clusters even when their fit is questionable. While clustering, we used conservative parameter settings to encourage many small clusters. The goal was to have many clusters, each with a few ideas (i.e., ideally one). This was done to ensure that when we summarize the clusters later, we have a small number of ideas per cluster to summarize.

#### *Step 4: Finding Representatives for Each Cluster*

With the clusters created, we now want to find a single representative member to pass to the next step in the workflow. For example, if we clustered 5000 pieces of extracted information and formed 400 clusters, then we want 400 representatives - one for each cluster - to pass to the generative model. The goal is to have a more manageable quantity of information for the model to remember in its prompt. In essence, this step compresses 5000 pieces of information to 400, assuming each cluster contains a large amount of redundant information. Finding a representative reduces this redundancy without losing information.

To identify a representative, we first embedded a concatenated version of the ideas in each cluster. For example, if a cluster contained entries such as “helpful office hours,” “lots of office hours,” and “good office hours,” we could combine them into a single string like “helpful office hours lots of office hours good office hours,” that we can then embed. These cluster embeddings were then compared with the embeddings for the original extracted information to find the most similar piece of original extracted information. We used the same embedding model as in the prior embedding step and calculated the cosine similarity between the embeddings for each cluster and the embeddings for the original extracted information.

Put another way, this representative identification step selects the extracted information with the highest cosine similarity to the cluster embedding. This “most similar” idea is assigned as the representative of that cluster. Let’s imagine that the 295<sup>th</sup> cluster had grouped six pieces of extracted information. We concatenate those six ideas into a single string, which we then embed. We then calculate the cosine similarity between the embedding for that string and the embeddings for the extracted information from all the 5000 original documents. Finally, we select the extracted information that had the highest cosine similarity with the cluster embedding and assign it as the cluster’s representative. That representative is passed to the next step in the EECS workflow, initial codebook creation.

#### *Step 5: Create Initial Codebook*

Two more steps remain in the EECS workflow: creating the initial codebook and simplifying it to remove redundancies and improve its utility. In the preceding step, we found

representatives for each cluster that encapsulated the main cluster themes. Now, the representatives are sequentially given to a generative text model with instructions to summarize it. The aim is to create an initial qualitative codebook with the most common recurring topics in the SETs. This codebook can then be simplified and used in downstream analyses to understand the prevalence of the codes in the SETs. The codebook application step is beyond the scope of this paper and will be described in future work. However, to generate the codebook, we can use the prompt given in the [Appendix](#).

In response to the prompt (detailed in the [Appendix](#)), the model generates a summary of the cluster with a code, definition, and example if it determines that a new entry in the codebook is necessary. We subsequently extract the generated text in a short post-processing step. Ultimately, the text generated after the prompt serves as the codebook entry, which is utilized in the subsequent simplification step. Despite its complexity, the process is conceptually straightforward: create a codebook by examining a representative sample of extracted ideas, while striving to keep it concise by adding new entries only when essential. This necessity is determined by evaluating the existing content in the codebook. In essence, the EECS workflow mirrors traditional qualitative data analysis practices.

#### *Step 6: Simplifying the Codebook*

The concluding step in the EECS workflow involves the simplification of the codebook. This step is crucial as the initial codebook generated can often be lengthy and redundant, potentially complicating its utility in subsequent analyses. The aim is to streamline the codebook for easier application in downstream processes.

This final step is part of a retrieval-augmented generation (RAG) pipeline. One challenge in generating a codebook is that excessive entries are found, with many being redundant. Consequently, the prompt included instructions for the model to reference an existing codebook (i.e., the one we are inductively creating), and decide whether the extant codes cover the theme(s) in the new text under analysis. If not, a new code entry is warranted because the existing codes are insufficient.

While it might seem logical to provide the entire codebook to the model as part of the prompt, our practical experience has shown that it is more effective to offer the top  $k$  codes from the codebook. By “top  $k$  codes,” we refer to the  $k$  most similar codes based on semantic similarity between the cluster text and existing codes. This approach reduces the memory requirements on the model; rather than memorizing the entire codebook, the model only needs to recall the top  $k$  codes that we have provided it in the prompt. This is a tradeoff because the model may not always make an informed decision about the necessity of a new code if it only sees the top  $k$  codes. However, in our experiments, the model demonstrated the ability to make such informed decisions, as evidenced by the reasoning steps it provided in the output.

To determine the top  $k$  codes, we embed each entry in the codebook using the same embedding model as in the prior steps and calculate the cosine similarity between the embeddings for each codebook entry and the embeddings for the cluster representative. Subsequently, we select the top  $k$  entries in the codebook with the highest cosine similarity with the cluster representative embedding as the top  $k$  codes. For instance, if the new cluster representative is “gave lots of examples in class,” the top  $k$  codes might include “in-class demonstrations,” “real-life scenarios,” and “helpful lectures.” These codes are then included in the prompt provided to the model. As with some of the other steps, we employ a generative text model and prompt outlined in the [Appendix](#). The design of this prompt ensures its adaptability for diverse document types, enabling the use of the same generative text model across various contexts.

## Results

### Extracting Ideas from the Original Texts

After pre-processing and screening short responses from our random sample of 5000 SETs, we ended up with 4672 unique original SETs. Utilizing the EECS workflow, we initially extracted 12,046 ideas from this set of 4672 original comments. The distribution of the extracted ideas is shown in [Figure 3](#). The average number of extracted ideas was 3.3 with a median value of 3. The right skew of the distribution was expected because most students only have a few things to say while a few outliers have much more that they want to share.

[Table 1](#) shows an example of the extracted ideas from a single SET. In this example, two ideas were extracted: the instructor provided examples, and the instructor explained the problem-solving process.

### Summarizing the Clusters

We clustered the 12,046 extracted pieces of information into 272 clusters. Summarizing the 272 clusters yielded a codebook with 232 codes (some of which were redundant). An example of the input and output for this step is given in [Table 2](#). Note that this step ran with two slightly different prompts. For the first  $k$  steps, the process ran without a RAG implementation since there were not enough codes in the codebook for the RAG part of the prompt. After the first  $k$  iterations of the process, we then used the prompt in the [Appendix](#), which included the top  $k$  codes from the codebook as the RAG implementation. Doing this split of the prompts ensured there were enough entries in the codebook to start using for the nearest-neighbor matching part of the RAG process.

### Simplifying the Codebook

From the 232 codes generated in the previous step, we simplified the codebook to 159 codes. This simplification

proceeded in two steps. The first was a RAG implementation that asked the model to check each code and for a similar one already in the codebook. An example of that output is given in [Table 3](#).

The second step in the codebook simplification process was a check on whether the code was sufficiently clear. An example of the input and output for this step is given in [Table 4](#). In the example, the original code “encouraging active participation” was reviewed and the model suggested keeping the code because it was specific and clear enough for its intended purpose.

An example of an alternative suggestion from the model is given in [Table 5](#). In this example, the model was given the code “clarification and simplification” and suggested discarding this code and creating two new codes – “clarification” and “simplification” – to better understand how students responded to feedback forms.

