Automatically Generating Lecture Slides with Generative Pretrained Transformers

Matthew Renze
Johns Hopkins University
mrenz1@jhu.edu

Abstract

In this paper, we explore the automation of script-to-slide generation using three Generative Pretrained Transformer (GPT) models. We created a training set of script-slide pairs from a single presentation. Next, we created three GPT models using this training data. We used custom fine-tuning for GPT-3 and few-shot learning for GPT-3.5 and GPT-4. These three models used the on-sample training scripts to generate test slides. To assess the relative performance of each model, we compared the generated slides against the training slides and scripts. GPT-4 outperformed the other models in terms of accuracy, similarity, and relevance. However, it underperformed the other models on runtime performance and cost efficiency. This research demonstrates the potential of using GPT models to automate lecture slide generation, highlighting areas for further investigation and improvement. All code, data, and supplemental materials are available on GitHub[1].

Keywords

Slide Generation, Generative Pretrained Transformer (GPT), Natural Language Processing (NLP)

1. Introduction

The purpose of this research is to automate script-to-slide generation using Generative Pretrained Transformers (GPTs) like GPT-4. Lecture scripts are provided as plain-text documents and presentation slides are generated by the models as markdown documents.

1.1 Background

Online course instructors spend considerable time and effort creating and maintaining lecture videos. One of the key tasks in this process is creating slides to display on-screen during lecture videos. These slides are typically produced manually using presentation software like Microsoft PowerPoint [1]. However, they can also be produced as plain-text documents using Marp markdown [2].

Markdown is a plain-text format for writing structured documents based on standard conventions using a lightweight, human-readable format [3]. The Markdown Presentation Ecosystem (Marp) is a set of tools for creating presentation slides using markdown. The ecosystem includes a language specification based on CommonMark, command-line tools, and IDE plugins to create presentations in markdown and export them into HTML, PDF, and PowerPoint files [2].

[1] https://github.com/matthewrenze/en-705-603/tree/main/paper

The Generative Pre-trained Transformer (GPT) is a large language model (LLM) created by OpenAI. It uses deep learning to generate human-like text. A human user provides the model with a text prompt, and the model produces text to complete the prompt as its output. Several versions of GPT exist, including GPT-3, GPT-3.5, ChatGPT, and GPT-4. Each version has specific capabilities and limitations as well as various pros and cons [4], [5].

GPTs have been used to automate a variety of tasks. These tasks include language translation, interactive tutoring, code generation, and code debugging [4]. More recently, Microsoft announced that GPT-4 will be integrated into productivity tools like Microsoft Office 365. This integration will allow tools like PowerPoint to automatically generate elements for presentation slides in highly structured ways [6].

However, I am currently unaware of any attempts to perform automatic script-to-slide generation using plain-text scripts and markdown slides.

Using plain-text and markdown would allow instructors to edit their lecture content like code using a plain-text editor or integrated development environment (IDE). They could store content in a version control system like Git and GitHub. It would also allow them to automatically render plain-text scripts and markdown slides into lecture videos using Marp, text-to-speech, and text-to-avatar software.

With the aid of automated script-to-slide generation, the entire course creation and update process could be transformed from a manual workflow into an automated "DevOps" pipeline. This change would allow instructors to use modern software development tools and practices to create, edit, render, publish, and update their online course videos [7]

1.2 Objective

My hypothesis is that we can automate the script-to-slide generation process from plain-text scripts to markdown slides using a GPT language model.

Based on previous successes with automated text and code generation, it seems plausible that we should be able to fully automate or at least semi-automate most of the work involved. However, I still anticipate that a human instructor would oversee the process, review the output, and manually edit and revise the generated slides before manually recording or automatically rendering it as a lecture video.

To test this hypothesis, first, I created a dataset of lecture scripts and their corresponding slides from an existing online lecture video. Next, I trained a series of three GPT models with custom fine-tuning and few-shot learning. Then, I used the scripts to generate new slides with the three GPT models. Finally, I analyzed the performance of the three GPT models and compared their results.

1.3 Previous Literature

Currently, there appears to be no existing research papers on script-to-slide generation from plain-text scripts to markdown slides using LLMs like GPT. However, there have been several attempts to perform similar tasks, including document-to-slide generation for research papers formatted as XML markup.

Early slide-generation systems used rule-based heuristic approaches [8]–[11]. However, recent attempts used classic machine learning approaches to identify important sentences and keywords [12]–[14]. Even more recently, deep learning methods have been used to extract and summarize important text from documents [15], [16].

In addition, there have been various attempts to automate other aspects of online course content creation and maintenance. These methods include automatically generating lecture videos and automatically generating titles, outlines, and text for HTML-based courses [17]–[21].

2. Methods

In this section, we will discuss the methods used for data collection, pre-processing, model training, and evaluation in this research project.

