FINE-TUNING VS CONTEXT-INJECTION:

USING GPT FOR AMBIGUOUS QUESTION-ANSWERING ON PROPRIETARY DATA

by

REX VANHORN

(Under the Direction of Frederick Maier)

ABSTRACT

Current large language models (LLMs) have demonstrated abilities that, just a few short years ago, would have seemed impossible e.g., question answering. While LLMs like OpenAI's GPT can do impressive unanticipated things, to maximize their value, the models need to be trained on or have access to additional, often proprietary, data. I compare two popular methods, fine-tuning and context-injection (a specific application of RAG), for integrating additional data into the LLMs for use in the task of question answering. A suite of semantic measurements is evaluated for use in comparing the answers generated by the methods. I use the best performing measurement, Ada 002 with Cosine Similarity, to show that context-injection, using vector embeddings and semantic search, generates answers that are semantically closer to the desired answers, while lacking hallucinations or confabulations. We also provide qualitative and stylistic observations from the experiments further segmenting the two methods.

INDEX WORDS: GPT, Artificial Intelligence, OpenAI, Large Language Models, Oral Data, Data Pipeline, Fine-Tuning, Context- Injection, Prompt-Engineering, STS

FINE-TUNING VS CONTEXT-INJECTION:

USING GPT FOR AMBIGUOUS QUESTION-ANSWERING ON PROPRIETARY DATA

by

REX VANHORN

B.A. Spanish, The Ohio State University, 2002

M.B.A. Finance, The Ohio State University, 2005

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF ARTIFICIAL INTELLIGENCE

ATHENS, GEORGIA

2023

FINE-TUNING VS CONTEXT-INJECTION:

USING GPT FOR AMBIGUOUS QUESTION-ANSWERING ON PROPRIETARY DATA

by

REX VANHORN

Major Professor:	Frederick Maier
Committee:	Jason Anastasopoulos
	Khaled Rasheed
	Yuri Balashov

Electronic Version Approved:

Ron Walcott
Vice Provost for Graduate Education and Dean of the Graduate School
The University of Georgia
December 2023

DEDICATION

This work is dedicated to my patient wife Jackie, who endured too many overly enthusiastic discussions about GPT and the coming artificial intelligence revolution, which in different times would have automatically qualified her for sainthood. The following research was only possible as a direct result of her unwavering support, steadfast encouragement, and gracious, though feigned, interest in the subject.

This work is dedicated, too, to my biggest fan, my mother, who lost her battle with cancer as I was approaching the finish line. It is only through her decades of support and sacrifice that the opportunity to do this work is possible. I hope you can read this from across the river, Mom.

My kids probably supported me, too.

ACKNOWLEDGEMENTS

Special thanks go to my thesis committee: Dr. Fred Maier, Dr. Jason Anastasopoulos, Dr. Yuri Balashov, and Dr. Khaled Rasheed, who accepted and supported the opportunity to investigate new tools, new methods, and new models, and pushed this research to be the best it could be, never accepting anything less.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION AND MOTIVATION

Language models such as OpenAI's GPT-3 (Brown et al., 2020) have exploded in popularity and ability over the past few years, to the point that many anticipate that systems based upon them will have a profound impact on life and industry. (Rodriguez, 2023) These models do not explicitly perform information retrieval but rather use statistical relationships to produce the most-probable subsequent words and phrases to a provided query or prompt. Language models have nonetheless shown themselves to be surprisingly good at tasks such as question answering, or at least providing novel responses within the logical context of the submitted text.

GPT-3 and other similar language models are available pre-trained (out-of-the-box) but the model's internal weights can be fine-tuned by providing additional corpora or other information to the pre-trained model. Until recently, fine-tuning was the only option for influencing an LLM's output. It is still a popular option for question answering (QA), though it may not be the optimal one given its inclination towards "hallucinations" and "catastrophic forgetting" (Kirkpatrick et al., 2017).

Alternatives have been subsequently developed to mitigate the limitations of fine-tuning, including a state-of-the-art technique called 'context-injection' herein. Context-injection is a specific of application of Retrieval-Augmented Generation (RAG)[1], which introduces semantically-similar information into the prompt of an LLM to improve its output. As fine-tuning and context-injection are two popular methods for eliciting answers from LLMs, both

[1] See Chapter 2 for more information on RAG.

could be applied in a given application, though it is likely that one will perform better and be more appropriate in that application than the other. This research seeks to compare the two methods in the task of answering open-ended, ambiguous questions. Ambiguous questions are interpretive or multi-perspectival questions which have multiple, subjectively-correct answers, depending on a given perspective. Doctrinal (e.g., Was Jesus God or just a teacher?) or political questions (e.g., What is the purpose of taxation?) are examples of ambiguous questions.

1.1 Use-case and Dataset

In the case of direct, objectively-answerable questions, standard question sets, such as the SQuAD[2], are available, and their evaluation metric fairly clear; e.g., Exact-Match and F1 for SQuAD. However, this research focuses on the task of open-ended, ambiguously-answerable QA, for which we have not found a standard dataset with matching corpus, and methods of evaluation thereof are more difficult. Therefore, using the pipeline described in this work, we created a dataset with the help of a local pastor, who offered a sufficiently-large oral corpus (available as audio recordings) detailing his church's doctrinal beliefs. The corpus was transcribed, and the pastor provided a set of doctrinal questions and standard answers based on the corpus.[3]

For the research, answers were generated with GPT-3, ChatGPT and GPT-4 using both fine-tuning and context-injection (as well as combinations of the two). These answers are then compared to the standard answers to determine which method delivers responses most semantically similar to the original author's.

[2] See Chapter 3 for more information on the SQuAD

[3] See Chapter 4 for more information on the dataset.

1.2 Research Questions

This thesis focuses on addressing the following questions.

To what extent does fine-tuning influence GPT-3's responses towards the desired output?

GPT-3 contains 175 billion parameters and was originally trained using 45 terabytes of text-based data (Brown et al., 2020, Mahowald et al., 2023). Several studies have shown that, while GPT offers impressive abilities out-of-the-box, performance on some tasks can be improved by fine-tuning on specific data. One goal of this research is to determine to what extent we can effectively train GPT, and by extension other sufficiently-large LLMs, to offer desired responses to open-ended questions using proprietary, abstract, oral data from one information domain. Chapter 6 outlines how fine-tuning directs GPT's output towards a desired response, and the data's impact on that process.

To what extent does the amount of data used in fine-tuning impact performance?

GPT-3 was trained on 45 TB of text, (Mahowald et al., 2020) which roughly equates to 45,000,000,000,000 characters. By comparison, Shakespeare's complete works consist of approximately 3,500,000 characters (Brainly, n.d.). Recent research has demonstrated that it is possible to influence GPT-3's responses by fine tuning using a significantly smaller dataset. A second goal of this research is to determine if the amount of information, in terms of fine-tuning input text, positively correlates to the semantic similarity of GPT's output to a desired response. Chapter 5 describes the process for measuring semantic similarity. Chapter 6 outlines how data volume is related to the semantic similarity of the generated responses to the desired responses.

Does one method consistently generate answers semantically closer to a ground truth? Are the responses grounded in the training corpus, or do they exhibit *hallucination*?

Responses generated through fine tuning undergo a fundamentally different generation process than those generated via context-injection. This research measures the semantic similarity of answers generated through each method against the standard answers; generated answers that are more semantically similar to the standard answers are 'better' than answers with lower semantic similarity. While factual correctness is an important aspect of performance, we do not seek to gauge the truthfulness of the responses, except to the extent that the model/method manifests model hallucination (Raunak et al., 2021, and Ji et al., 2023) which is described in Chapter 2.