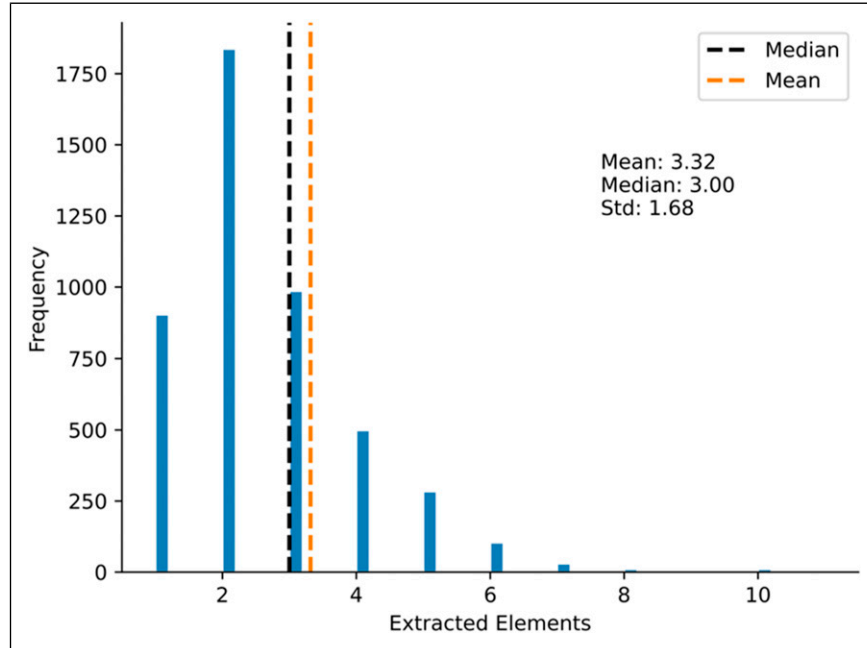
### Comparison with the Human-Generated Codebook

As previously mentioned, a subset of SET data was originally employed in prior studies investigating the teaching and learning dynamics within first-year engineering courses (M. Soledad et al., 2017; M. M. Soledad, 2019). These SETs underwent manual thematic coding, revealing 39 distinct codes.

Established frameworks provide one way of synthesizing responses to open-ended responses in SETs (Creswell, 2014). Soledad (M. Soledad et al., 2017) leveraged the Academic Plan Model (Lattuca & Stark, 2009) for this purpose, which acknowledges the crucial role of students’ perspectives in curriculum planning. The APM has informed prior work investigating students’ learning experiences and instructors’ curricular decision-making processes (e.g., (D. Knight et al., 2012; D. B. Knight et al., 2016)). It provides a structure for categorizing and presenting qualitative data on students’ learning experiences and their perspectives on the learning environment. According to the APM, the learning environment consists of eight elements, described in [Table 6](#).

The initial coding revealed that the analyzed SETs corresponded with these APM elements. However, the 39 human-generated codes fell into only four categories from the APM: one in *Adjustments*, one in *Assessments*, 13 in *Faculty characteristics*, 18 in *Instructional processes*, and six in *Instructional resources*.

In contrast, our EECS method generated 80 distinct codes. Five were excluded due to ambiguity or vagueness, yielding 75 useable codes, of which six pertained to *Adjustments*, five to *Assessments*, 22 to *Faculty characteristics*, 25 to *Instructional processes*, 15 to *Instructional resources*, and two to *Content and Sequence* — a category not initially identified through manual coding. *Content and Sequence* refers to the arrangement and content of subject matter in a course. This difference between the human coding and the EECS process may indicate that the larger



**Figure 3.** Histogram of the number of extracted ideas from the original SET comments.

**Table 1.** Extracted Ideas From a Single SET.

Student's text	Model response
She did examples and explained how problems were done	My summary: 1. Provided examples 2. Explained problem-solving process

**Table 2.** Example of Summarizing the Clusters.

Prompt	Model response
<b>Input</b> The instructor used powerpoint presentations effectively	<b>Output</b> Label: Effective PowerPoint use Definition: The use of well-designed and engaging PowerPoint slides by instructors. Example: "The instructor used powerpoint presentations effectively."

**Table 3.** Example of Step 1 for Code Consideration.

Input	Output
Quick email response	<p>Step 1: Examine existing codebook. ["Insufficient detail," "well prepared organization," "lack of action suggestions," "feedback quality," "response speed"]</p> <p>Step 2: Read new written response in &lt;text&gt;. "Quick email response time."</p> <p>Step 3: Determine if there is an existing label to describe the text. The given text discusses "email response time," which seems related to "response speed." Therefore, I will use this code from the existing codebook</p> <p>Step 4: If no suitable label exists in the existing codebook, generate a new one and provide its definition and an example from the &lt;text&gt; tag. Since there is already a relevant label ("response speed"), no new label needs to be suggested. No new label suggested</p>



**Table 4.** Example of the Simplification Step With Accepted Original Code.

Input (code)	Output (judgement)
Encouraging active participation	<p>My response: The given code, “encouraging active participation,” seems like a good code as it specifically addresses an aspect of teaching that instructors might be interested in receiving feedback on from their students. It is clear and informative for the research team to understand what kind of feedback they are looking at when analyzing responses related to this theme</p> <p>My suggestion: Keep the code</p> <p>Alternate suggestion: None, as the current code is already specific and clear enough for its intended purpose</p>

**Table 5.** Example of the Simplification Step With an Alternative Suggestion.

Input (code)	Output (judgement)
Clarification and simplification	<p>My response: The given code, “clarification and simplification,” seems to be addressing two distinct aspects related to feedback forms – making instructions or questions clearer (clarification) and presenting them in a more straightforward manner (simplification). While both are important for improving the quality of responses from students, they can also be considered as separate codes. This would allow researchers to better understand which aspect is being addressed by each theme</p> <p>My suggestion: Discard this code and create two new codes – “clarification” and “simplification.” By separating these themes into distinct codes, the research team will have a more precise understanding of how students are responding to feedback forms</p> <p>This can help identify specific areas for improvement in future iterations of the form design process</p> <p>Alternate suggestion:</p> <ul style="list-style-type: none"> <li>- Clarification: Refers to making instructions or questions on feedback forms clearer and easier to understand.</li> <li>- Simplification: Addresses presenting information on feedback forms in a more straightforward manner, reducing complexity and potential confusion for students</li> </ul>

**Table 6.** Elements of the Academic Plan Model Adapted From [Soledad \(2019\)](#).

Element	Description
Adjustments	Enhancements and adjustments to the academic plan based on experience, evaluations, and student feedback
Assessment	The strategies used to determine whether decisions about elements of the academic plan are optimal
Faculty characteristics	Perceptions about the availability, approachability, and focus of instructors
Instructional processes	The instructional activities by which learning may be achieved, such as scaffolding, assessment practices, and in-class problem descriptions
Instructional resources	The materials, technologies settings used in the learning process, and the opportunities to engage with the instructor outside of class time
Content and sequence	The arrangement and content of subject matter toward achieving learning outcomes

number of SETs processed contain information beyond the smaller initial sample used for the human coding – a promising finding for similarly large datasets. The complete list of EECS-generated codes alongside the human-generated codes is provided in Tables 7, 8, and 9 in the [Appendix](#).