2.1 Data Collection

The dataset for this research project was created from an existing online course titled "Intro to Data for Data Science". It was created by the author of this paper and is freely available under a Creative Commons license [22].

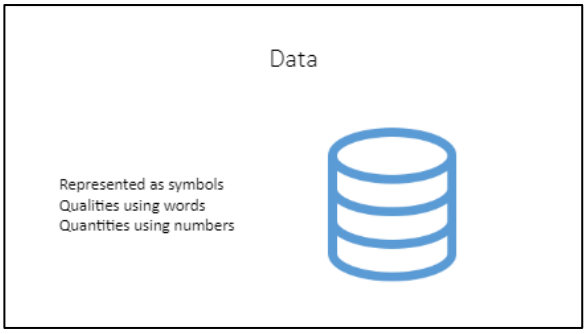
The course is composed of 7 modules. Each module contains multiple sections, for a total of 41 sections. Each of the 41 sections has a corresponding script, slide deck, and video clip. The full course contains a total of 190 slides and corresponding script blocks.

All slides for the course are contained in seven Microsoft PowerPoint files – one for each module. The script for each lecture slide was written in the notes section for the slide. Stage directions and animation cues were also included in the scripts – denoted by square brackets, e.g., "[1]" and "[Pause]".

2.2 Pre-processing

First, the PowerPoint presentation files were programmatically converted into Marp Markdown. Next, the Marp markdown was manually edited to fix any text, image, or formatting errors. Then, alt-image tags were added to each image to act as a textual proxy for each image. Finally, the scripts and corresponding slides were programmatically extracted from the markdown and cleaned.

These data pre-processing steps produced a dataset composed of pairs of files for each script and corresponding slide deck. Each script and slide deck file were named using the corresponding module ID and the section ID to maintain correlation. For example, "1-2 - script.txt" pairs with "1-2 - slide.md".

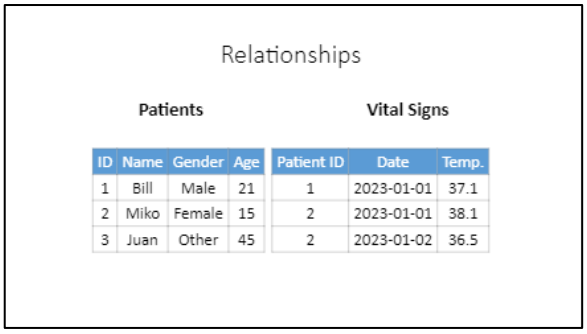



```
<!-- _class: title-two-content-left -->

# Data

Describe observations
Qualities
Quantities

![image](images/462-11.png)
```




```
<!-- _class: title-two-content-comparison -->

# Relationships

## Patients
| ID | Name | Gender | Age |
| -- | ---- | ------ | --- |
| 1  | Bill | Male   | 21  |
| 2  | Miko | Female | 15  |
| 3  | Juan | Other  | 45  |
```

```
<!--
[1] Data are a collection of symbols that describe
observations of the world around us.

They record facts about the natural world that we
live in.

[2] These include descriptions of the qualities of
things in our world ...

... for example, colors, shapes, and textures.

[3] In addition, they include measurements of
quantities of things in our world...

... for example: size, weight, and velocity.
-->
```

```
## Vital Signs
| Patient ID | Date       | Temp. |
| ---------- | ---------- | ----- |
| 1          | 2023-01-01 | 37.1  |
| 2          | 2023-01-01 | 38.1  |
| 2          | 2023-01-02 | 36.5  |

<!--
Now we can navigate the relationship forward from
any patient to get their vital signs.

Or we can navigate the relationship backwards from
a vital sign to get the patient's name and
information.

Relationships allow us to connect data from table
to table in many ways.

However, we'll have to defer these various types
of relationships to a more advanced course on data
science.
-->
```

Figure 1 - Examples of two pre-processed slides: slide image on top with slide markdown and notes below

2.3 Models and Training

Three GPT models were trained and tested using the dataset described above. The three versions of GPT compared were GPT-3, GPT-3.5, and GPT-4. Different versions of GPT require different training methods. In addition, each version of GPT has various capabilities and limitations.

The GPT-3 model was trained via fine-tuning to create a custom model. The training data was composed of all script-slide pairs provided as JSON. The model was trained using the default hyperparameters: batch_size = 1, learning_rate_multiplier = 0.1, n_epochs = 4, prompt_loss_weight = 0.01 [23].

The GPT-3.5 model was trained via few-shot learning. The examples were provided as script-slide pairs in the user prompt at inference time. The max number of tokens for GPT-3.5 is limited to 4096 tokens, so the example set had to be hand-picked and hand-crafted to minimize their token size [5].

The GPT-4 model was also trained via few-shot learning. However, its max token limit is 8,192 tokens [5]. So, roughly double the number of examples were used to train the model via few-shot learning. However, this token limit still required all examples to be hand-crafted to reduce their size.

