Do Language Models Understand Human Needs on Text Summarization?

Anonymous ACL submission

Abstract

With the popularity of large language models and their high-quality text generation capabilities, researchers are using them as auxiliary tools for text summary writing. Although summaries generated by these large language models are smooth and capture key information sufficiently, the quality of their output depends on the prompt and the generated text is somewhat procedural to a certain extent. In order to understand whether large language models truly understand human needs, we construct LecSumm, in which we recruit 200 college students to write summaries for lecture notes on ten different machine learning topics, and analyze real-world human summary needs in the dimensions of summary length structure, modality and content depth. We further evaluate fine-tuned and prompt-based language models on LecSumm and show that the commercial GPT models showed better performance in summary coherence, fluency and relevance, but still fall shot in faithfulness and can better capture human needs even with advanced prompt design while fine-tuned models do not effectively learn human needs from the data. Our LecSumm dataset brings new challenges to both fine-tuned models and prompt-based large language models on the task of human-centered text summarization.

1 Instruction

With the huge amount of training data, the development of large language models (LLMs), such as the GPT series (OpenAI, 2024), the PaLM series (Aakanksha Chowdhery, 2022), Mistral (Jiang et al., 2023) and LLaMA (AI@Meta, 2024), have achieved remarkable success by unifying the generative paradigm with different NLP tasks(Wei et al., 2024a,c; Wan et al., 2023; Wang et al., 2023a; Qin et al., 2023; tse Huang et al., 2024). In certain NLP fields, such as text summarization, LLMs achieve decent performance without additional training data and even surpass traditional models supervised fine-tuned models (Zhang et al., 2024). Recent studies employ LLMs as auxiliary tools for human-centered NLP (Passali et al., 2021; Hu et al., 2023), ranging from human-centered design to human-in-the-loop interaction with LLMs.

When generating human-centered summaries with LLMs, specific human needs can be incorporated through two different approaches : (i) Explicitly, add external constraints to the summarization model, such as prompt design and different hyper-parameter settings. (ii) Implicitly, construct specific source-target summary datasets that reflect human needs to finetune the language model, enabling it to learn the hidden need from the data. Our research question is: **Do language models really understand human needs on text summarization?**

To answer this question, we first design a lecture note summarization task to discover human needs in real-world data and construct a dataset containing human-centered summaries, the framework of the task is shown in Figure 1. We recruited 200 university students and designed a task of writing lecture note summaries: 10 different topics related to machine learning were given to the annotators, together with the corresponding lecture notes, and the participants were required to write summaries based on the lecture notes. There is no hard limitation (e.g. length, structure) on the summary written process, participants are allowed to use related materials to equip the lecture notes, the only limitation is it has to be written by the participants and cannot be returned by machines. What we observe is that different annotators utilize a combination of various dimensions to reflect their individual needs when writing summaries, these human needs range in four dimensions: structure, modality, length, and content depth.

Then, we construct the LecSumm dataset which includes the provided lecture notes and human writ-

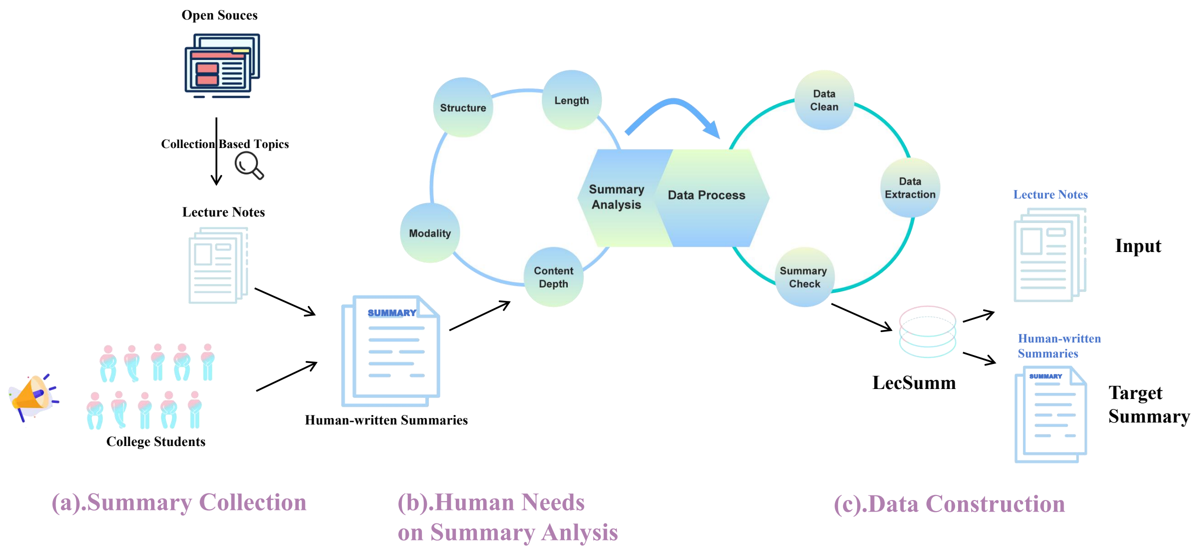

Figure 1: The framework of our work is divided into three stages: (a) Summary Collection. We designed a summary collection task and recruited annotators online. They were asked to write summaries based on the provided lecture notes gathered from open-source data on specific topics. (b) Human Needs Analysis on Summaries. Human-written summaries were analyzed from four dimensions: length, structure, modality, and content depth to discover human needs. (c) Dataset Construction. The dataset was constructed after summary checking and data processing, with lecture notes as input sources, and human-written summaries as target summaries.

ten summaries, by which are we experiment with fine-tuned supervised models, and prompt-based zero-shot LLMs. We find that zero-shot LLM can better understand human needs given proper prompting design. Our main contribution is presented as follows:

- We design a human-centered text summarization task to collect human-guided summaries for lecture notes, and propose a LecSumm dataset containing human-centered summaries.