Comparing the model’s codes with the APM demonstrates the method’s alignment with traditional human-generated codes. Because the human-generated codebook is aligned with the APM elements, we should expect the model’s codes to align. Once categorized according to the APM, the human- and model-generated codes are more meaningful side-by-side, allowing for a clearer comparison.

Examining the distribution of codes across categories, our method mirrors a similar trend to the manual analysis. Notably, the model’s codes were categorized independently of the original codebook and were cross-checked by someone familiar with the initial coding. Furthermore, there was substantial overlap between the original codes and model-generated codes within each category, indicating that the model effectively captured analogous topics, terms, and characteristics. Following the final simplification step, the model’s codes were largely constructive and informative, facilitating their integration into subsequent analyses or feedback for instructors.

Moreover, the significant concordance between the method's codes and those manually derived indicates the workflow's capacity to encapsulate the essence of the analyzed SETs. This finding holds promising implications for the research community, as it suggests that inductive thematic analyses can be efficiently conducted on extensive datasets using the EECS workflow within a relatively short timeframe. These implications will be further explored in the following section.

## Discussion

### *Comparison to Traditional Codebook Generation*

These results underscore several key insights. Namely, the EECS workflow effectively replicates the codebook generation process typical of traditional qualitative data analysis. Not only did this process reproduce codes identified in traditional qualitative analysis, but it also identified new codes with greater granularity than those in the original codebook. This increased granularity may be due to the workflow's ability to detect subtler distinctions in the data than a human coder might perceive, or because the workflow evaluated a significantly larger volume of data. The EECS workflow processed over 4500 unique SETs, far exceeding the scope of the original analyses, which likely contributed to the identification of new codes as patterns that were barely noticeable in the original analyses became detectable with more data.

When compared to traditional qualitative methods, the EECS workflow offers notable advantages, including the ability to efficiently process and analyze larger datasets. This scalability comes at reduced costs because larger datasets can be evaluated faster and with fewer resources than comparable human-based approaches. This large-scale textual analysis can reveal patterns that might be overlooked in smaller datasets analyzed by human coders. Consequently, a promising avenue for future research is to explore how this workflow can apply to longer texts in less constrained environments. For example, it could be used to analyze student essays, research articles, and administrative records where connections and relationships can be drawn between extensive textual data.

Despite this promising opportunity, there are drawbacks to consider. Although models may be able to detect subtle differences, they may lack a nuanced understanding of context and meaning known by a human coder. Some of these aspects may be captured by the prompts provided to models, such as the data type, data collection methods, or study objectives, but human coders can interpret the intricacies of language, cultural references, and underlying sentiments in ways that current models might not fully grasp. For instance, irony, sarcasm, or complex emotional tones may be challenging for models to accurately identify without specific training (Potamias et al., 2020; Weitzel et al., 2016). Additionally, models may introduce biases embedded in their training data and underlying algorithms (Mao et al., 2023; Sheng et al., 2019; Vig et al.,

2020). While human coders can reflect on and adjust for potential biases during analysis, automated models may propagate these biases unless carefully monitored and calibrated. This propagation extends to biases within the data corpus itself that could be reflected in the outcomes generated by the EECS workflow. For instance, if the data were not representative of the broader population, the codes and themes identified might be similarly skewed, leading to the overrepresentation or underrepresentation of certain themes. Furthermore, short or ambiguous text analyzed by the workflow could inject uncertainty into the resulting outputs. This could also occur if the texts were too similar, leading to the over-interpretation of minor variations that might not be meaningful in a broader context. Given these considerations, while the EECS workflow provides a powerful tool, it may be most effective when used in conjunction with human oversight, ensuring that the richness, depth, and accuracy of qualitative analysis are preserved.

### *Comparison to Other NLP Methods*

While the EECS workflow successfully replicated and extend human coding efforts, it is crucial to consider its performance in comparison with other popular approaches to computer-assisted thematic analysis. One popular method for analyzing text data is topic modeling. Topic models are statistical models used to identify topics in a collection of documents. One of the most popular topic modeling algorithms is latent Dirichlet allocation (LDA) (Blei et al., 2003). The LDA model is a generative probabilistic model used to discover topics within a collection of documents. LDA assumes that each document is a mixture of topics that are represented as a group of words or related words. These topics, or themes, exist across multiple documents, and each word in a document is assumed to be generated by one of its topics. The goal of an LDA model is similar to the goal in the EECS workflow: identify the latent topics related to the observed words in each unit of analysis (e.g., document, essay, article, SET). In practice, LDA works backward from the observed words in the documents to uncover the latent topic structures. One advantage of this is quick training – LDA models can be trained in a matter of minutes. In one application, Abram et al. (2020) used LDA to analyze nearly 170,000 words from nurse interviews and found it reduced the project time by at least 120 hours and costs by \$1500. This analysis was completed in 2 minutes, not including the time required to process the data and write the program. By contrast, the EECS workflow can take hours to complete depending on corpus size and computational resources. Furthermore, while LDA possesses specific metrics to determine the number of topics to extract from the data (Wallach et al., 2009), the EECS workflow does not currently, and instead relies on the user to determine the number of clusters to extract from the data.

On the other hand, the EECS workflow relies more on semantics than syntax. Typical topic models rely on word stems, so

words like “homework” and “assignments” are treated differently even though they may be similar in some contexts. In the EECS workflow, this could be remedied using prompts that specify the context or during clustering. Additionally, traditional topic models provide a list of frequent words for each topic, leaving researchers to interpret their meaning. This could create ambiguity and divination in their results. By contrast, the EECS workflow generates coherent labels and accompanying definitions. To illustrate, [Abram et al. \(2020\)](#) noted that their method captured the overall thematic descriptions in the interviews but did not provide rich, nuanced themes mirroring those from traditional qualitative methods. Had the EECS workflow been used, the processes could capture semantic relationships and contextual details better than word frequency-based approaches, like theirs. The text embeddings used in the workflow, when used alongside more advanced clustering algorithms, could identify more subtle thematic groupings that better reflect the meaning in the data, while including elements of the broader context.

Modern variations of topic models such as BERTopic ([Grootendorst, 2022](#)) attempt to address limitations of traditional topic models, to varying degrees of success. BERTopic harnesses the power of transformer-based models, the artificial neural network architecture that has revolutionized many NLP tasks ([Gillioz et al., 2020](#)). When used with the same SET dataset, BERTopic and LDA produced less discernible topics, and the coherence scores also suggested fewer topics. Interpretability and granularity were two areas where EECS was more useful than BERTopic. On the other hand, BERTopic is a more widely available and better-known approach, and topic modeling is quicker to run. Nonetheless, the reliance on syntactic similarity requires significant assistance from transformer-based models to try to capture semantic similarities.

Transformers, introduced by [Vaswani et al. \(2017\)](#), have become the foundation of state-of-the-art NLP models due to their ability to capture the long-range dependencies in sentences ([Gillioz et al., 2020](#)). By weighing the importance of different words in a sequence and processing data in parallel, transformers can learn intricate patterns in language, making them invaluable for NLP applications ([Gillioz et al., 2020](#); [Katz, Shakir, et al., 2023](#)). Our method, like BERTopic, employs transformer-based models for the codebook generation process. However, we are not unique in recognizing the potential of transformers for NLP-based qualitative thematic analysis. [Ganesh et al. \(2022\)](#) introduced the response construct tagging (RCT) task to analyze students’ open-ended responses from course surveys, focusing on identifying affective states related to transformative experiences and engineering identity development. Their work showcased the scalability and value of transformer models like RoBERTa ([Liu et al., 2019](#)) for qualitative coding in educational research contexts.