2.4 Evaluation Methods

The performance of the models was determined by comparing their generated slides to reference slides. Slide accuracy, similarity, and relevance were measured for each generated slide deck. The models were also evaluated on their element-wise performance, runtime performance, and cost efficiency.

A series of performance metrics were calculated to assess each model's performance:

- **Slide Count Accuracy** compares the number of sources slides vs. generated slides. This measure is critical because it determines if the slides and script can be programmatically merged.
- **BLEU (BiLingual Evaluation Understudy) score** measures the similarity of the source slides to the generated slides [24]. BLEU scores were calculated for each slide deck and slide element.
- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score** measures the similarity of the source and generated slides based on matching n-grams. [25].

- **Cosine Similarity** measures the similarity of the source and generated slides irrespective of the size of the documents [26]. It was computed for both entire slide decks as well as slide elements.
- **Content Relevance** measures the cosine similarity between the source *script* and the generated slides [26]. It determines how well the slides capture the essence of the reference script's text.
- A **Composite Score** was computed by scaling all of the previously described metrics from 0 to 1 using min-max scaling and calculating their arithmetic mean.
- **Runtime Duration** was computed by taking the total number of seconds between each inference. GPT-3's custom model training time was not taken into consideration in this metric.
- **Runtime Duration per Token** was computed by dividing the runtime duration by the number of prompt and completion tokens in each inference. This measurement is in milliseconds per token.
- The **Total Cost** of each model was computed by summing the training and inference costs for all three models. GPT-3 had both types of costs, but GPT-3.5 and GPT-4 had only inference costs.
- A **Qualitative Analysis** was performed by manually inspecting the generated slide text and visually inspecting the Marp-rendered slides.

3. Results

In this section, we will discuss the performance of the three GPT models in terms of accuracy, similarity, and relevance. We also discuss their element-wise, runtime, and cost performance.

3.1 Performance Analysis

GPT-4 outperformed the other two models with an average composite score of 0.66. GPT-3 performed second best, with a composite score of 0.65. GPT-3.5 performed the worst at 0.55.

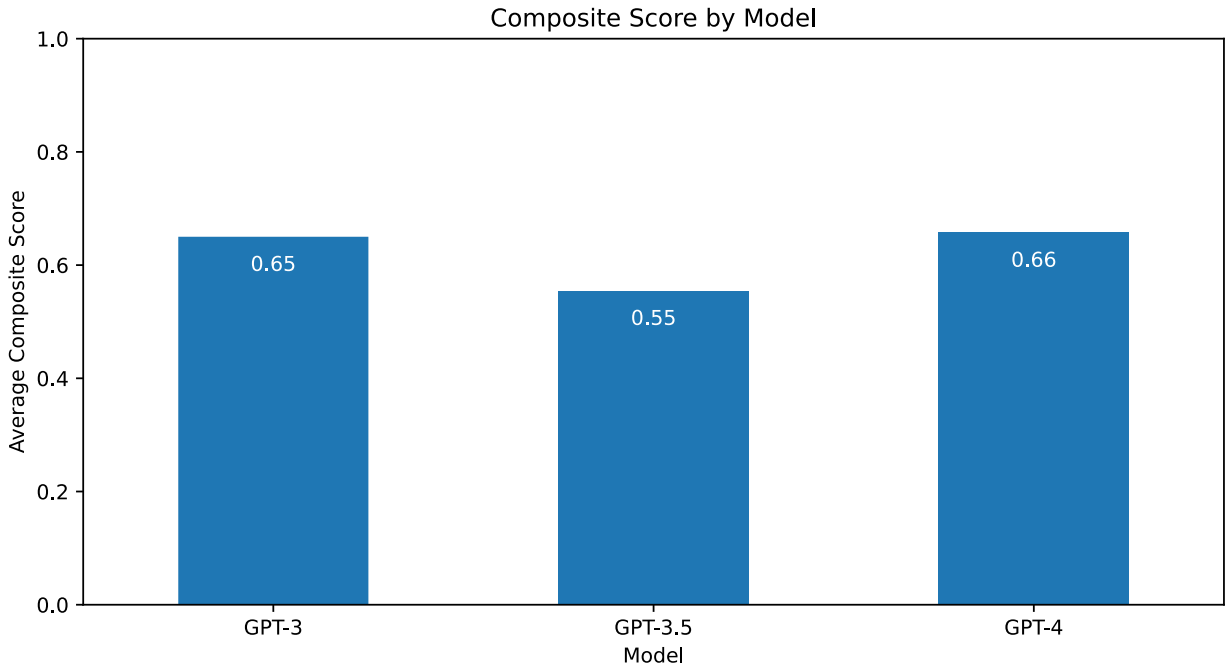


Figure 2 – Average composite score by model

In terms of individual accuracy, similarity, and relevance scores, GPT-4 outperformed all other models on all metrics except for ROUGE-2, where GPT-3 performed best. GPT-3.5 performed the worst on all performance metrics.