- Based on LecSumm, we analyze the human need bias for text summarization in four different dimensions: structure, modality, length, and content depth. We show that over half of the human written summaries tend to be unstructured, text-only and general.

- We experiment the human-centered text summarization modeling capability with fine-tuned and prompt-based zero-shot LLMs, and find that prompt-based zero-shot LLMs can better capture human needs while fine-tuned models do not effectively learn human needs from the data.

2 Lecture Note Summary Collection and Analysis

2.1 Human-centered Summary Collection

Data Collection We collected machine learning lecture notes which cover ten major topics, as shown in Table 2, the lecture notes are all open source and can be found on the Internet, most of them are released by public universities. Additionally, we recruited 200 university students from IT department and asked them to write summaries for the ten topics after reading the lecture notes. Apart from the lecture notes, these expert annotators can acquire additional material from the Internet, these additional material are regarded as specific human needs. An example of an annotator-written summary is shown in Appendix A.

Annotator Statistic Recruited annotators are students from university IT departments, while recruiting student participants, we also asked the annotators to provide the following information: gender, first language, qualification, and machine learning working experience. Table 1 shows that 82% of the annotators are native English speakers and 65% of them indicate that they have machine learning related working experience. This shows the high quality of the annotator group and will guarantee the real human needs in the annotation

Annotator	Percentage
Gender	53% Male / 47% Female
First Language	82% Native English
Qualification	30% B.S / 70% M.S
ML Experience	65% experienced / 35% non-exp

Table 1: Annotator statistics

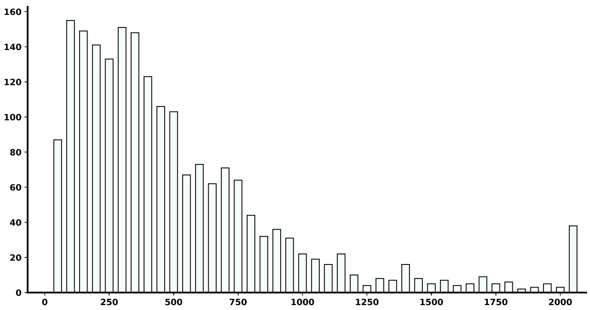

Figure 2: The length distribution of annotators-written summaries.

	Topic
1	Machine Learning Overview
2	Data Wrangling
3	Clustering Algorithms
4	Principal Component Analysis (PCA)
5	A supervised learning algorithm
6	Linear regression
7	Support Vector Machine(SVM)
8	Decision tree algorithms
9	Ensemble learning
10	Neural Networks and Deep Learning

Table 2: These are ten topics covered by the machine learning lecture notes we collected.

process.

2.2 Human Needs Analysis on Summaries

Upon receipt of the summaries from the annotators, we proceeded to examine the summaries in their entirety and identify the individual differences needs in real-world summaries across four dimensions: length, structure, modality, and content depth.

Length The length distribution of summaries, as shown in Figure 2. The average tokens in the note summaries are 500 tokens, and the longest one could be up to 6406 tokens. However, it could be observed that the majority of returned summaries had a length of less than 1000 tokens.

Structure In terms of structure, our primary focus was on the usage of headings in the summaries. Therefore, we designed four metrics to analyze this dimension: *No structure*: There are no headings in the summary. *Primary Heading*: This is the highest level of heading in a document. It is typically used to introduce the main content or theme and is the most prominent and important heading. *Secondary Heading*: This is a subheading under the primary heading, used to divide the main content further. Secondary headings help organize and structure information, making it easier for readers to find specific sections or subtopics. *Tertiary Heading*: This is a further subdivision under a secondary heading, used to classify information in more detail. Tertiary headings are typically used to introduce more specific content or sub-items, making the document's structure more detailed and hierarchical. It could be observed that approximately 90% of annotators have indicated a preference for the use of a simpler summary structure, yet it is evident that a small minority of individuals continue to employ tertiary Heading in their summaries from Table 3.

Modality Additionally, we noticed that the written summaries contained both text and image modalities. Consequently, we conducted an analysis of the proportion of summaries that contained *solely text* and those that contained *both text and images*. The result presented in Table 3 shows that summaries comprising a combination of text and images accounted for 22.90% of the total. These summaries serve to complement or emphasize the textual content with images, thereby enhancing the intuition and clarity of the conveyed information.

Content Depth The in-depth details of the content were considered. We referred to the previous researchers' criteria for the details of the content of the summary and redefined them as either *General* or *Detailed*. *General*: Only including material titles or summarize materials or learning process. *Detailed*: Include material contents and knowledge details. As shown in Table 3 demonstrated, 37.8% annotators were inclined to describe the details of their knowledge when writing their summaries. We further analyzed the detailed summary content and observed that only 5% of the students elaborated on formulas, principles, etc. when writing their summaries, and the remaining described the definition and framework of knowledge. At the same time, we analyzed the 10 summaries written by individuals and found that each individual had a different focus when writing them. For example, some indi-

Dimensions	Metrics	Percentage
Structure	No structure	60.80%
	Primary Heading	30.10%
	Secondary Heading	8.00%
	Tertiary Heading	1.20%
Modality	Only text	77.10%
	Text+image	22.90%
Content Depth	General	62.20%
	Detailed	37.80%

Table 3: We analyzed three dimensions of the summaries: structure, modality, and content depth. For each dimension, we designed different metrics to visually present the varying preferences of different annotators when writing summaries.