While these investigations demonstrated NLP’s supportive role in traditional thematic analysis, [Tai et al. \(2024\)](#) proposed a novel methodology using LLMs for deductive coding in qualitative research. Parallel to our research group’s work

([Gamieldien et al., 2023](#); [Katz et al., 2021](#)), they developed a deductive coding scheme with five predefined codes related to scientific identity and input sample interview excerpts and the codebook into ChatGPT to detect code presence and supporting evidence. Comparing the results to human coding, the authors found high alignment for most codes. Interestingly, areas of divergence prompted refinements in code definitions and applications, showcasing how LLMs can serve as external reviewers to identify potential blind spots, a finding we also observed. Transformer-based NLP techniques have shown potential in assisting and guiding qualitative thematic analysis, but research thus far has been limited to a few approaches revolving around ChatGPT and has not ventured far into inductive thematic analysis as our method proposes nor the use of open-source models.

Accordingly, there are several potential reasons our inductive process may work. To start, the initial information extraction step helps to standardize language and disentangle complex ideas into simpler, singular ideas. This is important because it allows us to analyze each idea separately in subsequent steps. Without this step, we would have to analyze the entire SET at once, which would be a much more difficult task. Additionally, the embedding step further helps funnel information into standardized formats because it captures the semantic meaning of words and phrases. Therefore, we have a way to move from the varied original text into an abstract space of ideas, which prior methods using dictionary-based approaches were not able to do. Also, the dimension reduction and clustering steps help to further compress the information more than prior steps. The result is a funnel that takes in a large amount of varied information and outputs a small number of themes.

### *Humans-in-the-Loop*

While our methodology is largely automated and programmatic, human involvement is indispensable. Human expertise and judgment play a pivotal role throughout the process, ensuring the credibility of the analysis and validating the outputs. For instance, our method generated 80 total codes, but five were discarded due to ambiguity or irrelevance. The EECS process yielded a codebook summarizing recurring statements and content of the input data, yet it may include themes and content that are uninformative or misaligned with research objectives. We advocate for researchers to actively engage in the codebook generation process, identifying irrelevant or misleading codes, manually reviewing and rejecting unnecessary ones, and verifying the overall quality and validity of the automated analysis.

Code quality is tightly linked to its usefulness. While the EECS process generates a codebook, the workflow by default is agnostic about how codes will be utilized – a result of embedding, clustering, and prompt designs. Hence, researchers must assess the relevance and applicability of generated codes to research questions, considering the required level of granularity and specificity for meaningful analysis. Establishing criteria for determining the saturation point in code generation is advised, as this varies between

investigations. Researchers must monitor the codebook to identify when no new codes or insights emerge from the data. Balancing code granularity and manageability is crucial, as codes become simpler with each iteration. Consideration for potential consolidation or hierarchical organization during code review can help conclude the generation process while maintaining an ideal level of granularity.

The prompts used in the EECS process are detailed in the [Appendix](#) and are specifically tailored for this application. If applying this process to a different context, crafting context-specific prompts is essential to guide the generative text model effectively. Prompt design should balance specificity and flexibility to avoid scope limitations or researcher biases ([Mao et al., 2023](#); [Sheng et al., 2019](#)). Additionally, there is potential for this method to be adapted to fine-tuned LLMs trained on domain-specific datasets, potentially enhancing accuracy, relevance, and efficiency ([Church et al., 2021](#)).

Ultimately, to maximize the utility of the EECS workflow, researchers should actively participate in the codebook generation process, from assessing code quality and utility to designing prompts and selecting models. This method is inherently reflective, requiring researchers to play an active role in code generation and analysis rather than passively relying on method outputs.

## Limitations

The current workflow presents several limitations, some of which suggest areas for future exploration. First, granularity levels within the codebook pose a challenge. Instances such as having separate codes for “in-class demonstrations” and “used experiments to show ideas” may exhibit semantic similarity, suggesting potential consolidation into a single code. Similarly, codes like “clarity” and “clarity on syllabus” could be collapsed or treated hierarchically, especially if clarity is evident in other aspects of the course. Addressing this challenge is a focus for future work, necessitating the development of methods to systematically handle semantically similar codes, including refining techniques for consolidation and hierarchical treatment.

Second, the efficacy of codebook generation heavily relies on the language models employed in the process. As models evolve, ongoing evaluation of the method’s efficacy and accuracy, particularly with diverse datasets, is essential. Variations in model size and fine-tuning techniques can significantly impact outcomes, requiring thorough evaluation across different models and datasets to ensure robustness and reliability.

Third, the interpretation of codes exhibits context dependency. For instance, the interpretation of “scaffolding” may vary when analyzing SET data compared to comments from a construction site. Exploring methods to incorporate context into the codebook generation process, whether through prompt adjustments or alternative approaches, is an avenue for future investigation.

Finally, while the process mimics human coding efforts, its superiority over alternative methodologies remains uncertain. Alternative approaches may exist, utilizing different prompts, variations in the RAG pipeline implementation, or alternative

methods of codebook simplification. Moreover, advancements in model architectures, such as selective state space models, may mitigate context window limitations and introduce novel possibilities. While our work demonstrates the potential of using generative text models and modern NLP techniques with open-source options to replicate human coding processes efficiently and at scale, there are diverse avenues for exploration in designing processes for document analysis. We encourage further research to build upon this foundation, facilitating a range of options for document analysis across various fields.

## Conclusion

Recent advances in NLP and LLMs promise opportunities to analyze text data in new ways. In this paper, we have presented a new analytic method for such analysis. By combining information extraction, text embedding, clustering, and summarizing, we have shown that it is possible to identify themes in a large corpus of documents without the typical time and labor costs associated with manual coding. Moreover, we have demonstrated how to accomplish this using open-source tools, thereby avoiding sending data to third-party service providers and proprietary models. The process we have described is scalable and can be applied to create a codebook that can be used in downstream analyses. We believe that this method has the potential to be useful for analyzing SETs and other types of writing in teaching and research settings. We also believe that this method has the potential to be useful for analyzing other types of documents, such as essays, research articles, and administrative records. The goal is for this paper to inspire others to explore the possibilities of NLP and LLMs for qualitative analysis in their fields.

## Appendix

Note that the curly brackets ({} ) are placeholders for text to be inserted in the prompt.

### Prompt for Information Extraction

You are an expert text analyst reading {data type}s collected in {data collection context}. I am going to send you one of these {data type}s. I need you to use your expertise to analyze the provided text in the <text> tag below and summarize it in an enumerated list. You should do this analysis by providing several short descriptive phrases that summarize each idea discussed in the {data type} that answered the prompt. When you suggest multiple items, separate each one in your response with a new line. You MUST remove anyone’s names and \*use gender neutral pronouns\* for deidentification purposes. Start your response with “My summary:”. Here is an example of input and desired output from a different context when there are only two topics,

but remember that you can suggest as many topics as you think are necessary for the text you summarize.