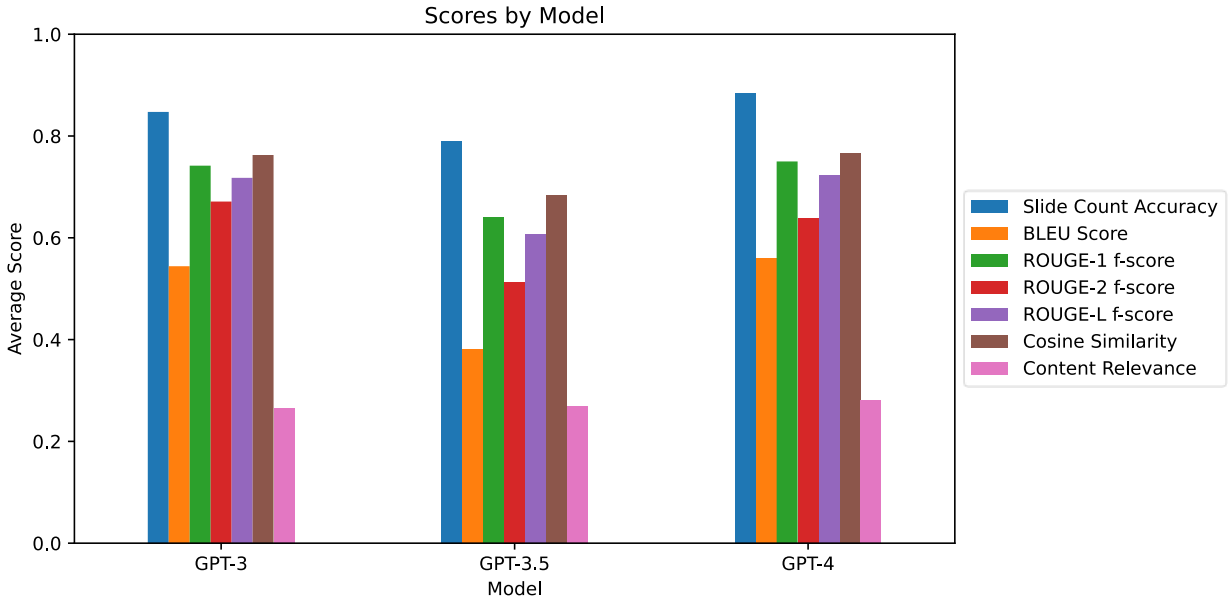


Figure 3 - Average accuracy, similarity, and relevance scores by model

3.2 Element-wise Analysis

All three models scored perfectly with their ability to generate slide headers. All three models scored progressively worse on slide class, title, images, and body – except for GPT-3, which scored higher on generating text in the body of a slide than the remaining slide elements.

No model was able to recreate footers successfully. This is because all slides with footers were contained in a single slide deck which was excluded from element-wise analysis because the count of slides between the generated and reference slide decks did not match.

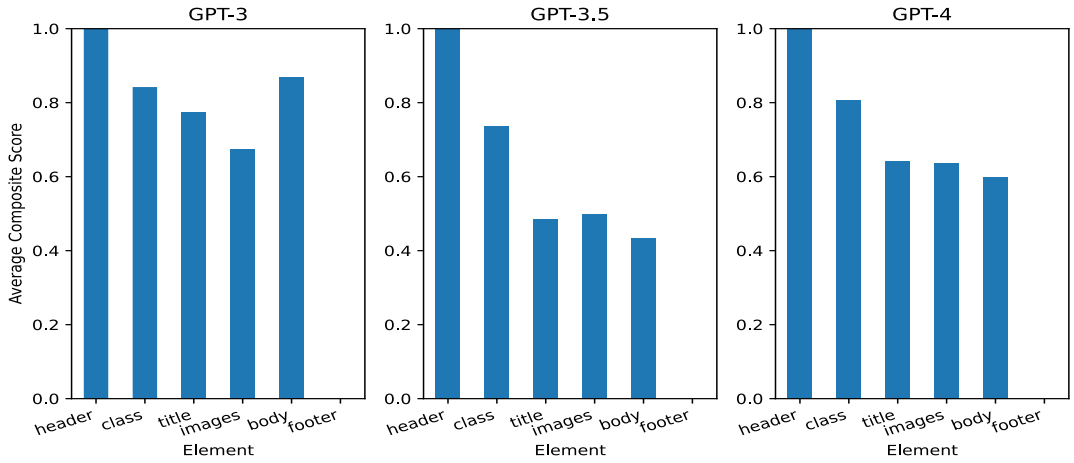


Figure 4 - Average composite score for each element type by model

3.3 Runtime Analysis

GPT-3 was the fastest-performing model, with an average duration of 6.19 seconds per slide deck generated. GPT-3.5 was a close second with an average of 7.00 seconds per inference. GPT-4 was the slowest model, with an average runtime of 27.27 seconds per inference.