Chapter 5 details the two approaches we used to evaluate our method for measuring semantic similarity. We tested 9 state-of-the-art approaches to NLP on the task of comparing similar and dissimilar texts to assess their ability to correlate similar texts. We then applied the best-performing approaches to the MTEB standard test/dataset for semantic textual similarity (STS) and show that our findings match those of the STS community.

Do one method's generated answers more closely resemble the style of the training corpus?

Recent research has demonstrated that fine-tuning can also influence the style of GPT's responses. We seek to measure and assess the issue of authorship of a given text through the text's lexical characteristics and other stylistic attributes using three methods: Covington's (2009) Computerized Propositional Idea Density Rater (version 5.1), *Signature* Stylometric System, and direct comparison.

1.3 Contributions

We created the dataset by programming a Python-based pipeline that automatically accesses the oral data, transcribes it, preprocesses it, and generates the necessary outputs for fine-tuning and context-injection. We designed the pipeline such that it could automatically add new documents to both the fine-tuning and context-injection corpora as they become available. Chapter 4 goes into detail about the design and operation of the pipeline.

The following experiments were performed in support of this research:

1) GPT-3 was subjected to a subset of the SQuAD 2.0 assessment to gauge its performance in machine reading comprehension (MRC). GPT-3 correctly answered all questions correctly on the subset of data, outperforming both the systems and humans tested in the original SQuAD paper. Chapter 3 details the SQuAD dataset and how it was used.
2) GPT-3 was fine-tuned on the first chapter of a collegiate textbook and prompted with the homework questions from that chapter, using fine-tuning and context-injection. The textbook's author then graded the questions as though they were provided by a student of the master-level class. GPT-3 generated C-quality answers to actual homework questions with fine-tuning, and A-quality answers with context-injection. Chapter 3 documents this experiment.
3) To evaluate the optimal model for assessing semantic similarity, two experiments were conducted:
    a. Nine methods for calculating semantic similarity were evaluated using comparisons of various similar and dissimilar large texts (> 512 tokens). In those tests deep learning models (e.g., BERT) offered the best overall performance.

b. Four popular deep learning models were evaluated using the SemEval 2016 (English-only) and 2017 (cross-language) datasets on the task of STS, which showed that OpenAI's Ada 002 embedding model offers the best performance on calculating semantic similarity for our research case. Chapter 5 describes the complete process and datasets used in these evaluations.

4) We evaluated the stylistic properties of the outputs generated through fine-tuning and context-injection and found that fine-tuning resulted in text that was significantly similar in style to the fine-tuning corpus, while context-injection had no appreciable impact on the style of the generated text. See the Appendix for the evaluation and analysis.

5) To evaluate fine-tuning and context-injection as methods of integrating proprietary knowledge into LLMs, which is currently only possible with GPT-3, we considered four cases:

- GPT base, pre-trained with no fine-tuning and no contextual prompt.
- GPT fine-tuned on provided sermon data.
- GPT pre-trained and with context-injection (i.e., providing related context).
- GPT fine-tuned and with context-injection.

Based on these experiments, we conclude that:

- Fine-tuning generates answers that are roughly equivalent to the base GPT's answers, though take the style of the fine-tuning corpus and often exhibit "hallucination" and "catastrophic forgetting."
- There appears to be no correlation between the amount of data and the quality of answers generated; GPT only needs the *right* information to successfully answer a question.
- Context-injection generates answers that are significantly semantically closer to the ideal answer than both fine-tuning and base GPT offer, without hallucination.

GPT-3 with context-injection outperform the fine-tuned models' best answers by 4.1% and all answers on average by 21.5%. On average over all questions, the answers generated by GPT-3 with context-injection were 21.1% semantically closer than those of base GPT-3 when using Ada 002 to judge semantic similarity.

- When approaches are analyzed in terms of performance and computational/cost, context-injection scales more cost- and computationally efficiently. This is discussed in Chapter 6.

6) We evaluated ChatGPT (3.5 Turbo) and GPT-4 with context-injection against their associated base model. In both cases, context-injection produced answers that were semantically closer to the ideal answer. ChatGPT with context-injection on average generated answers 2.2% semantically closer to the standard answer than base ChatGPT, with its greatest margin (Question 5) being 12.1%. GPT-4 with context-injection generated answers on average 5.6% closer than base GPT-4, with its greatest margin being Question 9, at 29.3%. GPT-4 with context-injection also performed best overall, with an average semantic similarity of .75201 and generating the best answer to 12 of the 20 questions over all methods tested. See Chapter 6 for more information.

These experiments and their results are summarized in Chapter 7, along with a discussion of the use cases for which each method is better suited.

CHAPTER 2

LITERATURE REVIEW

This research focuses on the abilities and limitations of fine-tuning compared to context-injection with a corpus of transcribed oral data. The research is conducted using OpenAI's GPT because of its documented task performance abilities in QA, machine reading comprehension (MCR) and in-context learning (ICL). Additionally, this research applies the current SOTA research in semantic search and similarity, demonstrating that the chosen measurement for calculating them is reliable and supported by the community.

Much attention on the GPT language model has been focused on ChatGPT, in our opinion because using ChatGPT is easy and free, as of the time of this writing. As a result, the two are often erroneously conflated, despite the important differences between the two; GPT is a true LLM whereas ChatGPT is a smaller model that was developed through RLHF specifically for conversation. Bubeck et al. (2023) offer a detailed comparison of ChatGPT and GPT-4, which is generalizable to GPT-3.

A research experiment similar to the one proposed herein was conducted by Mosbach et al. (2023) They compared basic fine-tuning and 'vanilla' In-context Learning (ICL) on pattern-matching classification tasks, and find that performance improves for both methods as the model size increases, though fine-tuning outperforms ICL, and more so when trained on additional data. The researchers highlight that the relative performances are highly variant, indicating that their findings are not necessarily generalizable across all tasks. My research supports this finding by exception in that my research shows ICL, through context-injection, outperforms fine-tuning on open QA, and it also mirrors their high-level comparisons of the two methods.

2.1 Fine-tuning as a source of influence

Fine-tuning is the process of iteratively changing a model's internal weights in pursuit of increased performance on a specific task, or to add information to the underlying training corpora of the model. Fine-tuning has been used to perform a specific task, e.g., classification, or influence a model's response using new information. Dunn et al. (2022) fine-tuned GTP-3 using a small dataset of 500 pairs of prompts and completions, and through the fine-tuned model, they were able to extract useful information in the domain of material chemistry. Fine-tuning, however, does not always lead to improved results. Bommarito et al. (2022) attempted to fine-tune GPT-3 to take the Bar Exam, and while they found that GPT-3 passed two sections, their fine-tuning attempts resulted in worse performance than the performance of the base GPT-3 Davinci model[4]. While there is only one method for fine-tuning GPT-3, using OpenAI's fine-tuning API, there are various approaches to fine-tuning, and various hyperparameters that can be modified. In the case of both Bommarito et al. and Dunn et al., they trained their models using training pairs of prompt and completions. Dunn linked English paragraphs containing the relevant information in written form as the prompt, to the desired extracted information, e.g., formula, structure, or application, as the completion. Bommarito et al fine-tunned GPT-3 on various permutations of legal questions (prompt) and explanations (completion) and elicited GPT's response on multiple choice questions, which is substantially different than the fine-tuning concept used in this research. In this research, we fine-tune GPT by submitting only

[4] Bommarito el al. noted that a scarcity of high-quality data may have contributed to the poor results.

the desired training corpus as a completion, leaving the prompt blank. In this manner, the underlying corpus is 'learned' and can be used for text.