Example input: “Jared did a great job responding quickly to emails and turning in good work.”

Example output: “My summary:

1. Responded quickly to emails
2. Turned in good work.”

Notice how the main ideas are summarized and there are no names or pronouns included here. Also, notice how the response did not make up information that was not in the input. You must NEVER make up information that is not in the input text you receive because there is a severe penalty for that. If the text you receive is very short and says “nothing,” do not make up new things.

Here is the text for you to summarize: `<text>{text}</text>`

Begin your analysis now.

### *Prompt for Summarizing the Clusters*

Act as if you are {persona}. You specialize in creating short labels to summarize the main idea being discussed in a collection of text. The labels you generate will be used to systematically label all the {data type}s to identify key themes in an entire dataset of text. I need your help analyzing a cluster of {data type}s from {data collection context}. In theory, the cluster of {data type}s should be semantically similar and related to each other, but occasionally it may not be homogenous. Your task is to generate a small number of labels that capture the main concepts or themes discussed in the majority of the {data type}s you review. To do this, please start by reviewing the {data type}s provided to you in the `<text>` tag. Each of the {data type}s is separated by a new line. After reading the {data type}s in the cluster, please then generate your summary labels. For each label you generate, please provide:

- The label (a short phrase)
- A brief definition of what the label represents
- An example of a {data type} from the actual text in the `<text>` tag

The goal is to capture the essence and meaning of the \*majority\* of comments in each cluster of {data type}s through the labels applied. Your response should start “Label:” followed by your label and then on a new line “Definition:” followed by your definition for that label.

For example:

Label: `<label 1>`

Definition: `<definition 1>`

Example: `<example 1>`

As a reminder, here is the cluster of {data type}s for you to analyze:

### *Prompt for Editing the Codebook*

Act as if you are the world’s best qualitative data analyst with expertise in generating qualitative codebooks for thematic analysis. Your task is to do the following four steps. Step 1: examine an existing codebook given to you in the `<existing codebook>` tag. Step 2: read a new code given to you in the `<text>` tag. Step 3: determine whether the existing codebook already has enough labels to describe the possible new code in the `<text>` tag. Step 4: Either accept the new code because it is sufficiently different from other codes or reject the new code because it is redundant. This is admittedly subjective, but you should be able to use your expert judgment to decide whether or not existing codes together provide sufficient coverage at an abstract level. Either way, be sure to provide your reasoning for whether or not you think you need to accept or reject the possible new code. Please note that there is a penalty for adding redundant codes, so you should only create a new code if you are certain the existing ones are insufficient. You should start your response with “My analysis:” followed by your step-by-step reasoning. You should conclude your response with “My final verdict:” followed by “Keep code `<code>`” or “Reject code `<code>`” where `<code>` is a placeholder for the actual label you are judging. For your reference, here is the existing codebook `< existing codebook>{codes}</existing codebook>`.

As before, note that “{codes}” is a placeholder where the top k codes are placed for each. With this prompt, the model then generates a summary of the cluster with a code, definition, and example.

### *Prompt for Simplifying the Codebook*

Act as if you are the world’s best social science researcher. You specialize in qualitative data analysis and thematic coding. I need your help. I have a list of codes in a codebook to label themes in {data type}s collected from a {data collection context}. These codes need to be useful and informative for the research team, meaning they need to be specific and clear. Your task is to read a code from the codebook that I will give you in the `<text>` tag and tell me whether or not the code is a good code that I should keep in the codebook or a bad code that I should discard. Format your response as “My response:” followed by your step-by-step reasoning about whether it is a good code. Then, on a new line, say “My suggestion:” followed by your suggestion for whether or not to keep the code. If you think we should discard the code then you should add yet another line that says “Alternate suggestion:” followed by a better version of the code you reviewed. Here is the code for you to review: `<text>zoom meeting requirement</text>`



## EECS Versus Manual Code Comparison

**Table 7.** EECS Versus Manual Code Comparison (Part 1).

Adjustments	Adjustments (EECS)	Assessment	Assessments (EECS)
Student feedback-based adjustments	Improved understanding Course improvement recommendations Student experience and satisfaction Course feedback and dissatisfaction Positive feedback Feedback quality (timeliness & satisfaction)	Emphasis on conceptual understanding	Comprehensive resource assessment Content understanding Conceptual difficulties Comprehensive content coverage Time constraints and preparation issues

**Table 8.** EECS Versus Manual Code Comparison (Part 2).

Faculty characteristics	Faculty characteristics (EECS)	Instructional processes	Instructional processes (EECS)
Being accessible	Teaching quality and effectiveness	Behavior, classroom engagement	Engineering problem-solving strategies
"Best professor"	Motivational factors	Online utilities	Engaging presentation style and note taking
"Organized"	Instructor support, responsiveness, and appreciation	Work-out problems and examples	Effective use of visual aids and note-taking effectiveness
"Presented material very clearly"	Clarity	Assessment practices	Task preparation
"Well-prepared"	Simplification	Clarify prior misconceptions	Detailed explanation
Behavior, classroom engagement	Instructor-student interactions	Demonstrate structures and processes to solve work-out problems	Mathematical explanation
Being accessible	Classroom environment management	Emphasize and discuss core concepts	Student engagement
Emphasize and discuss core concepts	Support for struggling students	Homework	Classroom interaction & participation
Foster student success	Support and assistance	Instructor-provided notes	Active learning strategies
General perception of teaching strategy	Positive impact and accessibility	Multimedia	Lab experiences & adoption
Provide guidance and feedback	Flexibility in scheduling	Office hours	Supervision & safety procedures
Show concern and interest in students' lives	Consistency in schedule	Practical applications	Structured lessons & teaching methods
Time management	Public humiliation Inclusive atmosphere	Provide guidance and feedback Real-world applications	Practical application and feasibility Value of examples and demonstration
	Effective communication	Review sessions	Real-life applicability and relevance
	Clear communication Communication strategies and styles	Scaffolding Use of physical artifacts	Technical help Skill improvement
	Focused attention Positive emotions & behavioral traits	Work-out problems and examples	Enhanced learning Exam preparation and guidelines
			Grading and timeliness issues

(continued)

**Table 8.** (continued)

Faculty characteristics	Faculty characteristics (EECS)	Instructional processes	Instructional processes (EECS)
	Humor in teaching (examples & instructor role)		Experiment explanation request
	General humor usage in instruction		Engineering career preparation experiences
	Insufficient detail		Homework quality & support
			Student engagement and learning environment
			Challenging course load

**Table 9.** EECS Versus Manual Code Comparison (Part 3).

Instructional resources	Instructional resources (EECS)	Content and sequence (EECS)	Drop codes - EECS
Classroom technology	Powerpoint presentation effectiveness and clarity	Course structure & organization	Engineering major/courses selection
Demonstrate structures and processes to solve work-out problems	Organizational structure	Duration and utilization	Increased self-confidence
Instructor-provided notes	Resource efficiency		Lack of specific feedback
Online utilities	External support		Layout confusion
Review sessions	Technology for enhanced engagement		Lack of specific help
Work-out problems and examples	Equipment maintenance & issues		
	Platform usefulness evaluation		
	Online platform types		
	Non-linear slide structure		
	Dyknow presentation design		
	Office hours effectiveness		
	Effective review sessions		
	Planning services		
	Reference materials (formula sheets)		
	Graduate teaching assistant role		

## Acknowledgements

This work used generative artificial intelligence as part of the method that we investigated and reported. This is different from uses of generative artificial intelligence to write the manuscript itself, which we did not use here. We thank the reviewers for their helpful comments and suggestions. We also thank the students who participated in the study.