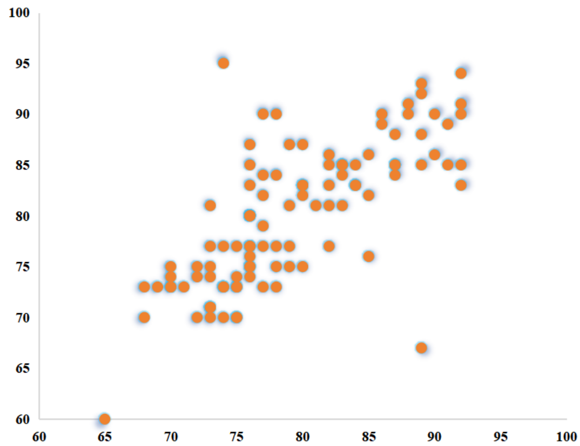

Figure 3: The relationship between two expert annotators' scores. It can seen there is a strong positive correlation between the two annotators.

viduals tended to focus on model principles while others preferred to systematize knowledge under a specific topic.

Summary Each annotator has different needs when writing summaries. From the analysis above, it can be seen that some annotators prefer a concise and straightforward structure with general content, while others like to use a combination of text and images to enhance understanding. Combining different human needs, we can design different prompts for large language models, e.g. *Please generate a summary containing tertiary Heading headings, details information and formulas about this source text.*

3 LecSumm

3.1 Summary Quality Control

In this section, we invited two university professors as expert annotators to evaluate the summaries written by annotators based on the following four dimensions to ensure the quality of the data:

Coherence: The overall quality of all sentences. "The summary should be well-structured and well-organized. It should not just be a collection of related information, but should build coherent information about a topic from one sentence to the next."

Consistency: The factual consistency between the summary and its source. A factually consistent summary only contains statements that are present in the source document.

Fluency: The quality of individual sentences. The sentences in the summary "should not have formatting issues, capitalization errors, or obviously ungrammatical sentences (e.g., fragments, missing parts), which would make the text difficult to read."

Relevance: The selection of important content from the source. The summary should only include important information from the source document. Annotators were instructed to penalize summaries containing redundant and superfluous information.

We randomly selected 100 summary samples. Experts were required to score the summaries based on the above four dimensions, with a maximum of 25 points for each dimension, and calculate the total score. To assess inter-annotator agreement, we calculated Krippendorff's alpha coefficient (Krippendorff, 2011).

The Krippendorff's alpha score is 75.82%, indicating that the experts showed very high consistency in their annotations. Figure 3 confirms this, showing that most of the annotation scores for the summaries are between 75 and 95. These results collectively indicate that the quality of the summaries is high and that the experts exhibited a high level of consistency in their evaluations.

3.2 Dataset Construction

The data including the provided lecture notes and human-written summaries was extracted into plain text, removing all external information besides summaries. A total of 200 samples were obtained after cleansing and filtering, including the ten lecture notes as fixed input together with 2,000 human written summaries as targets. [1] The average input

[1]The dataset will be released for the research community.

document length of LecSumm is 6.5k, about one-third of the documents are over 9k, and the topics are in Table 2. We split our dataset into the train (1,600, 80%), validation (200, 10%), and test (200, 10%) subsets.

3.3 Dataset Properties

In this section, we use four indicators to evaluate the intrinsic characteristics of datasets: coverage, density, redundancy and n-gram overlap. We choose five commonly used English long document datasets for comparison. **CNN-DM** (Nallapati et al., 2016) are news corpus from CNN and Daily Mail websites. **PubMed and arXiv** (Nallapati et al., 2016) are from scientific papers. **Big-Patent** (Sharma et al., 2019) consists of records of U.S. patent documents. **GovReport** (Huang et al., 2021) is a collection of reports published by the U.S. Government Countability Office and Congressional Research Service.

Coverage (Grusky et al., 2018) quantifies the proportion of words in a summary that originate from an extractive fragment within the document. Its calculation method is as follows:

$$Coverage(D,S) = \frac{1}{|S|} \sum_{f \in F(D,S)} |f| \quad (1)$$

where D and S represent the document and its summary respectively. $F(D,S)$ is the set that includes all extractive fragments. $|\bullet|$ signifies the length of a token sequence. A higher coverage score indicates that more content is directly copied from the document when generating the summary.

Density (Grusky et al., 2018) is similar to coverage, where the sum of fragment lengths is changed to the sum of squares of lengths:

$$Density(D,S) = \frac{1}{|S|} \sum_{f \in F(D,S)} |f|^2 \quad (2)$$

In the event that the length of each fragment is relatively brief, the density value will be comparatively low. This implies that if two summaries share the same coverage value, the one with a lower density might exhibit greater variability, due to the fact that its fragments are relatively short and discontinuous.

Redundancy (Bommasani and Cardie, 2020) is used to evaluate whether sentences in a summary are similar to each other.

$$Redundancy(S) = \underset{(a,b) \in M \times M, a=b}{mean} R_L(x,y) \quad (3)$$

where M is sentence set of summary S, (a,b) is a sentence pair. R_L is ROUGE-L F1-score. Redundancy can be utilized to measure the degree to which sentences in a summary repeat information unnecessarily. In essence, a high-quality summary ought to strive for maximum conciseness.

Table 4 shows coverage, density, redundancy and n-gram overlap scores of several datasets. To be specific, LecSumm achieves highest scores on coverage and redundancy, which means that fewer summary contents in the datasets are extracted from documents, and every summary has less repeated information, which further shows that human-centered summaries vary significantly. LecSumm's performance on the density metric is moderate due to the presence of numerous specific terms, definitions, and concepts in lecture notes. These token sequences tend to be long and difficult to rephrase, necessitating their retention in the human-written summaries, which leads to a decrease in the density score. Nevertheless, the coverage metric indicates that LecSumm's summaries still possess the highest level of abstraction. Taking these three metrics into consideration, it is evident that LecSumm performs best in terms of abstractiveness and conciseness.