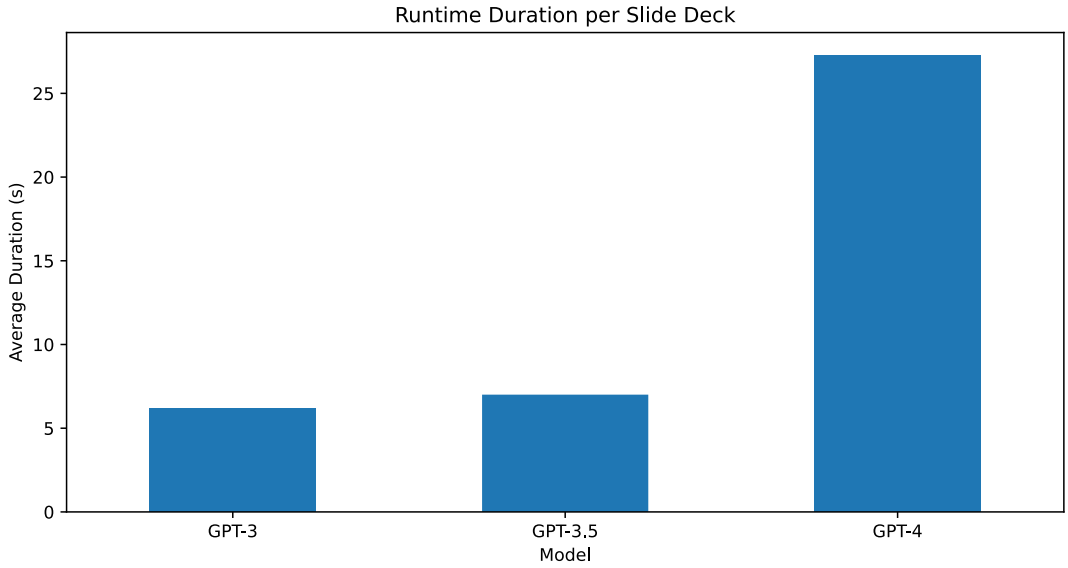


Figure 5 - Average runtime duration per slide deck by model

However, the performance per token tells a very different story. GPT-3.5 performed fastest at 1.98 milliseconds per token. GPT-4 performed second at 3.81 ms per token. GPT-3 performed slowest at 10.01 ms per token. This discrepancy is based on differences in the number of tokens used for inference.

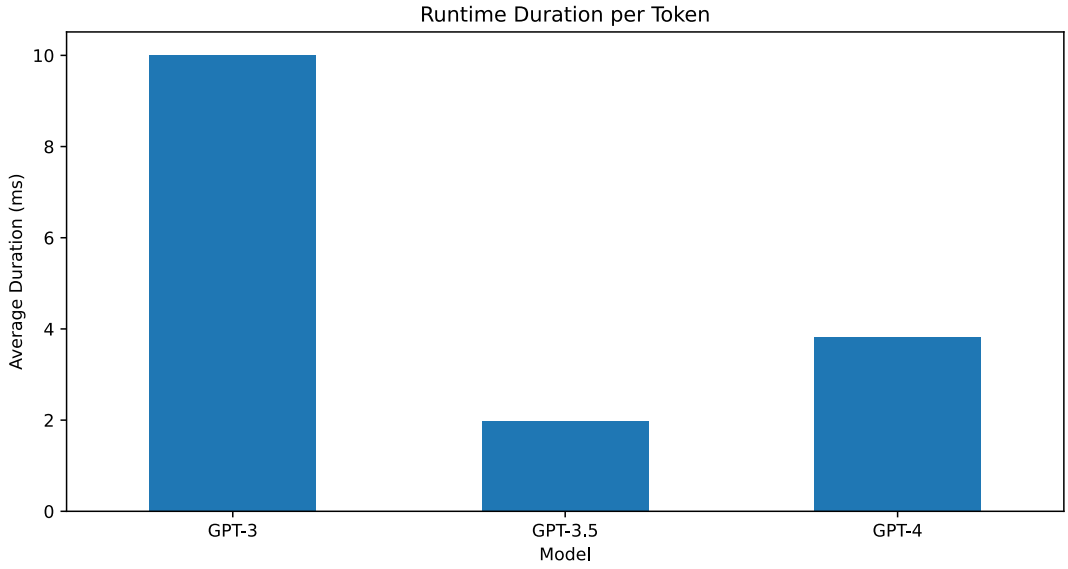


Figure 6 - Average runtime duration per token by model

3.4 Cost Analysis

Overall, GPT-3.5 was significantly less expensive, with a total cost of $0.29 for both few-shot training and inference. GPT-3 was moderately expensive at $6.72 for a custom-trained model and inference. GPT-4 was the most expensive at $8.94 for both few-shot training and inference.