David Rozado (2023a) found that ChatGPT has a very left-leaning bias. To counteract that bias, Rozado created a "RightWingGPT" (Rozado, 2023b) through fine-tuning. Rosado reported that he was successful in eliciting the desired, 'right-leaning' output through fine-tuning, though the model is not available to the public. Similarly, Lee and Hsiang (2022) fine-tuned GPT-2 using data from patent claims. They found that they were able to generate 'coherent' claims automatically, though no specific evaluation metrics were applied.

Oniani et al. fine-tuned GPT-2 on a scientific dataset of curated COVID-related information. (Oniani et al., 2020), and were able to generate reliable answers to questions about COVID. The researchers used semantic similarity measurements (embeddings generated via: TF-IDF, BERT, BioBERT and USE) to cull each of the model's answers to the top 5 most-similar sentences. These answers were then evaluated by human Medical experts, and found to be factually-correct answers to the questions. The researchers found, too, that the deep learning embedding model outperformed the others. In contrast to Oniani et al., the generated answers in this research are not modified in any way, and the generated answers are evaluated through automated semantic similarity measurements. We considered manual evaluation by human experts for this research however due to privacy concerns, and to eliminate human-induced subjectivity in the evaluation process (also a factor in Oniani et al.'s research), automated quantitative evaluative measures are used instead, as described in Chapter 5.

Oniani et al.’s research was based on GPT-2, while this research is based on GPT-3. Though GPT-2 employs a smaller model, 1.5 billion parameters, and smaller pre-training corpus, 40 GB, it uses the same pre-training and fine-tuning processes as GPT-3. (Radford et al., 2019). Therefore, Oniani et al.’s results are reasoned to be generalizable to this research on GPT-3, just as this research shall be generalizable to similar and future LLMs.

In another study, Sawicki et al. (2022) showed that GPT-2 developed the ability to write poems in the style of 19th century poets through fine-tuning. They fine-tuned a base GPT-2 model on a collection of poems from Byron and Shelley, and used a BERT classifier to distinguish between the generated text and author’s original text. In a follow-up study by Sawicki et al., (2023) the researchers were unable to duplicate their findings on GPT-3 and GPT-4 using prompt engineering alone, concluding that the fine-tuning process is the source of the generated text’s adopted style. Our research duplicates their findings; that text generated from a fine-tuned model adopts the style of the original training data. We determine this through stylometric analysis with Signature software, propositional idea density, and grammatical analysis.

Several studies have demonstrated the propensity for LLMs to generate nonsensical, confabulated, or other information that is not entailed by the training source. (Ji et al., 2023) This phenomenon is referred to as *hallucination*, and poses an obvious risk whenever the generated text is expected to be reliable. Ji et al. (2023) explore hallucination in generative question-answering (GQA), noting that almost all work in this regard requires human evaluation of correctness, though semantic overlap can be assessed through semantic similarity. This research assesses hallucination indirectly using similarity metrics, as described in Chapter 5, and directly through human evaluation, as described in Appendix A.

2.2 Context-injection

In contrast to fine-tuning, context-injection, a phrase coined in this research to describe the specific retrieval-augmented generation (RAG) process, which uses a pre-trained LLM to answer questions using semantically-related, additional information supplied from the 'training corpus.' RAG has three main components: retrieval source, retrieval metric and integration method. (Li et al. 2020) The retrieval source in our research is the transcribed podcasts. The retrieval metric is cosine similarity calculated using the Ada 002 embedding model, thereby putting context-injection into the Dense Vector retrieval paradigm. The integration method is simple data augmentation, whereby the external data is submitted as part of the prompt. More complex implementations might use Skeleton Extractions, which serves to extract entity or other ground-truth information before further processing submits the information to the LLM. (Cai et al. 2019) We define the process of context-injection in Chapter 4 and discuss potential extensions of the RAG paradigm in Chapter 7.

Context-injection evolved from ICL and relies on MRC. These abilities were lacking in GPT-1 and GPT-2, and only revealed themselves in GPT-3. (Brown et al. 2020, Zhao et al., 2023) Context-injection is new and therefore sparsely studied in scholarly research (though slightly more prominent in industry publications). It may be employed in different ways, depending on applicable semantic evaluation measurements and retrieval constraints.

In-context Learning (ICL) is a few-shot learning technique consisting of the submission of examples of the to-be learned task to the LLM, along with a query, whereby the LLM uses the examples to predict some outcome based on the query. Brown et al. (2020) first described ICL through GPT-3's few-shot learning ability by demonstrating its ability to perform complex tasks for which it was not directly trained. Liu et al. (2021) improve on GPT-3's ICL abilities in their research by improving the in-context examples that are submitted in the prompt. Specifically, they generate embeddings for each instance of the 'training' contextual data, which in the case of QA, is a set of questions and answers. When a test question is presented, a vector embedding is generated for the question, and k-NN search is conducted to find the semantically-closest question/answer pair. *k* question/answer examples are submitted to the LLM along with the test question, in the format of the question/answer pair, leaving the answer blank. This process substantially improves the LLM's few-shot performance across various benchmarks. KATE, the ICL process proposed by Lui et al. is very similar to the process of context-injection proposed in this research. In both cases, semantically similar 'examples' (referred to this research as 'context') are submitted in the prompt along with the test query. The dataset examples/contexts from which the answer is drawn is vectorized and indexed, the *k* closest neighboring examples/contexts are retrieved and submitted to the LLM for processing. Lui et al. use cosine similarity and Euclidean distance as the similarity measurements, and BERT for the embedding models. In this research cosine similarity and Euclidean distance, along with newer, BERT-like embedding models are considered for the mechanism underpinning the semantic search/similarity, though cosine similarity was chosen as the similarity measurement and Ada 002 as the embedding model, the details of which are given in Chapter 5.

A major difference between Lui et al.'s implementation of ICL and context-injection is that ICL relies on the LLMs pattern matching abilities to derive the correct answer. For example, ICL might submit various states and their capitals in expectation of the LLM returning the capital of a different state. In this research, however, we do not submit question and answers, or other QA examples to the LLM, but rather long-text-based, semantically-related information is submitted to the LLM in the prompt, along with a natural language command asking the LLM to consider the information in its answer. As such, this process deviates from ICL and towards machine reading comprehension (MCR), whereby the answer to the query is (presumably) submitted in the prompt, and the LLM is asked to parse the contextual information and return the answer. Baradaran et al. (2020) offers a comprehensive history of MRC, its popular datasets and evaluation metrics. SQuAD (Rajpurkar et al. 2016) continues to be one of the most-popular datasets for evaluating MRC systems. In 2018, NL Net matched human-level performance in MRC tasks on the SQuAD. The same year BERT surpassed it and the following year XL Net improved on BERT's results. (Baradaran et al., 2020) In the context of this research, GPT-3 was tested on a subset on the SQuAD dataset, and determined to have 100% accuracy in MRC. See Chapter 3 for more information.

The most important distinction for this research, though, is that the above-mentioned studies focus on answering discrete, objective questions – i.e., questions with one, universally-accepted truthful answer – in an open domain. This research, however, explores LLMs' ability to answer open-ended, ambiguous questions – i.e., multi-perspectival questions, with many correct answers, depending on the perspective – in a closed domain. For that reason, the datasets and evaluation metrics used by those research teams are inappropriate for this study.