In addition to the aforementioned metrics, we further evaluate the abstractiveness of datasets. Specifically, we quantified it by calculating the percentage of novel n-grams in the summaries that didn't appear in the source text. Table 4 displays high percentages of novel tri-grams and 4-grams (Phang et al., 2023a). Combining the scores of coverage, density, and novel n-grams, it can be concluded that LecSumm possesses the best abstractiveness, making it more suitable for evaluating human-centered text summarization.

4 Experiments

We conducted a series of experiments on LecSumm to answer the question "Do Language Models Understand Human Needs on Text Summarization?"

4.1 Baseline

We use **LED** (Beltagy et al., 2020), **PEGASUS-X** (Phang et al., 2023b), and **LongT5** (Guo et al., 2022) as baselines. LED is based on Longformer, it combines local windowed attention and task-motivated global attention. PEGASUS-X uses a staggered block-local Transformer with global encoder tokens. LongT5 integrates attention ideas

Dataset	Coverage#rank	Density#rank	Redundancy#rank	% of novel n-grams			
				uni-	bi-	tri-	4-
CNN-DM	0.89#3	3.6#2	0.157#5	19.5	56.8	74.4	82.8
PubMed	0.893#4	5.6#5	0.146#4	12.4	44	65.3	76
arXiv	0.920#5	3.7#3	0.144#3	9.5	41	66.4	79.6
BigPatent	0.861#2	2.1#1	0.223#6	13.5	52.6	78.3	89.5
GovReport	0.942#6	7.7#6	0.124#2	5.7	32.7	56.3	68.9
LecSumm	0.860#1	5.5#4	0.122#1	13.3	53.3	77.4	85.6

Table 4: Intrinsic evaluations of different summarization datasets, including values and rankings, calculated on test sets only. Smaller coverage, density and redundancy values are deemed preferable. Percentages of novel n-grams in summaries of different datasets are also provided.

	Prompt
L	Please generate a summary with a maximum length of 300 words about source text.
L + C	Please generate a summary containing detailed information with a maximum length of 300 words about the source text.
L + C + S	Please generate a summary containing tertiary headings with a maximum length of 300 words about the source text.

Table 5: These are prompts containing human needs, and we can only restrict the output length, due to the limitations of the model API. Abbreviations are for L (Length), C (Content), and S (Structure).

from ETC and adopts pre-training strategies from PEGASUS. These models support long input at most 16k tokens. More details are in Appendix C.1.

In addition, we evaluate large language models under zero-shot settings. We choose **GPT-3-turbo**, **GPT-4-turbo** and **GPT-4o**[2], which support long inputs. These models are proprietary to OpenAI and have been finetuned on extensive datasets.

4.2 Settings

For pre-trained language models, we used the led-large-16384[3], pegasus-x-large[4], and long-t5-tglobal-large[5] models to summarize. We use an NVIDIA A100 80GB PCIe GPU for experiments. Models are used transformers4.35.2[6] to finetune for 10 epochs. We set the input token 8k, output token 1024, batch_size 2, the remaining parameters are default argument values.

For zero-shot LLMs, we use GPT-3.5-turbo[7], GPT-4-turbo[8] and GPT-4o[9] for implementation. We put the lecture notes as the source input and removed the figures. We designed the LLM prompts using these tertiary headings and content details as key elements of "human needs." The prompt design is shown in the table 5.

4.3 Evaluation

We utilized some automated evaluation metrics to assess the summaries generated by the models.

Rouge We use F1-score of ROUGE-1, ROUGE-2 and ROUGE-L[10], taking into account the completeness, readability and order of summary.

BertScore (Zhang et al., 2020) computes a similarity score for each token in the candidate sentence with each token in the reference sentence. It correlates better with human judgments.

SummaC (Summary Consistency; Laban et al., 2022) is focused on evaluating factual consistency in summarization. They use NLI for detecting inconsistencies by splitting the document and summary into sentences and computing the entailment probabilities on all document/summary sentence pairs, where the premise is a document sentence

[2]https://openai.com/
[3]https://huggingface.co/allenai/led-large-16384
[4]https://huggingface.co/google/pegasus-x-large
[5]https://huggingface.co/google/long-t5-tglobal-large
[6]https://huggingface.co/docs/transformers/index
[7]https://platform.openai.com/docs/models/gpt-3-5-turbo
[8]https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4
[9]https://platform.openai.com/docs/models/gpt-4o
[10]https://huggingface.co/docs/datasets/how_to_metrics

Model	ROUGE			Bertscore			SummaC	UniEval		
	R-1	R-2	R-L	P	R	L		coherence	fluency	relevance
Fine-tuned Models										
LED	15.83	4.04	10.38	80.70	79.25	79.90	68.21	49.59	76.08	50.13
PEGASUS-X	21.67	4.72	13.21	78.50	79.40	78.92	**80.10**	73.98	74.51	72.72
LongT5	21.67	4.74	13.22	78.50	79.41	78.91	80.02	73.98	74.51	72.73
Zero-shot LLM-Prompt (L)										
GPT-3.5-turbo	32.75	7.63	16.43	**82.89**	77.42	80.05	76.81	94.11	94.93	86.55
GPT-4-turbo	**38.11**	9.30	18.10	81.45	78.89	80.14	65.87	96.22	**95.40**	93.34
GPT-4o	38.01	**9.33**	17.14	80.13	79.22	79.66	75.31	**97.83**	91.97	**95.18**
Zero-shot LLM-Prompt (L+C)										
GPT-3.5-turbo	34.01	7.14	16.61	82.79	77.60	80.10	68.26	95.62	95.03	86.01
GPT-4-turbo	33.79	8.34	16.48	79.16	78.93	79.06	70.49	97.60	94.40	94.19
GPT-4o	34.49	7.34	15.25	78.94	78.66	78.79	67.25	96.62	95.01	92.11
Zero-shot LLM-Prompt (L+C+S)										
GPT-3.5-turbo	37.23	8.66	**18.23**	82.17	78.46	**80.25**	78.63	94.50	91.33	89.16
GPT-4-turbo	37.62	9.31	16.39	79.21	78.93	79.06	67.79	96.54	93.45	94.28
GPT-4o	30.31	8.76	15.35	77.55	78.45	77.99	73.54	92.57	89.28	90.94