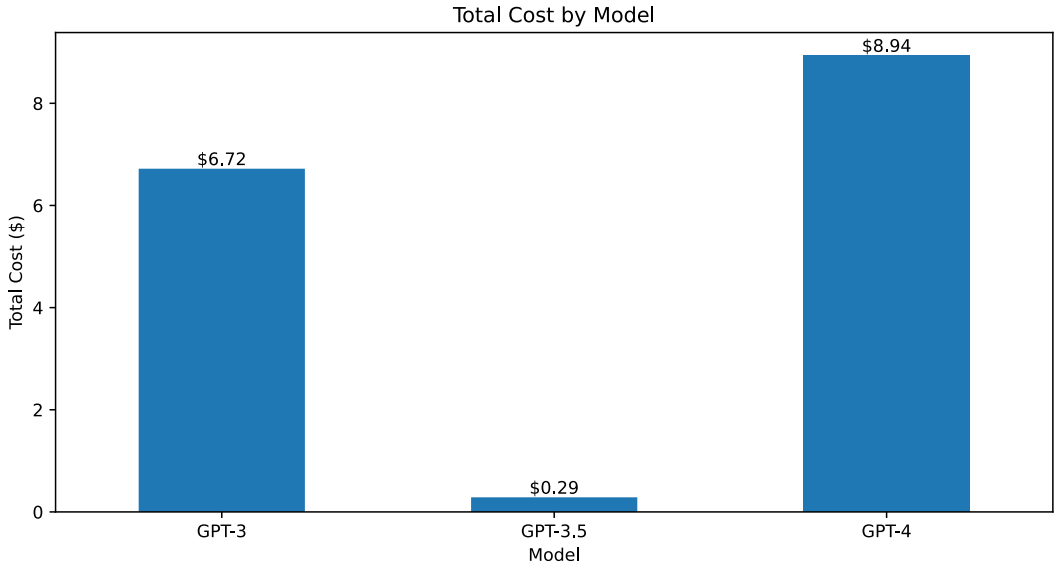
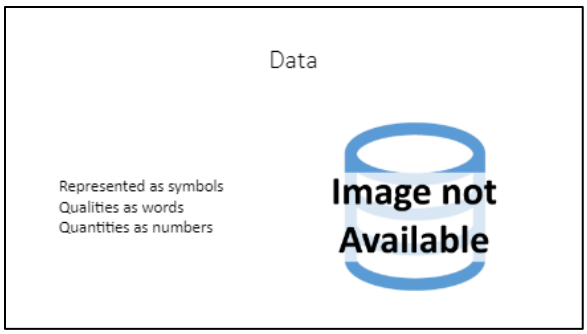


Figure 7 - Total cost by model

3.5 – Qualitative Analysis

Upon review of all 570 slides (i.e. 190 slides × 3 models), only three markdown syntax errors were found. These errors were generated by GPT-3.5 and GPT-4. However, several slides contained extra whitespace and issues with content layout. For example, providing three items in a two-content layout.

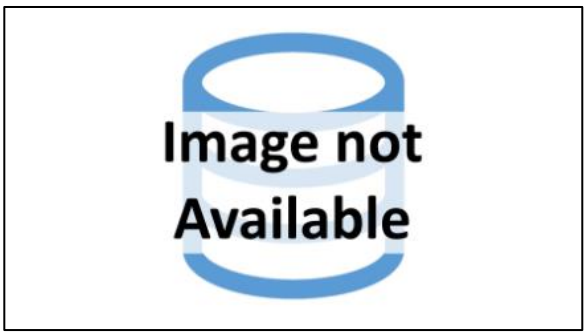



```
<!-- _class: title-two-content-left-center -->

# Data

Represented as symbols
Qualities as words
Quantities as numbers

![image An icon of a database in a flat minimalist
style](images/placeholder.png)
```

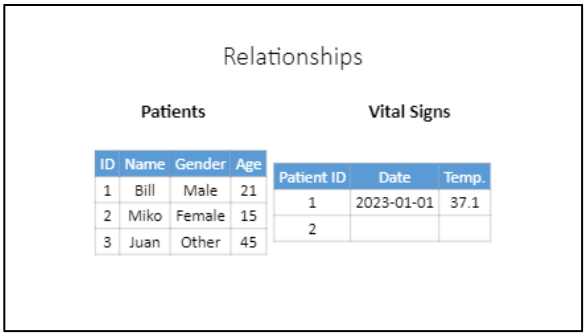



```
<!-- _class: one-pane -->

![bg contain A photo of Herman Hollerith's punch-
card-based "Tabulating
Machine"](images/placeholder.png)

<!-- _footer: Source: Wikipedia: Hollerith Tabulating
Machine -->
```

Figure 8 - Examples of two correctly generated slides: slide image on top with markdown below




```
<!-- _class: one-pane -->

![bg contain A photo of a big-data
```

```
<!-- _class: title-two-content-comparison -->

# Relationships

## Patients
| ID | Name | Gender | Age |
| -- | ---- | ------ | --- |
| 1  | Bill | Male   | 21  |
| 2  | Miko | Female | 15  |
| 3  | Juan | Other  | 45  |

## Vital Signs
| Patient ID | Date       | Temp. |
| ---------- | ---------- | ----- |
| 1          | 2023-01-01 | 37.1  |
| 2
```

Figure 9 - Examples of two incorrectly predicted slides: slide images on top with markdown below

4. Discussion

In this section, we will discuss the limitations of the research project, the implications of this work, and the potential items for future research.