2.2.1 Context-injection example:

Hypothetically, if an encyclopedia company wanted to employ LLMs to answer queries *only using the information from their encyclopedia*, they may consider using context-injection like so: The company generates an embedding vector for each article in their encyclopedia, and those vectors are stored in a database. When a user asks a question, the application generates a vector embedding of the question. The question vector is compared to all the encyclopedia article vectors (this is semantic search). The results of the semantic search – the articles most related to the question – are submitted to the LLM along with the question. The LLM is prompted to answer the question, *basing its response on the articles,* and the answer returned to the user.

2.2.2 Hallucination

Martino et al. (2023) performed a very similar research experiment, comparing fine-tuning to knowledge-injection, and evaluating each method's impact on hallucination. Their process for fine-tuning matched that of this research, and their application of knowledge-injection was analogous to this application of context-injection, with the only major difference being knowledge-injection relies on a knowledge graph for injecting ground truth into the prompt, whereas context-injection directly injects semantically-similar text. Martino et al.'s results mirrored those of this research: the injection of ground truth increased the truthfulness of the generated responses and reduced the instances of hallucination.

Consistent with Hämäläinen et al. (2023), the generated answers in this research will be compared with the standard answers using vector embeddings with cosine similarity. In their research, they explored GPT's ability to generate synthetic data. They prompted GPT with open-ended questions, and compared GPT's generated answers to answers provided by human subjects, using OpenAI's Curie model's embeddings. They used the resulting high cosine similarity values to determine that GPT is capable of generating synthetic data, substantially similar to the answers produced by humans. We apply the same paradigm in this research, except we use OpenAI's Ada 002 embedding model to generate the vector embeddings and calculate semantic similarity, as Ada 002 has since shown to outperform Curie.[5]

[5] "The new model, text-embedding-ada-002, replaces five separate models for text search, text similarity, and code search, and outperforms our previous most capable model, Davinci …" Specifically, Ada 002 outperforms all prior OpenAI models in measuring semantic similarity. https://openai.com/blog/new-and-improved-embedding-model

CHAPTER 3

EXPLORATORY EXPERIMENTS AND PROOF OF CONCENT

To demonstrate that the influencing GPT's output to correctly answer questions is possible, two proof-of-concept experiments were performed, the first using the Stanford Question Answering Dataset (SQuAD), and the second using the first chapter of the textbook, *The Neuroscience of Creativity* (Abraham, 2018). By employing the SQuAD and a simplified version of context-injection, whereby supporting information for answering a question was not sought, but rather provided, we demonstrate that GPT can correctly answer closed-domain, direct questions (where only one correct answer exists), when provided with the necessary information. By applying fine-tuning and context-injection with the textbook, we demonstrate that GPT can also correctly answer open-ended, ambiguous questions, which have a range of acceptable answers.

GPT-3 and the Stanford Question and Answers Dataset

In 2016, Rajpukar et al. compared human and machine performance on reading comprehension, and the humans significantly outperformed the machines. (Rajpurkar et al., 2016) We applied GPT-3 to a subset of the SQuAD test, and GPT-3 outperformed the humans and the models tested by Rajpurkar et al. (2016), earning a 100% score on the subset of questions as judged by a human evaluator, using the official answers from the SQuAD.

3.1 SQuAD Question-Answering

GPT-3 was tested with the first 100 questions from the SQuAD 2.0 using three methods:

1) The question with no additional prompt or information, and a response limit of 250 tokens.

2) The question with no additional prompt or information, though limiting the response to the number of tokens in the SQuAD dataset.
3) The question, including the Wikipedia article which contained the answer to the question (context-injection). The associated article is included in the SQuAD.

Using method (1), GPT correctly answered 61% of the questions in the dataset. When GPT's response was limited to the number of tokens in the SQuAD answer (2), GPT only answered 39% of the questions correctly. Using context-injection GPT was able to answer 100% of the questions correctly. This result surpasses those of all the systems and methods tested by Rajpurkar et al. (2016), *including human subjects*, demonstrating GPT's ability to correctly answer closed-domain, direct questions.

3.1.1 Computational Costs

OpenAI does not publish its computational resource information, so we use the financial cost in this research as a proxy for the computational cost of various prompting methods. Because OpenAI charges on a per token basis and context-injection increases the size of the prompt (269.19 tokens per question/answer pair vs 51.87 tokens for methods 1 and 2), it is unsurprising that context-injection is more expensive, costing $0.538 vs $0.104 for methods 1 and 2 over the 100 questions in the experiment.

3.2 Textbook Question-Answering

A similar test was conducted, though this time using a textbook and its review questions. The text of *The Neuroscience of Creativity* textbook was extracted into a text file. For the purposes of this experiment, the citation text, descriptive text, and references were included in this extraction. The text was then used to fine-tune a base, Davinci model.

We submitted the five ‘Further Reading’ questions at the end of Chapter 1 to GPT, and its generated answers were submitted for grading. We used base GPT-3, GPT-3 with fine-tuning and GPT-3 with context-injection in the following manner:

1. The questions were submitted to OpenAI’s Curie model with low temperature and the default limit on the completion (response) of 256 tokens.
2. The questions were submitted to OpenAI’s Davinci model, which is the most powerful model, with a low temperature and the default limit on the completion response of 256 tokens.
3. The Davinci model was fine-tuned on the text in the first chapter. The questions were submitted to this fine-tuned model exactly as they were asked in the book, with no context or special formatting or prompt engineering. The default token limit of 256 tokens was used to cap the response.
4. The same experiment in (3) was performed, except that the token limit for the response completion was set to the number of tokens returned by the base (not fine-tuned) Davinci model. This allowed for a direct comparison of the responses in experiments (2) and (3).
5. The questions were submitted to the base Davinci model along with the semantically-related sections of the textbook. Davinci has a token limit of 4096 tokens per prompt and response. As a result, only the top three overlapping sections were submitted as context to GPT in this experiment to avoid exceeding the technical limits.

In this experiment, the prompt was engineered in this format:

"""
Considering the information in the CONTEXT, answer the following question.
CONTEXT:
<The sections from the textbook with the highest overlapping semantic similarity were directly copied here.>
"""
If there is not sufficient information in CONTEXT to answer the question, say so.
QUESTION:
<The question to be answered is entered here.>

3.2.1 Textbook Results

GPT-3 Curie's responses to the five questions (1) were manually compared to GPT-3 Davinci's responses to the same five questions under the same circumstances (2). We found that the answers provided by GPT-3 Davinci were more logical, more complete, and generally directly answered the questions, whereas GPT-3 Curie's responses tended to ramble and did not directly answer the question.

GPT-3 Davinci's answers to the five questions as described above for test (3) and (4) were submitted to the textbook author for review. The author evaluated the answers as though they were submitted by students in the associated course, and concluded that the length-limited responses were generally superior to the longer responses, noting that the unlimited responses had a noticeable tendency to ramble, which invariably resulted in an answer that deviated from the question.

Finally, the length-limited responses which were generated with the fine-tuned model and through context-injection, were submitted to the author for manual grading. The fine-tuned responses received grades ranging from F to C, with an average grade of D, while the responses generated using context-injection received grades ranging from F to A-, earning an average grade of B.

3.3 Conclusions

The datasets for these two experiments are small, but still support for the following conclusions:

1) GPT (Davinci) can provide logical, coherent answers to open-ended, ambiguous questions using both fine-tuning and context-injection. In this experiment, we generated

C-quality answers to actual homework questions with fine-tuning, and A-quality answers with context-injection.