Table 6: It presents evaluation results of automatic summary metrics for LecSumm on pre-trained and zero-shot LLM.

and the hypothesis is a summary sentence. They aggregate the NLI scores for all pairs by either taking the maximum score per summary sentence and averaging (SCZS) or by training a convolutional neural network to aggregate the scores (SCConv). We report SCConv score and use the publicly available for implementation[11].

UniEval (Zhong et al., 2022) is a unified multi-dimensional evaluator which re-frames NLG evaluation as a Boolean Question Answering (QA) task, and by guiding the model with different questions to evaluate from multiple dimensions. We report coherence score, fluency score, relevance score computed by UniEval[12].

4.4 Do Language Models Understand Human Needs on Text Summarization?

Fine-tuned Model See Table 6, fine-tuned models perform relatively well in Summac scores and demonstrate good factual consistency. However, its Rouge and Unieval scores are lower, especially with Rouge not exceeding 30%, which differs significantly from its performance on common datasets like CNN/DM and Government(Phang et al., 2023b; Guo et al., 2022). We also analyze the summaries generated by fine-tuned models. While we observe some structure in the summaries generated by LongT5 and PEGASUS-X, they do not fully cover subsequent content, which may lead to vocabulary repetition and affect the model's evaluation. Overall, this indicates that during the training phase, the fine-tuned language models do not effectively learn the relationship between source documents and target summaries, nor accurately capture the human needs present in target summaries.

Zero-shot LLM We conduct automated metric evaluations on scenarios with and without the inclusion of human needs. Apart from SummaC score, the evaluation metrics for the GPT series are higher than those for the pre-trained language model, thanks to their robust performance and extensive pre-training data. As human needs continue to evolve and expand, different models show slight improvements across various metrics. Moreover, the generated summaries visually align with target summaries to an extent of 85%. However, we do not analyze the human needs of linguistic features in the target summaries, which results in slightly lower Rouge scores. Overall, language models can comprehend and generate summaries that align appropriately when provided with human needs.

[11]https://github.com/tingofurro/summac
[12]https://github.com/maszhongming/UniEval

5 Related Work

Large language model for text summarization Most LLMs adopt an autoregressive structure similar to GPT, capable of automatic text summarization (ATS) (Houlsby et al., 2019). However, as the model size increased, full parameter training became costly. Research gradually shifted towards more cost-effective and efficient methods, including fine-tuning and prompt engineering. Prompt engineering for LLMs involves exploring and formulating strategies to maximize the use of specific functions inherent in large language models (LLMs). This process requires optimizing the input text string to more effectively leverage the LLM's intrinsic knowledge, thereby enhancing the interpretation of the input text (Liu et al., 2023). This significantly improves the quality of the generated summaries. Notably, prompt engineering is advantageous because it does not require extensive training or relies only on a small number of samples (Narayan et al., 2021), thus reducing resource expenditure. The implementation of prompt engineering is based on methods such as template engineering, chain of thought (CoT), and agent interaction. Template engineering is another natural way to create prompts by manually creating intuitive templates based on human introspection (Zhao et al., 2023). Chain of thought (Wei et al., 2024b) is a series of intermediate reasoning steps that can significantly enhance the LLM's ability to perform complex reasoning tasks. To address issues of factual hallucinations and information redundancy in ATS, a summarization chain of thought (SumCoT) (Wang et al., 2023b) technique was proposed to guide LLMs in gradually generating summaries, helping them integrate finer-grained details from the source document into the final summary. Agents are artificial entities that perceive the environment, make decisions, and take actions (Xi et al., 2023). A three-agent generation pipeline, consisting of a generator, a lecturer, and an editor, can enhance the customization of LLM-generated summaries to better meet user expectations.

Human-centered text summarization Human-centered text summarization approach emphasizes designing and developing summarization models that align with the needs and preferences of human users. This approach primarily involves human-computer interaction for building the summarization models and leverages large language models (LLMs) as evaluators to assist in assessing the quality metrics such as fluency and factual consistency of the summaries (Cheng et al., 2022; Sottana et al., 2023). Additionally, this approach is applied to the construction of text summarization datasets, which involves two stages: data collection and data annotation. Existing research predominantly focuses on the data annotation stage, accomplished through human interaction (Gururangan et al., 2018). In contrast, human-centered data collection should prioritize simulating real-use scenarios so that the data reflects actual human needs. However, common datasets like CNN/DM, Xsum and government datasets (Narayan et al., 2018; Yasunaga et al., 2019; Koupaee and Wang, 2018) do not simulate real scenarios in their summary collection process and therefore fail to adequately reflect human needs.

6 Conclusion

We design a lecture note summarization task, which aims at obtaining human-centered summaries and analyzes the human preferences existing in the summaries from four dimensions: length, structure, modality, and content depth. Meanwhile, we build a new dataset LecSumm that, compared to publicly available datasets, exhibits higher levels of human-specific needs. By conducting automatic and manual evaluations of benchmark models, we find that prompt-based LLMs show better performance than capturing the human needs than fine-tuned models. We hope that our analysis results can provide insights for better prompt design, and our dataset can contribute to the research in human-centered text summarization.

Limitation

There are few limitations to our work: The topics of lecture notes are only limited to machine learning; We only recruited 200 participants due to the expensive annotation cost; The prompts that we try are also limited, automatic prompt design may be considered in future work.