## Declaration of Conflicting Interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.


## Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This material is based upon work supported by the National

Science Foundation under Grant No. #2107008 and the Virginia Tech Academy of Data Science Discovery Fund.

## ORCID iDs

Andrew Katz  <https://orcid.org/0000-0002-3554-9015>

Mitch Gerhardt  <https://orcid.org/0009-0006-4191-1654>

## References

- Abram, M. D., Mancini, K. T., & Parker, R. D. (2020). Methods to integrate natural language processing into qualitative research. *International Journal of Qualitative Methods*, 19, Article 160940692098460. <https://doi.org/10.1177/1609406920984608>
- Abrami, P. C., d'Apollonia, S., & Rosenfield, S. (2007). The dimensionality of student ratings of instruction: What we know and what we do not. In R. P. Perry, & J. C. Smart (Eds.), *The scholarship of teaching and learning in higher education: An*

- evidence-based perspective (pp. 385–456). Springer. [https://doi.org/10.1007/1-4020-5742-3\\_10](https://doi.org/10.1007/1-4020-5742-3_10)
- Alpaydin, E. (2021). *Machine learning*. MIT press.
- Ansari, A. N., Ahmad, S., & Bhutta, S. M. (2023). Mapping the global evidence around the use of ChatGPT in higher education: A systematic scoping review. *Education and Information Technologies*, 29(9), 11281–11321. <https://doi.org/10.1007/s10639-023-12223-4>
- Bala, I., & Mitchell, L. (2024). Thematic exploration of educational research after the COVID pandemic through topic modelling. *Journal of Applied Learning & Teaching*, 7(1), 40. <https://doi.org/10.37074/jalt.2024.7.1.40>
- Basit, T. (2003). Manual or electronic? The role of coding in qualitative data analysis. *Educational Research*, 45(2), 143–154. <https://doi.org/10.1080/0013188032000133548>
- Berdanier, C. G. P., Baker, E., Wang, W., & McComb, C. (2018). Opportunities for natural language processing in qualitative engineering education research: Two examples. In 2018 IEEE Frontiers in Education Conference (FIE) (pp. 1–6). <https://doi.org/10.1109/FIE.2018.8658747>
- Bhaduri, S. (2017). *NLP in engineering education: Demonstrating the use of natural language processing techniques for use in engineering education classrooms and research [Dissertation]*. Virginia Tech. <https://vtechworks.lib.vt.edu/server/api/core/bitstreams/d70f6e0b-7ee5-4bb5-a567-ee4221937acf/content>
- Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan), 993–1022.
- Boleda, G. (2020). Distributional semantics and linguistic theory. *Annual Review of Linguistics*, 6(1), 213–234. <https://doi.org/10.1146/annurev-linguistics-011619-030303>
- Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2), 77–101. <https://doi.org/10.1191/1478088706qp063oa>
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., & Amodei, D. (2020). *Language models are few-shot learners (arXiv:2005.14165)*. arXiv. <https://doi.org/10.48550/arXiv.2005.14165>
- Chen, X., Zou, D., Cheng, G., & Xie, H. (2020). Detecting latent topics and trends in educational technologies over four decades using structural topic modeling: A retrospective of all volumes of computers & education. *Computers & Education*, 151, Article 103855. <https://doi.org/10.1016/j.compedu.2020.103855>
- Chiu, M., Lim, S., & Silva, A. (2023). Visualizing design project team and individual progress using NLP: A comparison between latent semantic analysis and Word2Vector algorithms. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 37, Article e18. <https://doi.org/10.1017/S0890060423000094>
- Chowdhary, K. R. (2020). Natural Language processing. In K. R. Chowdhary (Ed.), *Fundamentals of artificial intelligence* (pp. 603–649). Springer. [https://doi.org/10.1007/978-81-322-3972-7\\_19](https://doi.org/10.1007/978-81-322-3972-7_19)
- Church, K. W., Chen, Z., & Ma, Y. (2021). Emerging trends: A gentle introduction to fine-tuning. *Natural Language Engineering*, 27(6), 763–778. <https://doi.org/10.1017/S1351324921000322>
- Cooper, G. (2023). Examining science education in ChatGPT: An exploratory study of generative artificial intelligence. *Journal of Science Education and Technology*, 32(3), 444–452. <https://doi.org/10.1007/s10956-023-10039-y>
- Creswell, J. W. (2014). *Research design: Qualitative, quantitative, and mixed methods approaches* (4 ed.). Sage.
- Crossley, S., Ocumpaugh, J., Labrum, M., Bradfield, F., Dascalu, M., & Baker, R. S. (2018). Modeling math identity and math success through sentiment analysis and linguistic features. In *International educational data mining society*. <https://eric.ed.gov/?id=ED593117>
- Crowston, K., Liu, X., & Allen, E. E. (2010). Machine learning and rule-based automated coding of qualitative data: Machine learning and rule-based automated coding of qualitative data. *Proceedings of the American Society for Information Science and Technology*, 47(1), 1–2. <https://doi.org/10.1002/meet.14504701328>
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-Training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 4171–4186). Association for Computational Linguistics. <https://doi.org/10.18653/v1/N19-1423>
- El Naqa, I., & Murphy, M. J. (2015). What is machine learning? In I. El Naqa, R. Li, & M. J. Murphy (Eds.), *Machine learning in radiation oncology: Theory and applications* (pp. 3–11). Springer International Publishing. [https://doi.org/10.1007/978-3-319-18305-3\\_1](https://doi.org/10.1007/978-3-319-18305-3_1)
- Erk, K. (2016). What do you know about an alligator when you know the company it keeps? *Semantics and Pragmatics*, 9, 17–21. <https://doi.org/10.3765/sp.9.17>
- Gamielien, Y., McCord, R., & Katz, A. (2023). *Utilizing natural language processing to examine self-reflections in self-regulated learning*. (SSRN Scholarly Paper 4487795): <https://doi.org/10.2139/ssrn.4487795>
- Ganesh, A., Scribner, H., Singh, J., Goodman, K., Hertzberg, J., & Kann, K. (2022). Response construct tagging: NLP-aided assessment for engineering education. In *Proceedings of the 17th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2022)* (pp. 250–261). <https://doi.org/10.18653/v1/2022.bea-1.29>
- Garman, A. N., Erwin, T. S., Garman, T. R., & Kim, D. H. (2021). Developing competency frameworks using natural language processing: An exploratory study. *The Journal of Competency-Based Education*, 6(3), Article e01256. <https://doi.org/10.1002/cbe.21256>
- Gemini Apps Privacy Hub—Gemini Apps Help. (nd). Google Support. Retrieved February 27, 2024, from: <https://support.google.com/gemini/answer/13594961?hl=en#zippy=%2Cwhy-is-human-review-of-my-gemini-apps-conversations-feedback-and-related-data-required>