2) The lengths of the responses that were generated using context-injection are much more varied, ranging from one sentence to a few, but were much more focused answers. The fine-tuned model, in contrast, tended to fill up the response token requirement, no matter how circuitous the word path or tenuous the connection to the question. Further testing to tease out the differences between context-injected and fine-tuned model responses revealed a pronounced degradation in the model's ability to follow directions, which is referred to in the literature as "catastrophic forgetting."

3) The responses offered by the fine-tuned model tracked very closely to the material in the book. The responses were not 'plagiarized', but the wording and phrases closely matched the style and word patterns of the text.

4) The fine-tuned model was prone to "hallucination" as it provided reference citations that did not exist. The answers created with context-injection did not hallucinate.

These experiments demonstrate that GPT-3 is capable of generating answers to open-ended, closed-domain, ambiguous questions with fine-tuning and context-injection.

CHAPTER 4

METHODOLOGICAL DESIGN OF THE EXPERIMENT

The research project compares answers generated by GPT alone, and by GPT with fine-tuning and context-injection. Further, we fine-tuned GPT-3 over 8 successive rounds with 10 documents per round to evaluate the impact of iterative fine-tuning. We presented the author of the training corpus with a list of questions, and received their ideal answers, which are known herein as the 'standard answers.' In the case of fine-tuning, we prompted GPT-3 with the same set of questions after each round of fine-tuning. With context-injection, we prompted GPT once to receive one set of generated answers. The responses from GPT were automatically evaluated against the standard answers for semantic similarity using Ada 002 and cosine similarity. Generated answers with higher cosine similarity were deemed better than those with lower cosine similarity.

4.1 Question Set

The two types of questions that are used to assess the performance of model are general doctrinal questions and specific source questions. The **general doctrinal question** set will consist of a list of 12 open-ended doctrinal questions that are commonly asked about the Christian faith, but that are not specifically addressed in the training corpus. These questions are used to determine the language model's ability to extrapolate 'knowledge' from the original training set when the answers to the questions are not directly present in the source material.

The **specific source question** set consists of 8 open-ended, doctrinally-based questions that are specifically answerable based on the content in the fine-tuning dataset. The specific-source questions directly correspond to one sermon from the ten sermons presented in the respective fine-tuning phase. The purpose of these questions is to gauge the language model's performance answering the questions before the model is fine-tuned on the sermons that address the question, as well as after. These specific source questions will be asked and assessed in each of the 8 fine-tuning rounds, but only one specific source question will be directly addressed by a sermon in the fine-tuning data in each iteration.

4.2 Training Corpus

The senior pastor at a local church, "the author," has provided 80, publicly-available sermons as audio files, and answered the 20 questions described above. These answers are the "ideal," standard answers, against which the models' answers are evaluated. The sermons range in length from approximately 36 minutes to 61.5 minutes, except for three outliers, whose durations ran 10 minutes, 14 minutes and 22 minutes for fine-tuning iteration phases 3, 8 and 2, respectively. The average duration of all sermons is 48.8 minutes, for a total transcribed sermon time of 3,902.5 minutes of data, corresponding to 3.08MB of transcribed text. 3.08MB is small compared to the training corpus for a modern LLM, but still considerably larger than most of the fine-tuning datasets in the studies referenced in Chapter 2.

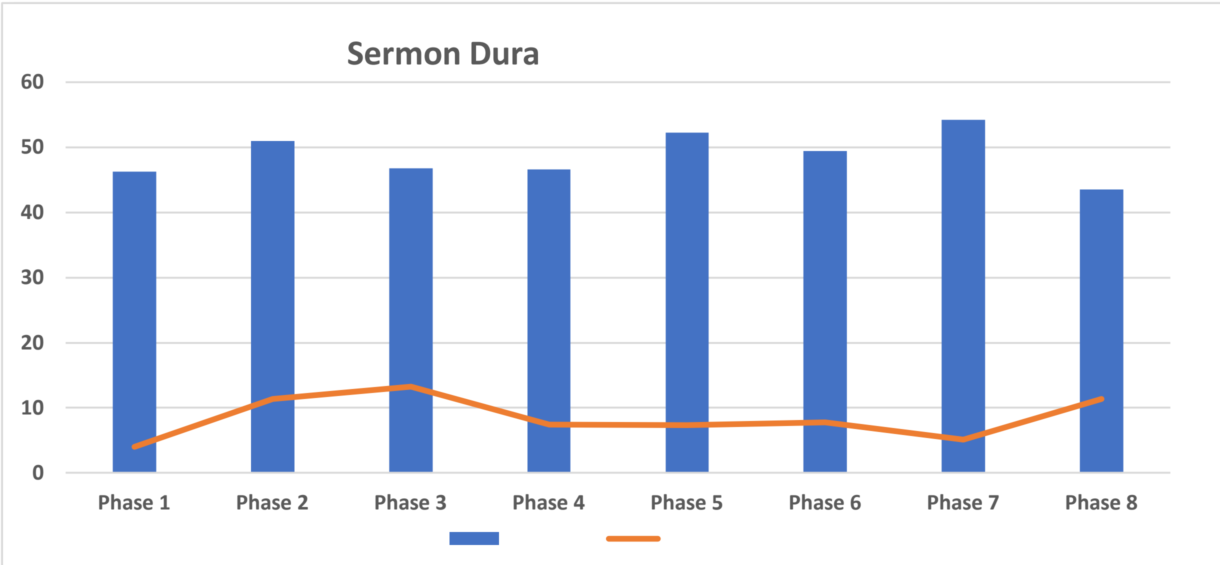


Figure 1 Each fine-tuning phase contains ten sermons. The graph shows the average duration of each of the ten sermons in the eight fine-tuning phases. The orange line shows the relative standard deviation of sermon lengths within each fine-tuning phase.

4.2.1 Pipeline

A Python-based pipeline was created to automatically generate the training corpus. The pipeline first consumes an RSS feed, which provides a list of published sermons. When new sermons are detected, they are submitted to AssemblyAI's public API for transcription, which provides the spoken data as text. The text is preprocessed and broken into a list of sentences. The pipeline then iterates through the sentences and chunks them into units of complete sentences, up to 600 tokens. These chunks are then converted into a. jsonl document used for fine-tuning, and a corresponding vector database for context-injection. The pipeline was programmed so that it could automatically transcribe, fine-tune and vectorize new documents as they became available.

4.2.2 Contextual Knowledge Units

Both fine-tuning and context-injection require breaking the training corpus into chunks (herein called knowledge units); completions for fine-tuning and contexts for context-injection. There are no logical breaks in the transcriptions, which serve to segment the knowledge units by topic, and therefore, there is no programmatic way to ensure that one logical subject is contained in one knowledge unit. The process for generating the corpus text is to fill the knowledge unit, sentence by sentence. This guarantees that the unit contains a complete set of atomic thoughts (i.e., sentences), but does not guarantee that one logically-contiguous idea resides in the unit.

The OpenAI community reports that in practice, the optimal size of the knowledge unit is 600 tokens. Through experimentation 600 tokens has shown generally to be large enough to contain a discrete set of information while being small enough to focus on one topic. This minimizes on average the semantic spillover and dilution of information. Our knowledge units are filled with up to 600 tokens. Semantic segmentation might be one opportunity for focused knowledge units, which is discussed in Chapter 7.