4.1 Limitations

First, this research project was limited in its assessment of model performance because there was only a single presentation available in the dataset. As a result, GPT-3 was trained and tested using the entire dataset. GPT-3.5 was trained using a portion of 5 slide decks. GPT-4 used 12 slide decks for training.

Because of this limited training and test data, all inferences made by the GPT-3 model and a portion of inferences from GPT-3.4 and GPT-4 were made using in-sample data. As a result, the analysis is likely overly optimistic and biased in its estimate of the performance of these models.

Also, with limited training data, the inferences made on the first and last modules are likely skewed. This is because these modules are typically different from those in the middle of an online course. More training examples from the beginning and end of multiple courses would likely benefit performance.

Second, the max token limit for few-shot learning with GPT-3.5 and GPT-4 was a significant limitation. GPT-3.5 was limited to 4,096 tokens, while GPT-4 was limited to 8,192 tokens. This meant that only a small portion of the training data could be used for few-shot learning, which had to be hand-crafted.

However, the currently unreleased GPT-4 32k will allow 32,768 tokens. This increase in tokens would provide much better few-shot learning since more training data could be included in the prompts [5].

Finally, this experiment only considered automated quantitative metrics for analyzing the performance of these models. The perceived quality of presentation slides is highly subjective, and their efficiency as a teaching tool is even more elusive to measure.

A survey with human subjects rating each presentation slide for quality would have benefited this study. In addition, quizzing individual learners using either the generated slides or reference slides would have allowed us to analyze the performance of these models in terms of learning outcomes.

4.2 Implications

This research project demonstrates the feasibility of automating script-to-slide generation with GPT. This work lays a foundation for future research on automated slide creation with markdown.

Automating this task has the potential to reduce the amount of time and effort instructors spend creating and updating slides. This would provide them more time for uniquely human aspects of teaching – for example, more one-on-one time with students and mentorship opportunities.

This research also builds upon existing research in document-to-slide generation for scientific research papers. It moves beyond rule-based heuristic models, machine learning, and deep learning to use large-language models for a similar task.

4.3 Future Research

To improve upon this research, I propose the following follow-up experiments:

First, using more out-of-sample test data would allow for a much more accurate assessment of the performance of these models. Additional training and test data could be created by converting additional presentations into plain-text scripts and markdown slides.

Second, the experiment could be repeated by generating slides one slide at a time rather than as an entire slide deck in one shot. This could improve the performance of the slide-generation model. Slides could also be generated element by element. This could also improve performance even further.

Third, once GPT-4 32k has been released, it would be advantageous to perform a follow-up experiment. Having access to over four times the number of tokens for few-shot examples would likely improve performance considerably – especially on edge cases that are rarely seen in the training set.

Fourth, a survey to determine the subjective quality of each generated slide would provide an additional dimension of performance. A set of quizzes could also be used to assess the learning effectiveness of these automatically generated slides.

Finally, it would be beneficial to continue automating the entire online-lecture creation and maintenance process. Generating lecture slides from a lecture script is only a small piece of the overall workflow. Additional experiments should be run in attempt to automate other tasks instructors routinely perform.

5. Conclusion

This research paper has demonstrated the potential of automating script-to-slide generation using GPTs. GPT-4 outperformed GPT-3 and GPT-3.5 in terms of accuracy, similarity, and relevance. However, it underperformed in terms of runtime performance and cost efficiency.

These findings indicate that GPTs have the potential to generate markdown lecture slides from plain-text lecture scripts effectively. Automating this task could reduce instructors' time and effort in creating and maintaining online course videos.

However, there are several limitations to this study. For example, using a single presentation for the training data required all three models to make predictions using in-sample data. In addition, limits on the maximum number of tokens reduced the few-shot learning examples available to GPT-3.5 and GPT-4.

Future research should explore using larger and more diverse training and test sets. Additional experiments could assess slide-level and element-level generation. Further experiments should also be performed with 32k tokens. Finally, subjective quality and learning effectiveness should be measured.

Overall, the use of GPT models to automate script-to-slide generation shows promise. This research paves the way to automate slide generation and a variety of other tasks performed by instructors involved in creating and maintaining online courses.

6. References

[1] Microsoft, "Microsoft PowerPoint." https://www.microsoft.com/en-us/microsoft-365/powerpoint (accessed Apr. 05, 2023).