- Gillioz, A., Casas, J., Mugellini, E., & Khaled, O. A. (2020). Overview of the transformer-based models for NLP tasks. In 2020 15th Conference on Computer Science and Information Systems (FedCSIS) (pp. 179–183). <https://doi.org/10.15439/2020F20>
- Grootendorst, M. (2022). *BERTopic: Neural topic modeling with a class-based TF-IDF procedure*. arXiv Preprint arXiv: 2203.05794.
- Hoel, A., & Dahl, T. I. (2019). Why bother? Student motivation to participate in student evaluations of teaching. *Assessment & Evaluation in Higher Education*, 44(3), 361–378. <https://doi.org/10.1080/02602938.2018.1511969>
- Hornstein, H. A. (2017). Student evaluations of teaching are an inadequate assessment tool for evaluating faculty performance. *Cogent Education*, 4(1), Article 1304016. <https://doi.org/10.1080/2331186X.2017.1304016>
- Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., & Saulnier, L., others. (2023). Mistral 7B. In *arXiv Preprint arXiv: 2310.06825*.
- Johri, A. (Ed.), (2023). *International handbook of engineering education research*. Taylor & Francis. <https://doi.org/10.4324/9781003287483>
- Johri, A., Wang, G. A., Liu, X., & Madhavan, K. (2011). Utilizing topic modeling techniques to identify the emergence and growth of research topics in engineering education. In *2011 Frontiers in Education Conference (FIE), October 2011, Rapid City, South Dakota*. T2F-1-T2F-6. <https://doi.org/10.1109/FIE.2011.6142770>
- Kastrati, Z., Dalipi, F., Imran, A. S., Pireva Nuci, K., & Wani, M. A. (2021). Sentiment analysis of students' feedback with NLP and deep learning: A systematic mapping study. *Applied Sciences*, 11(9), 3986. <https://doi.org/10.3390/app11093986>
- Katz, A., Norris, M., Alsharif, A. M., Klopfer, M. D., Knight, D. B., & Grohs, J. R. (2021). Using natural language processing to facilitate student feedback analysis. In 2021 ASEE Virtual Annual Conference Content Access, June 2021, virtual conference.
- Katz, A., Shakir, U., & Chambers, B. (2023a). *The utility of large language models and generative AI for education research (arXiv:2305.18125)*. arXiv. <https://arxiv.org/abs/2305.18125>
- Katz, A., Wei, S., Nanda, G., Brinton, C., & Ohland, M. (2023b). *Exploring the efficacy of ChatGPT in analyzing student teamwork feedback with an existing taxonomy (arXiv:2305.11882)*. arXiv. <https://arxiv.org/abs/2305.11882>
- Kenter, T., & De Rijke, M. (2015). Short text similarity with word embeddings. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (pp. 1411–1420).
- Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications*, 82(3), 3713–3744. <https://doi.org/10.1007/s11042-022-13428-4>
- Knight, D., Lattuca, L. R., Yin, A., Kremer, G., York, T., & Ro, H. K. (2012). An exploration of gender diversity in engineering programs: A curriculum and instruction-based perspective. *Journal of Women and Minorities in Science and Engineering*, 18(1), 55–78. <https://doi.org/10.1615/JWomenMinorScienEng.2012003702>
- Knight, D. B., Cameron, I. T., Hadgraft, R. G., & Reidsema, C. (2016). The influence of external forces, institutional forces, and academics' characteristics on the adoption of positive teaching practices across Australian undergraduate engineering. *International Journal of Engineering Education*, 32(2), 695–711.
- Lattuca, L. R., & Stark, J. S. (2009). *Shaping the college curriculum: Academic plans in context* (2nd ed.). Jossey-Bass.
- Li, X., & Li, J. (2023). *Angle-optimized text embeddings (arXiv: 2309.12871)*. arXiv. <https://arxiv.org/abs/2309.12871>
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019, July 26). *RoBERTa: A robustly optimized BERT pretraining approach*. arXiv.Org. <https://arxiv.org/abs/1907.11692v1>
- Madhani, N., Burstein, J., Elliot, N., Beigman Klebanov, B., Napolitano, D., Andreyev, S., & Schwartz, M. (2018). Writing mentor: Self-regulated writing feedback for struggling writers. In D. Zhao (Ed.), Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations (pp. 113–117). Association for Computational Linguistics. <https://aclanthology.org/C18-2025>
- Mao, R., Liu, Q., He, K., Li, W., & Cambria, E. (2023). The biases of pre-trained language models: An empirical study on prompt-based sentiment analysis and emotion detection. *IEEE Transactions on Affective Computing*, 14(3), 1743–1753. <https://doi.org/10.1109/TAFFC.2022.3204972>
- Marsh, R. (1984). A comparison of take-home versus in-class exams. *The Journal of Educational Research*, 78(2), 111–113. <https://doi.org/10.1080/00220671.1984.10885583>
- Mathew, A. N., Rohini, V., & Paulose, J. (2021). NLP-based personal learning assistant for school education. *International Journal of Electrical and Computer Engineering (IJECE)*, 11(5), 4522. <https://doi.org/10.11591/ijece.v11i5.pp4522-4530>
- McInnes, L., Healy, J., & Astels, S. (2017). hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11), 205. <https://doi.org/10.21105/joss.00205>
- McNamara, D. S., Crossley, S. A., & Roscoe, R. (2013). Natural language processing in an intelligent writing strategy tutoring system. *Behavior Research Methods*, 45(2), 499–515. <https://doi.org/10.3758/s13428-012-0258-1>
- Miles, M. B., Huberman, A. M., & Saldaña, J. (2014). *Qualitative data analysis: A methods sourcebook* (3rd ed.). Sage Publications, Inc.
- Min, B., Ross, H., Sulem, E., Veyseh, A. P. B., Nguyen, T. H., Sainz, O., Agirre, E., Heintz, I., & Roth, D. (2023). Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2), 1–40. <https://doi.org/10.1145/3605943>
- Muennighoff, N., Tazi, N., Magne, L., & Reimers, N. (2022). *MTEB: Massive text embedding benchmark*. arXiv Preprint arXiv: 2210.07316. <https://doi.org/10.48550/ARXIV.2210.07316>
- Najmani, K., Ajalloua, L., Benlahmar, E. H., Sael, N., & Zellou, A. (2023). BERTopic and LDA, which topic modeling technique to