4.2.3 Transcription Error Rate

Each sermon originated as an audio recording, which was automatically transcribed. Eight of the eighty transcribed sermons have been randomly and manually evaluated for fidelity using two calculated error rates: Transcription Error Rate (word-based) and Semantic Error Rate (sentence-based). Together these scores can be combined to create a fidelity score, which provides an overall evaluation of how close the transcribed text matches the original oral text, taking into account the number of errors and their magnitude, and comparing them against both the number of words and number of sentences in the sermon. This review found the word-to-word accuracy of the transcriptions to be greater than 98% correct, and the sentence-level semantic accuracy to be slightly better than 92%. These accuracies are roughly in line with previous studies, which show the human word-to-word transcription accuracy to be approximately 99%, while overall manual transcription errors averaged between 5 and 11%. (Feng et al., 2020) Therefore, our pipeline's transcription is at least as accurate as, if not better than, average human transcription performance.

4.2.4 Tokenization for Fine-Tuning and Context-Injection

OpenAI offers a specific method for fine-tuning GPT-3, which requires the fine-tuning data to be submitted to GPT-3 as a series of prompt/completion pairs. In our research, we fine-tune on a series of completions without a corresponding prompt. Each pair cannot exceed 2049 tokens in total, so each of our completions can have a maximum length of 2049 tokens. In practice, we limit each completion to approximately 600 tokens.

A token is approximately 4 characters, or ¾ word, but is not calculated based on the number of characters or any other physical attribute of the word, but rather using the tokenizing mechanism: Hugging Face's GPT2Tokenizer for GPT-2 and GPT-3, and Tiktoken for GPT-4.

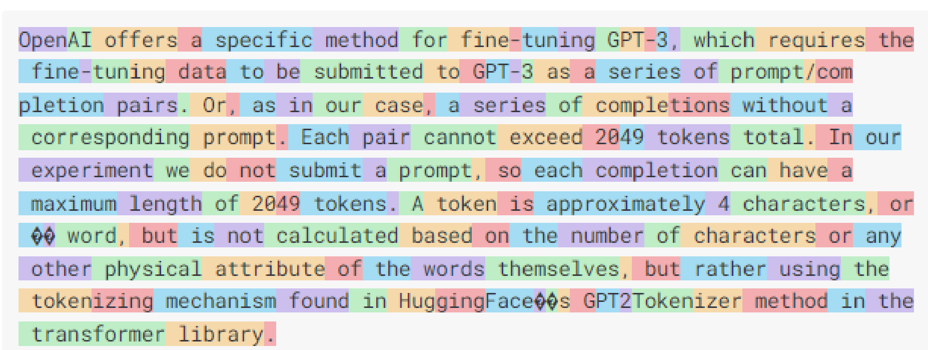

OpenAI offers a specific method for fine-tuning GPT-3, which requires the fine-tuning data to be submitted to GPT-3 as a series of prompt/completion pairs. Or, as in our case, a series of completions without a corresponding prompt. Each pair cannot exceed 2049 tokens total. In our experiment we do not submit a prompt, so each completion can have a maximum length of 2049 tokens. A token is approximately 4 characters, or �� word, but is not calculated based on the number of characters or any other physical attribute of the words themselves, but rather using the tokenizing mechanism found in HuggingFace��s GPT2Tokenizer method in the transformer library.

Figure 2: Example of the tokenization process using OpenAI's online tokenizer for GPT-3, showing how text is tokenized. The tokenized text contains 657 characters broken into 149 tokens. Each change in color represents a new token.

As GPT operates on tokens, not words, the research reports sizes and limits for prompts, completions, and contextual knowledge units in tokens. The knowledge units, for example, which are the completions in the case of fine-tunings and the sets of contexts in the case of context-injection, are broken into groups of 600 tokens. Hugging Face's GPT2TokenizerFast was used to tokenize text for fine-tuning. GPT-4 uses the Tiktoken tokenizer.

Example Python code for using the GPT-3 tokenizers:

```
from transformers import GPT2Tokenizer
from transformers import GPT2TokenizerFast
# instantiate the two tokenizer (fast and accurate)
tokenizerFast = GPT2TokenizerFast.from_pretrained("gpt2")
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
```

4.3 Fine-tuning

GPT-3 provides the ability to fine-tune the base, pre-trained model by submitting additional text for (re-)training of an existing base model. As GPT is a proprietary system, the exact effect of fine-tuning on the underlying model is not published. This fact is acknowledged in several of the studies noted in Chapter 2. The fine-tuning operation does (at least partially) retrain the model with the fine-tuning corpus, and the internal model weights are adjusted accordingly. We do know the rigidity of the model is at least partially determined by the number of fine-tuning epochs. OpenAI's documentation recommends 4 epochs of fine-tuning for general use, 2 epochs for "creative" application of the fine-tuning corpus, and more than 4 epochs if we desire the model to return a specific completion to a specific prompt. The hyperparameters were experimentally varied and the defaults found to offer the best overall results for our research, including 4 training epochs. Sawicki et al. (2022) also found 4 fine-tuning epochs to be the optimal number of training epochs to avoid overfitting.

4.3.1 Models

OpenAI offers three main families of models: GPT-3, ChatGPT, and GPT-4. GPT-3 consists of four individual models (listed in increasing order of ability): Ada, Babbage, Curie and Davinci. As only GPT-3 offers fine-tuning, all fine-tuning took place with GPT-3. There are four variations to the GPT-4 models, which include the original and updated versions of the GPT-4 model, each with 8K and 32K token limits. Similarly, ChatGPT, which is also known as GPT-3.5 Turbo, offers the same four styles of models as GPT-4, which differ based on training date and token lengths.

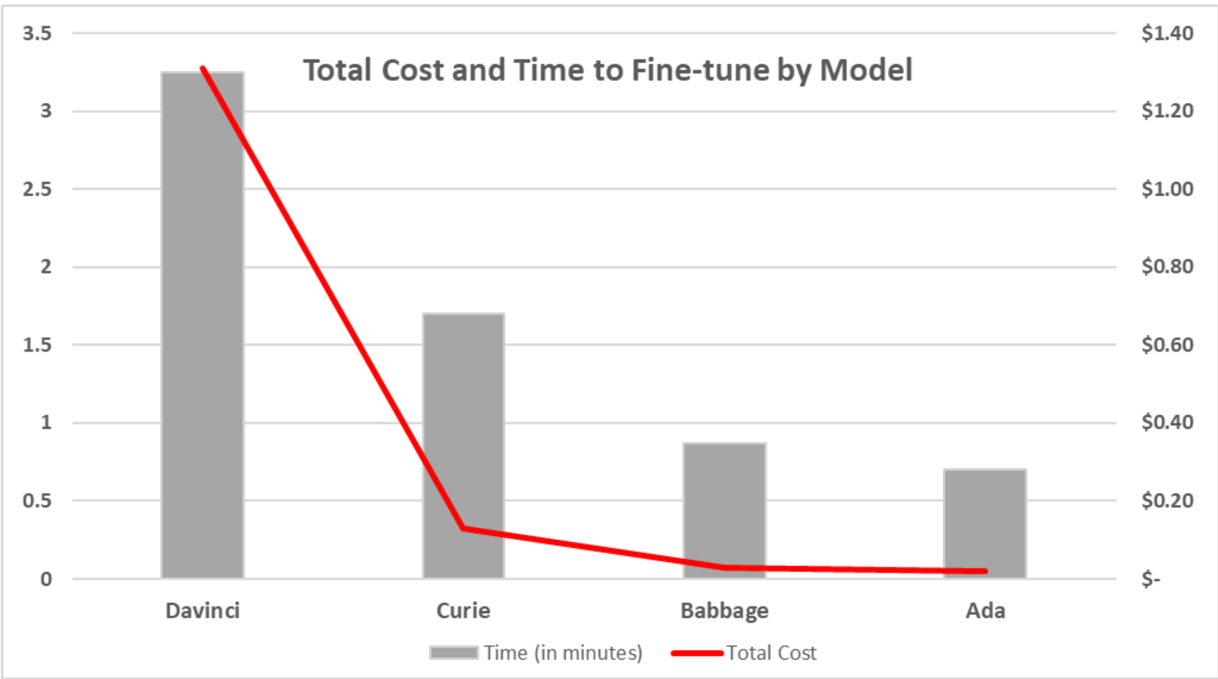


Figure 3: The time and financial cost of fine-tuning each model with the first chapter of the textbook. The fine-tuning document contained 18 prompt/completion pairs (the prompts were blank) consisting of 53,340 characters for a total file size of 53KB.