Ethics Statement

Data collection approval was received from an ethics review board. No identified personal information is collected in the data collection process. All codes and data used in this paper comply with the license for use.

References

Sharan Narang Aakanksha Chowdhery. 2022. Palm: Scaling language modeling with pathways. *Preprint*, arXiv:2204.02311.

AI@Meta. 2024. Llama 3 model card.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *Preprint*, arXiv:2004.05150.

Rishi Bommasani and Claire Cardie. 2020. Intrinsic evaluation of summarization datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8075–8096, Online. Association for Computational Linguistics.

Ruijia Cheng, Alison Smith-Renner, Ke Zhang, Joel R. Tetreault, and Alejandro Jaimes. 2022. Mapping the design space of human-ai interaction in text summarization. *Preprint*, arXiv:2206.14863.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719, New Orleans, Louisiana. Association for Computational Linguistics.

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. LongT5: Efficient text-to-text transformer for long sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736, Seattle, United States. Association for Computational Linguistics.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. *Preprint*, arXiv:1803.02324.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Yebowen Hu, Kaiqiang Song, Sangwoo Cho, Xiaoyang Wang, Hassan Foroosh, and Fei Liu. 2023. Decipherpref: Analyzing influential factors in human preference judgments via gpt-4. Proceedings of the 2023 Conference on Empirical Methods in Natural Language ....

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. 2021. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *Preprint*, arXiv:1810.09305.

Philippe Laban, Tobias Schnabel, Paul N. Bennett, and Marti A. Hearst. 2022. SummaC: Re-visiting NLI-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, 10:163–177.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Shashi Narayan, Yao Zhao, Joshua Maynez, Gonçalo Simões, Vitaly Nikolaev, and Ryan McDonald. 2021. Planning with learned entity prompts for abstractive summarization. *Transactions of the Association for Computational Linguistics*, 9:1475–1492.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Tatiana Passali, Alexios Gidiotis, Efstathios Chatzikyriakidis, and Grigorios Tsoumakas. 2021. Towards human-centered summarization: A case study on financial news. In *Proceedings of the First Workshop on Bridging Human–Computer Interaction and Natural Language Processing*, pages 21–27.

Jason Phang, Yao Zhao, and Peter Liu. 2023a. Investigating efficiently extending transformers for long input summarization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3946–3961, Singapore. Association for Computational Linguistics.

Jason Phang, Yao Zhao, and Peter Liu. 2023b. Investigating efficiently extending transformers for long input summarization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3946–3961, Singapore. Association for Computational Linguistics.

Libo Qin, Qiguang Chen, Fuxuan Wei, Shijue Huang, and Wanxiang Che. 2023. Cross-lingual prompting: Improving zero-shot chain-of-thought reasoning across languages. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2695–2709, Singapore. Association for Computational Linguistics.

Eva Sharma, Chen Li, and Lu Wang. 2019. BIGPATENT: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213, Florence, Italy. Association for Computational Linguistics.

Andrea Sottana, Bin Liang, Kai Zou, and Zheng Yuan. 2023. Evaluation metrics in the era of gpt-4: Reliably evaluating large language models on sequence to sequence tasks. *Preprint*, arXiv:2310.13800.

Jen tse Huang, Man Ho Lam, Eric John Li, Shujie Ren, Wenxuan Wang, Wenxiang Jiao, Zhaopeng Tu, and Michael R. Lyu. 2024. Emotionally numb or empathetic? evaluating how llms feel using emotionbench. *Preprint*, arXiv:2308.03656.

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. 2023. GPT-RE: In-context learning for relation extraction using large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3534–3547, Singapore. Association for Computational Linguistics.

Jiaan Wang, Yunlong Liang, Fandong Meng, Beiqi Zou, Zhixu Li, Jianfeng Qu, and Jie Zhou. 2023a. Zero-shot cross-lingual summarization via large language models. In *Proceedings of the 4th New Frontiers in Summarization Workshop*, pages 12–23, Singapore. Association for Computational Linguistics.

Yiming Wang, Zhuosheng Zhang, and Rui Wang. 2023b. Element-aware summarization with large language models: Expert-aligned evaluation and chain-of-thought method. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8640–8665, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024a. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024b. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, Yong Jiang, and Wenjuan Han. 2024c. Chatie: Zero-shot information extraction via chatting with chatgpt. *Preprint*, arXiv:2302.10205.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. 2023. The rise and potential of large language model based agents: A survey. *Preprint*, arXiv:2309.07864.

Michihiro Yasunaga, Jungo Kasai, Rui Zhang, Alexander R. Fabbri, Irene Li, Dan Friedman, and Dragomir R. Radev. 2019. Scisummnet: A large annotated corpus and content-impact models for scientific paper summarization with citation networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7386–7393.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *Preprint*, arXiv:1904.09675.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. 2024. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57.

Jinming Zhao, Ming Liu, Longxiang Gao, Yuan Jin, Lan Du, He Zhao, He Zhang, and Gholamreza Haffari. 2020. Summpip: Unsupervised multi-document summarization with sentence graph compression. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 1949–1952, New York, NY, USA. Association for Computing Machinery.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *Preprint*, arXiv:2303.18223.

Ming Zhong, Yang Liu, Da Yin, Yuning Mao, Yizhu Jiao, Pengfei Liu, Chenguang Zhu, Heng Ji, and Jiawei Han. 2022. Towards a unified multi-dimensional evaluator for text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2023–2038, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

A Human-written Summary

In §2.1, we collect some annotator-written Summaries. Figure 4 shows the example of an annotator-written summary, and we can observe that it includes a lecture notes topic, primary headings, and formulas.