- extract relevant topics from videos in the context of massive open online courses (MOOCs)? In S. Motahhir, & B. Bossoufi (Eds.), *Digital technologies and applications* (pp. 374–381). Springer Nature Switzerland. [https://doi.org/10.1007/978-3-031-29860-8\\_38](https://doi.org/10.1007/978-3-031-29860-8_38)
- Ozyurt, O., & Ayaz, A. (2022). Twenty-five years of education and information technologies: Insights from a topic modeling based bibliometric analysis. *Education and Information Technologies*, 27(8), 11025–11054. <https://doi.org/10.1007/s10639-022-11071-y>
- Paladines, J., & Ramirez, J. (2020). A systematic literature review of intelligent tutoring systems with dialogue in natural language. *IEEE Access*, 8, 164246–164267. <https://doi.org/10.1109/ACCESS.2020.3021383>
- Persing, I., & Ng, V. (2016). Modeling stance in student essays. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 2174–2184). <https://doi.org/10.18653/v1/P16-1205>
- Pope, C., Ziebland, S., & Mays, N. (2000). Qualitative research in health care: Analysing qualitative data. *BMJ British Medical Journal*, 320(7227), 114–116. <https://doi.org/10.1136/bmj.320.7227.114>
- Potamias, R. A., Siolas, G., & Stafylopatis, A.-G. (2020). A transformer-based approach to irony and sarcasm detection. *Neural Computing & Applications*, 32(23), 17309–17320. <https://doi.org/10.1007/s00521-020-05102-3>
- Privacy policy. (2023, November 14). *OpenAI*. <https://openai.com/policies/privacy-policy>
- Reyes, V., Bogumil, E., & Welch, L. E. (2021). The living codebook: Documenting the process of qualitative data analysis. *Sociological Methods & Research*, 53(1), 89–120. <https://doi.org/10.1177/0049124120986185>
- Sajadi, S., Ryan, O., Schibeli, L., & Huerta, M. (2023). WIP: Using generative AI to assist in individual performance feedback for engineering student teams. In *2023 IEEE Frontiers in Education Conference (FIE)* (pp. 1–5). <https://doi.org/10.1109/FIE58773.2023.10343517>
- Shaik, T., Tao, X., Li, Y., Dann, C., McDonald, J., Redmond, P., & Galligan, L. (2022). A review of the trends and challenges in adopting natural language processing methods for education feedback analysis. *IEEE Access*, 10, 56720–56739. <https://doi.org/10.1109/ACCESS.2022.3177752>
- Sheng, E., Chang, K.-W., Natarajan, P., & Peng, N. (2019). The woman worked as a babysitter: On biases in language generation. In K. Inui, J. Jiang, V. Ng, & X. Wan (Eds.), *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)* (pp. 3407–3412). Association for Computational Linguistics. <https://doi.org/10.18653/v1/D19-1339>
- Soledad, M., Grohs, J., Bhaduri, S., Doggett, J., Williams, J., & Culver, S. (2017). Leveraging institutional data to understand student perceptions of teaching in large engineering classes. In *2017 IEEE Frontiers in Education Conference (FIE)*, October 2017, Indianapolis, IN. <https://doi.org/10.1109/FIE.2017.8190608>
- Soledad, M. M. (2019). *Understanding the teaching and learning experience in fundamental engineering courses [dissertation]*. Virginia Tech. <https://hdl.handle.net/10919/101098>
- Somers, R., Cunningham-Nelson, S., & Boles, W. (2021). Applying natural language processing to automatically assess student conceptual understanding from textual responses. *Australasian Journal of Educational Technology*, 37(5), 98–115. <https://doi.org/10.14742/ajet.7121>
- Sproule, R. (2000). Student evaluation of teaching: Methodological critique. *Education Policy Analysis Archives*, 8, 50. <https://doi.org/10.14507/epaa.v8n50.2000>
- Sunar, A. S., & Khalid, M. S. (2024). Natural Language processing of student's feedback to instructors: A systematic review. *IEEE Transactions on Learning Technologies*, 17, 741–753. <https://doi.org/10.1109/TLT.2023.3330531>
- Tai, R. H., Bentley, L. R., Xia, X., Sitt, J. M., Fankhauser, S. C., Chicas-Mosier, A. M., & Monteith, B. G. (2024). An examination of the use of large language models to aid analysis of textual data. *International Journal of Qualitative Methods*, 23, Article 16094069241231168. <https://doi.org/10.1177/16094069241231168>
- Terry, G., Hayfield, N., Clarke, V., & Braun, V. (2017). The Sage handbook of qualitative research in psychology. In *The Sage handbook of qualitative research in psychology* (pp. 17–36). Sage Publications Ltd. <https://doi.org/10.4135/9781526405555>
- Troussas, C., Papakostas, C., Krouska, A., Mylonas, P., & Sgouropoulou, C. (2023). Personalized feedback enhanced by natural language processing in intelligent tutoring systems. In C. Frasson, P. Mylonas, & C. Troussas (Eds.), *Augmented intelligence and intelligent tutoring systems* (pp. 667–677). Springer Nature Switzerland. [https://doi.org/10.1007/978-3-031-32883-1\\_58](https://doi.org/10.1007/978-3-031-32883-1_58)
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention is all you need*. Long Beach, CA, USA: 31st Conference on Neural Information Processing Systems (NIPS 2017).
- Vig, J., Gehrmann, S., Belinkov, Y., Qian, S., Nevo, D., Singer, Y., & Shieber, S. (2020). Investigating gender bias in language models using causal mediation analysis. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (Vol. 33, pp. 12388–12401). Curran Associates, Inc. [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/92650b2e92217715fe312e6fa7b90d82-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/92650b2e92217715fe312e6fa7b90d82-Paper.pdf)
- Vuong, T., Andolina, S., Jacucci, G., & Ruotsalo, T. (2021). Does more context help? Effects of context window and application source on retrieval performance. *ACM Transactions on Information Systems*, 40(2), 1–40. <https://doi.org/10.1145/3474055>
- Wachtel, H. K. (1998). Student evaluation of college teaching effectiveness: A brief review. *Assessment & Evaluation in Higher Education*, 23(2), 191–212. <https://doi.org/10.1080/0260293980230207>



- Wallach, H. M., Murray, I., Salakhutdinov, R., & Mimno, D. (2009). Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 1105–1112).
- Wang, Z., Chen, J., Chen, J., & Chen, H. (2023). Identifying interdisciplinary topics and their evolution based on BERTopic. In *Scientometrics*. <https://doi.org/10.1007/s11192-023-04776-5>.
- Wankhade, M., Rao, A. C. S., & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, 55(7), 5731–5780. <https://doi.org/10.1007/s10462-022-10144-1>
- Weitzel, L., Prati, R. C., & Aguiar, R. F. (2016). The comprehension of figurative language: What is the influence of irony and sarcasm on NLP techniques? In W. Pedrycz, & S.-M. Chen (Eds.), *Sentiment analysis and ontology engineering: An environment of computational intelligence* (pp. 49–74). Springer International Publishing. [https://doi.org/10.1007/978-3-319-30319-2\\_3](https://doi.org/10.1007/978-3-319-30319-2_3)
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., & Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *In A Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations* (pp. 38–45). Association for Computational Linguistics.
- Xiao, Z., Yuan, X., Liao, Q. V., Abdelghani, R., & Oudeyer, P.-Y. (2023). Supporting qualitative analysis with large language models: Combining codebook with GPT-3 for deductive coding. In *Companion Proceedings of the 28th International Conference on Intelligent User Interfaces* (pp. 75–78). <https://doi.org/10.1145/3581754.3584136>
- Yun, E. (2020). Review of trends IN PHYSICS education research using topic modeling. *Journal of Baltic Science Education*, 19(3), 388–400. <https://doi.org/10.33225/jbse/20.19.388>