OpenAI charges different, higher rates for both the training and usage of fine-tuned models, as shown in the table below.

Table 1: The current financial cost to train and use a GPT-3 model, by model type. The training costs are per epoch, so training a model over 10 epochs will cost $0.30 per 1000 tokens. The usage cost is per token in request/response. These costs are used as a proxy for computational cost.

GPT-3 Model Name	Fine-Tuned Model Training	Fine-Tuned Model Usage
Ada	$0.0004 / 1K token	$0.0004 / 1K token
Babbage	$0.0006 / 1K token	$0.0004 / 1K token
Curie	$0.0030 / 1K token	$0.0004 / 1K token
Davinci	$0.0300 / 1K token	$0.0004 / 1K token

4.3.2 Epochs

An epoch is one full cycle through the fine-tuning dataset. The following graph shows the total time and cost of fine-tuning the same dataset over 10 epochs.

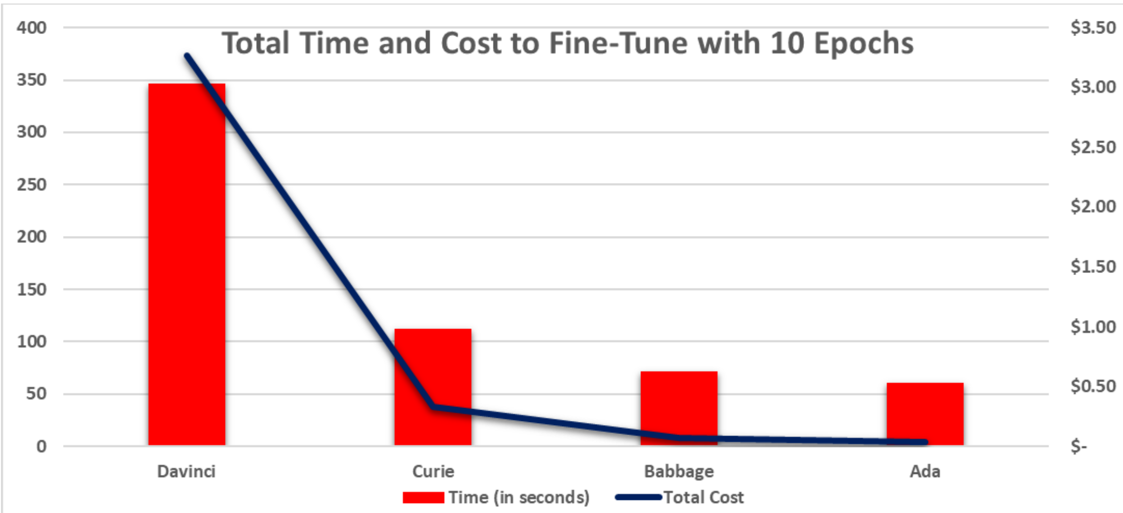


Figure 4: The graph shows the time and financial cost of fine-tuning each model with the same dataset, at 10 epochs. The costs to fine-tune the models, in order, were $3.27, $0.33, $0.7 and $0.4, and the times to train each model were 5:13, 1:52, 0:72, and 1:01 respectively, at 10 epochs. The fine-tuning costs were roughly 2.5 times higher than the cost to fine-tune at 4 epochs, which is consistent with OpenAI's pricing model.

In order to gauge ability, each base-model was fine-tuned on the *Creativity* text described above, then the prompt, *What is creativity?* was submitted to each model, using a temperature of 0.15 and a maximum response of 50 tokens. Each model's response is displayed in the table below.

Table 2: The output by model type given the prompt, *What is creativity?*, with a limit of 50 tokens at 4 epochs.

Model	Model Output at 4 Epochs (limit of 50 tokens)
Davinci	What are the defining features of creative outputs? How do we recognize creativity when we see it? How do we measure creativity? These are the questions that we will need to address in order to understand the nature of creativity. Let us begin by taking a
Curie	What is creativity? What is creativity? What is creativity? What is creativity? What is creativity? What is creativity? What is creativity? What is creativity?
Babbage	What is originality? What is innovation? What is surprise? What is surprise? What is originality? What is innovation? What is surprise? What is originality? What is innovation? What is surprise? What is originality? What is
Ada	"Creativity is the ability to create and to use ideas and knowledge to produce and realize desired outcomes." "What is knowledge?" "Knowledge is the ability to recognize,

To further compare the base models, and their potential for fine-tuning, each model was fine-tuned on the *Creativity* dataset described above for 10 epochs. Each fine-tuned models' response to the question *What is creativity?* is displayed in the following table:

Table 3: The output by model type given the prompt, What is creativity?, with a limit of 50 tokens at 10 epochs, showing the impact of additional epochs of training on each model type's output. Davinci's output is significantly more coherent. The output parrots the text, and exhibits hallucination.

Model	Model Output at 10 Epochs (limit of 50 tokens)
Davinci	The simplest way to approach this question is to begin with a definition of creativity that will serve as a starting point. There are several definitions of creativity that are used in the literature (Dacey & Madaus, 1969)*, and there is considerable agreement regarding… **This is the referenced line in the original text: While several researchers have attempted to define creativity accurately and comprehensively across the decades (Dacey & Madaus, 1969),*
Curie	Is creativity a quality or an ability? Is creativity a skill or a capacity? Is creativity a phenomenon or an enterprise? And if so, what is it? And what is it supposed to be? And how do we determine if an entity or an…
Babbage	What makes something creative? What is original? What is true? What is real? What is true? What is real? What is true? What is original? What is original? What is true? What is original? What is true? What…
Ada	What is discovery? What is innovation? What is success? What is failure? What is opportunity? What is necessity? What is malus? What is malar? What is malo? What is mane? What is mare? What…

4.3.3 Impact of Epochs on Output

The base Davinci model was experimentally, repeatedly fine-tuned on a pair of the same prompt and completion, over 15 epochs to demonstrate the output drift at each training step. A series of prompts and responses were recorded at every epoch. Four epochs offered the best responses; fewer epochs 'ignored' the fine-tuning corpus, while too many epochs resulted in simply parroting the corpus. OpenAI recommends that the optimal number of training epochs is between 2 epochs, for more creative responses, and 5 epochs for more deterministic responses. GPT-3's default value for fine-tuning is 4 epochs, which offered the best output in our experiment. Unless otherwise noted, all fine-tuning for this research was performed over 4 epochs.

4.4 Context-Injection

In this research, partially in relation to the transcribed nature of the dataset, context-injection follows this process:

1. The pipeline creates the knowledge units.
2. The knowledge units are vectorized using the Ada embedding model.
3. The vector and the corresponding knowledge unit text are stored in a table.
4. At run-time:
    a. the query question is vectorized,
    b. the query vector is compared to the vectors of all knowledge units,
    c. the knowledge units for the *k* nearest neighbors are returned, and
    d. these *k* knowledge units are submitted as context in the prompt.
5. The LLM uses the MRC and ICL with the context to answer the query question.