[2] Marp Team, "Marp" https://marp.app/ (accessed Apr. 05, 2023).

[3] M. Cone, "The Markdown Guide." https://www.markdownguide.org/ (accessed Apr. 05, 2023).

[4] Open AI, "GPT-4," 2023. https://openai.com/product/gpt-4 (accessed Apr. 05, 2023).

[5] Open AI, "Models." https://platform.openai.com/docs/models/overview (accessed Apr. 05, 2023).

[6] J. Spataro, "Introducing Microsoft 365 Copilot – your copilot for work," Mar. 16, 2023. https://blogs.microsoft.com/blog/2023/03/16/introducing-microsoft-365-copilot-your-copilot-for-work/ (accessed Apr. 05, 2023).

[7] Atlassian, "What is DevOps." https://www.atlassian.com/devops (accessed Apr. 05, 2023).

[8] S. M. Al Masum, M. Ishizuka, and Md. T. Islam, "Auto-Presentation: A Multi-Agent System for Building Automatic Multi-Modal Presentation of a Topic from World Wide Web Information," in *IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IEEE, 2005, pp. 246–249. doi: 10.1109/IAT.2005.2.

[9] Y. Wang and K. Sumiya, "A method for generating presentation slides based on expression styles using document structure," *International Journal of Knowledge and Web Intelligence*, vol. 4, no. 1, p. 93, 2013, doi: 10.1504/IJKWI.2013.052728.

[10] T. Shibata and S. Kurohashi, "Automatic Slide Generation Based on Discourse Structure Analysis," 2005, pp. 754–766. doi: 10.1007/11562214_66.

[11] K. Gokul Prasad, H. Mathivanan, T. V. Greetha, and M. Jayaprakasam, "Document Summarization and Information Extraction for Generation of Presentation Slides," in *2009 International Conference on Advances in Recent Technologies in Communication and Computing*, IEEE, Oct. 2009, pp. 126–128. doi: 10.1109/ARTCom.2009.74.

[12] S. Wang, X. Wan, and S. Du, “Phrase-based presentation slides generation for academic papers,” *AAAI’17: Proceedings of the Thirty-First AAAI Conference on AI*, pp. 196–202, Feb. 2017.

[13] S. Syamili and A. Abraham, “Presentation slides generation from scientific papers using support vector regression,” in *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*, IEEE, Mar. 2017, pp. 286–291.

[14] Y. Hu and X. Wan, “PPSGen: Learning-Based Presentation Slides Generation for Academic Papers,” *IEEE Trans Knowl Data Eng*, vol. 27, no. 4, pp. 1085–1097, Apr. 2015

[15] E. Sun, Y. Hou, D. Wang, Y. Zhang, and N. X. R. Wang, “D2S: Document-to-Slide Generation Via Query-Based Text Summarization,” May 2021.

[16] A. Sefid, J. Wu, P. Mitra, and C. L. Giles, “Automatic slide generation for scientific papers,” *CEUR Workshop Proc*, vol. 2526, pp. 11–16, 2019.

[17] W. Wang, Y. Song, and S. Jha, “AutoLV: Automatic Lecture Video Generator,” *arXiv pre-print*, 2022, Accessed: Apr. 05, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2209.08795

[18] C. Cheong, “Automating Video Generation of Lecture Videos,” in *Proceedings of the 2020 AIS SIGED International Conference on Information Systems Education and Research*, 2020, p. 24. Accessed: Apr. 05, 2023. [Online]. Available: https://aisel.aisnet.org/siged2020/24/

[19] A. Alkhatlan and J. Kalita, “Intelligent Tutoring Systems: A Comprehensive Historical Survey with Recent Developments,” Dec. 2018.

[20] K. linnaosa, “Mini Course Generator - AI Assistant to Speed Up Course Creation.” https://minicoursegenerator.com/ai-assistant/ (accessed Apr. 05, 2023).

[21] Alexander Londo and JazzJune Inc., “World’s First Online Course Created Entirely Using Artificial Intelligence is Now Available,” *EIN Presswire*, 2022.

[22] M. Renze, “Intro to Data for Data Science,” *MatthewRenze.com*, Apr. 29, 2019. https://matthewrenze.com/courses/intro-to-data-for-data-science/ (accessed Apr. 05, 2023).

[23] OpenAI, “OpenAI API - Fine Tuning,” *Open AI*. https://platform.openai.com/docs/guides/fine-tuning/advanced-usage (accessed Apr. 05, 2023).

[24] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU,” in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL ’02*, Morristown, NJ, USA: Association for Computational Linguistics, 2001, p. 311. doi: 10.3115/1073083.1073135.

[25] C.-Y. Lin, “ROUGE: A Package for Automatic Evaluation of summaries,” in *Conference: In Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, 2004, pp. 74–81.

[26] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd edition. Elsevier, 2012. doi: 10.1016/C2009-0-61819-5.