This topic was all about the nonlinear machine learning models: Neural Networks and Deep Learning. The following points were covered:

- Artificial Neural Networks Motivation:
  - Linear models may not be sufficient for extremely complex problems (decision boundaries are extremely nonlinear)
  - SVMs can construct nonlinear functions, but only with fixed transformations (based on the kernel)
  - When it is unclear what kernel function should be used, and it is preferable to learn these features from the data itself
- System components:
  - Input layer
  - Hidden layer(s)
  - Combiners
  - Nonlinear activation functions
  - Output layer
- Complexity
  - Depends on number of layers and number of neurons in each layer
  - Depends on how the layers are connected (i.e., dense, sparse)
- Perceptron
  - Binary (classifier) that has only a single layer
  - Inputs are fed into network and the weighted sum is calculated (based on weights associated with each of the inputs. This weighted sum is then fed into an activation function which produces the output
  - Uses a sign function as the activation function
  - A multi-layer perceptron is called a neural network
- Multi-layer perceptron:
  - For data that is not linearly-separable, a more complex architecture than the perceptron is required (i.e., more layers are required)
  - Use a sigmoid (logistic) function as the activation function, which outputs a value between 0 and 1 with a s-shaped distribution:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Figure 4: This is a section of an annotator-written summary.

B Human-written Summary Check Guidelines

Two expert annotators score summaries independently, they need to complete 100 subtasks, each of which consists of the source document and human-written summaries. We have developed a guideline for annotators, see Fig 5.

C Experiments

C.1 Fine-tune Models

LED (Beltagy et al., 2020) is a Longformer variant designed to support long document generative sequence-to-sequence tasks. LED incorporates Longformer's attention mechanism, enabling effective handling of long sequence documents. With its attention mechanism that scales linearly, LED can process documents with thousands of tokens, making it suitable for long document generation and processing tasks.

PEGASUS-X (Phang et al., 2023b) is an extension of the PEGASUS model designed to address the challenge of long input summarization tasks. Through additional pretraining on long inputs, PEGASUS-X can handle inputs of up to 16K tokens without requiring model parallel training. By combining a staggered, block-local Transformer with global encoder tokens, PEGASUS-X strikes a good balance between performance and efficiency.

LongT5 (Guo et al., 2022) integrates attention ideas from ETC, and adopts pre-training strategies from PEGASUS into the scalable T5 architecture. It uses a new attention mechanism called Transient Global (TGlobal), which mimics ETC's local/global attention mechanism, but without requiring additional side inputs.

C.2 Unsupervised Models on LecSumm

Unsupervised Models We use **TextRank** (Mihalcea and Tarau, 2004), **SummPip** (Zhao et al., 2020) to evaluate LecSumm. TextRank is a classical extractive summarization model. SummPip is unsupervised multi-document Summarization-based sentence graph compression.

Implement details We used the TextRank in summanlp [13]and SummPip[14] algorithms,and parameters: nb_clusters, nb_words in SummPip are 14 and 20 respectively.

Results Analysis We also evaluate unsupervised models using metric in §4.3. See Table 7, Textrank performs excellently in SummaC scores because its summaries are extracted directly from the original text in a proportional manner, preserving the original sentence structures. This ensures that the generated summaries remain factually

[13] https://github.com/summanlp/textrank
[14] https://github.com/mingzi151/SummPip

Metrics		Model	
		TextRank	SummPip
ROUGE	R-1	12.55	25.91
	R-2	5.73	4.57
	R-L	5.79	11.54
Bertscore	P	77.05	73,98
	R	80.31	79.42
	L	78.61	76.59
SummaC		97.53	58.97
UniEval	coherence	69.61	9.60
	fluency	75.71	26.60
	relevance	67.02	9.83

Table 7: These are unsupervised model evaluation results on LecSumm.

and contextually consistent with the original material. For Summpip, its summaries are generated through sentence clustering and compression. Consequently, Summpip scores lower in fluency and coherence in terms of linguistic features. Additionally, our manual analysis of its generated summaries revealed that the extracted sentences tend to focus more on minor details, which corroborates the results of the automatic evaluation.

D Examples of generated Summaries by models

See Figure 6 and Figure 7, We give some generated summary examples. We can observe that GPT-3.5 basically understands the human needs in the prompt, and its generated summary better aligns with the human needs mentioned in the prompt.

Human Written Summary Check Guidelines

This guideline is intended to give annotators a clear understanding of the task and requirements before manual annotation.Be sure to read the following content carefully.

This task is used to assess the quality of human-written summaries. You need to complete 100 tasks, each of which will provide you with an original document and a human-written summary. You need to score each summary based on four evaluation dimensions, with a maximum score of 25 points for each dimension. The four evaluation dimensions are:

- **Coherence:** The overall quality of all sentences. "The summary should be well-structured and well-organized. It should not just be a collection of related information, but should build coherent information about a topic from one sentence to the next."
- **Consistency:** The factual consistency between the summary and its source. A factually consistent summary only contains statements that are present in the source document.
- **Fluency:** The quality of individual sentences. The sentences in the summary "should not have formatting issues, capitalization errors, or obviously ungrammatical sentences (e.g., fragments, missing parts), which would make the text difficult to read."
- **Relevance:** The selection of important content from the source. The summary should only include important information from the source document. Annotators were instructed to penalize summaries containing redundant and superfluous information.

Please fill in the scores for each dimension in the table below and calculate the total score.

Task Number	Coherence	Consistency	Fluency	Relevance	Total Score

Annotation results are only used for this study. All the information will be anonymized and your personal preferences will not be disclosed.You do not have to bear any responsibility for the risk caused by your annotation results.

Figure 5: This is a human-written summary check guideline for annotators.

Reference

The supervised learning algorithms of KNN and decision tree are discussed in machine learning. KNN compares and classifies new data points in a training set by comparing them to the nearest neighbours. The technique of assigning weights to data points and labelling test point same as the nearest available neighbour is used by KNN for both classification and regression. Like KNN, the Voronoi diagram is built on nearest neighbours. The nearest neighbour method known as the distance-weighted nearest neighbour distributes weights to neighbours based on how far they are from the test point, such as the inverse square of distances. Any training point may have an impact on a specific occurrence.

TextRank

The decision tree can be explained by two entities, namely decision nodes and leaves.\nThe leaves are decisions or final outcomes, and the decision nodes are where the data is displayed.\nSo this was a simple example of decision tree and how can we interpret a decision tree.\nIn the next sections, you're going to learn how to construct a decision tree.\nActivity Classification trees Classification and Regression Trees (CART) is a term introduced by Leo Breiman to refer to decision tree algorithms that can be used for classification or regression predictive modeling problems.\nFor a classification tree, we assign each test instance to the majority class (mode) of the training instances in the region where it belongs.\nYou can consider this action as a being like a data point voting itself into a region which results in selecting the majority.\n

SummPip

a tree , you will use the code example of decision trees are called classification regression model we can be used . shape) output : (10 max_depth_range = range (1 , ' accuracy comparison of training validation set ') figure : plot that results from using the overfit approach from the plot , we can see that as the depth of the tree increases , the decision tree starts to overfit . pyplot as plt our cleaned up dataset is in the data directory : code example # 2

LED

Decision trees are a map of the possible outcomes of a series of related choices. Decision trees can be used to weigh possible actions against one another based on their costs, benefits and probabilities. Decision trees can be used to weigh possible actions against one another based on their costs, benefits and probabilities. Decision trees can be used to weigh possible actions against one another based on their costs, benefits and probabilities. Decision trees typically starts with a single root node, which branches into possible outcomes. Regression trees Classification trees are a type of decision tree algorithm that can be used for classification or regression predictive modeling problems. Regression trees Regression trees are a type of decision tree algorithm that can be used for classification or regression predictive modeling problems.

PEGASUS-X

Decision trees are a popular machine learning algorithm that uses feature selection to determine the most important classification features. DT operates by recursively segmenting the data into subsets based on the most informative features until a stopping criterion is reached. A criterion such as information gain or the Gini index is used to determine which feature is the most informative. At each node of the tree, the feature with the highest score is chosen as the splitting criterion. Feature importance of using Decision Trees (DT) Decision trees are a popular machine learning algorithm that uses feature selection to determine the most important classification features. DT operates by recursively segmenting the data into subsets based on the most informative features until a stopping criterion is reached. A criterion such as information gain or the Gini index is used to determine which feature is the most informative.

Figure 6: These are generated summaries by unsupervised and fine-tuned models.

Zero-shot LLM+Pprompt(L)

Setting up Python involves downloading and installing the Anaconda distribution of Python 3, which simplifies the process of managing editors and packages needed for coding. Instructions are provided for Windows 10 and Mac platforms, as well as guidance on using Jupyter Notebook for coding. The text covers importing modules, matrix operations like addition and subtraction, and matrix multiplication using the NumPy library in Python. It explains scalar operations on matrices, matrix multiplication, and the concept of matrix division, which involves inverting matrices. The importance of conformability in matrix operations is highlighted, along with the conditions for matrix inversion. Code examples demonstrate how to perform these operations in Python using NumPy. The text emphasizes the need for practice and experimentation with coding examples to gain confidence in working with matrices and linear algebra operations. It encourages further exploration of the NumPy library for advanced functionalities. Overall, the text provides a comprehensive guide to setting up Python, working with matrices, and performing common linear algebra operations using Python and NumPy.

Zero-shot LLM+Pprompt(L+C)

Setting up Python involves downloading and installing the Anaconda distribution of Python 3, which simplifies the control of editors and packages needed for data science tasks. Instructions are provided for Windows 10 and Mac platforms, including video tutorials. To start coding, users can open Anaconda and launch Jupyter Notebook to work in a kernel and shell environment. The text also covers importing modules and packages in Python, along with examples of matrix addition, subtraction, multiplication, and division using the NumPy library. Matrix operations such as scalar multiplication, matrix multiplication, and matrix inversion are explained with code examples in Python. The importance of conformability in matrix operations and the determinant of a matrix for determining its invertibility are highlighted. The text emphasizes the need for experimentation with the provided Python coding examples to gain confidence in performing basic matrix operations. Additionally, it encourages further exploration of linear algebra operations using the NumPy library for vectors and matrices.

Zero-shot LLM+Pprompt(L+C+S)

Setting up Python Instructions for downloading and installing Anaconda distribution of Python 3\nTo start coding in Python, you need to install the Anaconda distribution of Python, which provides data science libraries and tools. The process involves downloading and installing Anaconda on different platforms like Windows 10 and Mac. Once installed, you can open Anaconda Navigator to access Jupyter Notebook for coding.\n\n### Python Modules and Packages\nPython allows you to store function definitions in files and import them as modules in your programs. You can import specific functions or entire modules from packages to use in your code. Popular packages like NumPy provide functions for matrix operations and linear algebra.\n\n### Matrix Operations in Python\nUsing the NumPy library, you can perform matrix operations like addition, subtraction, multiplication, and division in Python. Matrix addition and subtraction involve adding or subtracting scalar values or matrices element-wise. Matrix multiplication requires conformable dimensions for the operands, and matrix division involves inverting the matrix.\n\n### Using NumPy for Linear Algebra\nNumPy library simplifies linear algebra operations in Python by providing functions for creating vectors, matrices, and performing common linear algebra operations. By importing NumPy and matplotlib libraries, you can automate the creation of specific types of matrices and visualize data using plots.

Figure 7: These are generated summaries by GPT3.5.