Context-injection does not require pre-training, but does require vectoring the knowledge units and submitting significantly more tokens into the prompt than when using a fine-tuned model. The two costs associated with context-injection are vectorization and per-token prompting costs. Vectorization is possible with free models, though they are subject to performance and token limits. Vectorizing with Ada 002 currently costs $0.0001 per 1K tokens, which is significantly cheaper than fine-tuning. The overall costs of context-injection are discussed in Chapter 6.

CHAPTER 5

QUANTITATIVE AUTOMATED MEASUREMENTS

We assess several approaches to semantic similarity to determine the best measurement of semantic similarity. The following table shows the most common approaches for measuring text-/semantic similarity, as proposed in Han, et al (2021). Of the proposed measurements, 9 were considered, as described below. Each approach was used to compare two sets of texts. The texts were both positivity related (including cross-language assessments), and either partially related or completely unrelated. Ada 002 embedding engine offered the best performance.

5.1 Overview of Similarity Measurements

Table 4: List of most-common measurements of similarity by lexical ('corpus'), knowledge-based, and deep-learning, adapted from Han et al. (2019). Only the considered methods shown. The knowledge-based similarity measurements references in Han do not apply to this research, so they are excluded. * Citations at time of writing. Otherwise, citations at time of Han et al. publication.

Similarity Type	Method	Year	Published	Citations
Lexical/Corpus-based Similarity	TF-IDF	1988	Information processing & management	12996*
	Jaccard	1901	Bull Soc Vaudoise Sci Nat	4455*
	LSA	1990	Journal of the American Society for Info. Science	12000
	LDA	2003	Journal for Machine Learning Research	7471
	Word2Vec	2013	ICLR	1358
	Doc2Vec	2014	International Conference on Machine Learning	1438
Knowledge-based Similarity	N/A	N/A	N/A	N/A
Deep Learning-based Similarity	BERT	2019	ARXIV	72926*
	Ada	2023	OpenAI	-

From the list provided in Han et al. (2019), the following measurements were evaluated for their performance in determining semantic similarity.

5.1.1 Lexical Similarity

1. TF-IDF: Originally described in 'Term-weighting approaches in automatic text retrieval' (Salton et al., 1988), it calculates the relative importance of a term within a set of documents, or to generate a vector that represents a document. (Not listed)

2. Jaccard similarity: Originally described in 'Étude comparative de la distribution florale dans une portion des Alpes et des Jura,' (Jaccard, 1901), it is a general mathematical operation that measures the similarity of two sets (Not listed)
3. Euclidean distance: Attributed to Greek mathematician, Euclid, it uses geometry to calculate the distance between two points in in a 2+ dimensional space. (Not listed)
4. LSA – Latent Semantic Analysis. As described in Kontostathis et al. (2006), LSA is primarily used for topic modeling and not appropriate for our intended purpose.
5. LDA – Latent Dirichlet Allocation. As described in Blei et al. (2003) primarily used for topic modeling and not appropriate for our intended purpose.
6. Word2Vec – Originally described by Mikolov et al. (2013) and elucidated by Jatnika et al. (2019), Word2Vec determines semantic relationships between words and is therefore not appropriate for this research.
7. Doc2Vec – An extension to Word2Vec, described in Le et al. (2014), and is recognized as inferior, for our purposes, to BERT and successors. (Mendsaikhan et al., 2020)

5.1.2 Semantic Neural Network-based Similarity

8. OpenAI Ada 002 embeddings with cosine similarity.
9. Hugging Face models, which are successors to the BERT[6] (Kenton et al., 2018) architecture for embeddings, with cosine similarity. The three most-popular Hugging Face embedding models at the time of this writing were chosen for this research:
    a) all-MiniLM-L6-v2 [7] (based on the MiniLM architecture)

[6] The three selected models are not technically BERT-based models, but rather successors to the BERT architecture. They are referred to as BERT models herein to serve as newer examples to Han's deep-learning models, and to distinguish them from OpenAI's Ada model, which uses OpenAI's proprietary libraries for interaction (whereas the "BERT" models exist in and interoperate with the Hugging Face ecosystem).
[7] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

b) all-mpnet-base-v2 [8] (based on the MPNet architecture)

c) msmarco-MiniLM-L-6-v3 [9] (based on the MiniLM architecture)

5.2 Experimental Paradigm for Similarity

A two-pronged approach to assessing the application of semantic similarity was used. First, the approaches described above were applied to one chapter of different translations of the Christian Bible, as no dedicated datasets for long-text evaluation seem to exist. The Christian Bible is the most-translated (Hare, 2014), and the most-carefully translated, book in the world (Noah, 2005), and the various translations of the Bible all *say* the same thing, though use different words (barring additions and exceptions). Despite the purpose and history of the various translations, the texts have completely similar *meanings*, which offers the opportunity to compare the performance of these different semantic similarity measurements.

The first chapter of the Book of Mark was chosen as the representative text from the different translations of the Bible. The following translations were compared: ASV (American Standard Version), BBE (Basic Bible in English), KJV (King James Version), NIV (New International Version), NKJV (New King James Version), NLT (New Living Translation), and WEB (World English Bible).

Additionally, the following additional texts are added to the experimental paradigm:

- NIV – 1st 10: only first ten verses of Mark 1, to test the impact of differing text lengths on the similarity measurement.

[8] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[9] https://huggingface.co/sentence-transformers/msmarco-MiniLM-L-6-v3

- NIV – 1st 10 (Opp.): first ten verses of Mark 1 but modified be oppose/negate the original text, and is used to assess the models' output on partially-related texts.
- HFA, Hoffnung für Alle, which is a German-language translation of the Bible, which was added to measure how the approaches handle texts in different languages.
- SCH2000, SCHLACHTER 2000, which is a German-language Bible written based on the original Greek translation of the Bible.
- LBLA, Biblia de Las Americas, which is a Spanish-language Bible, published in 1986.
- NBV, Nueva Biblia Viva, which is another Spanish-language Bible translation which was translated into contemporary Spanish.
- U.S. Constitution; the beginning of the Constitution, as the same length as the first chapter of Mark, to control the impact that differing lengths have on the similarity calculation, and serves as a negative test.

5.3 Similarity Results

The comparisons were made using Natural Language Toolkit (NLTK) and Sklearn's standard implementations of TF-IDF, Euclidean Distance and Jaccard similarity between all the tested texts. All the lexical measurements report wide semantic variation among all the English translations, similar variation in the partially-related and unrelated texts, and little relatedness between the tests in different languages. The lexical approaches are therefore poor measures of semantic similarity for this research's purposes.

Similar experiments were conducted using the deep learning embedding models, described above, calculating the similarities between the Bible translations and negative tests, using cosine similarity as the output measurement. From these experiments we found that these embedding models are more robust than the lexical models, as they correctly report a high degree of similarity between related texts, low similarity between unrelated/partially-related texts, and in the case of Ada 002, a high degree of similarity between similar texts in different languages.[10]

Cosine Similarity	ASV	BBE	KJV	NIV	NKJV	NLT	WEB	NIV - 1st 10	NIV (Opp.)	Constitution	HFA	SCH2000	LBLA	NBV
ASV	1.000000	0.943471	0.994491	0.944588	0.958063	0.905694	0.998300	0.944588	0.844569	0.779384	0.847810	0.913400	0.888264	0.883308
BBE	0.943471	1.000000	0.939109	0.944557	0.917588	0.914662	0.943146	0.944557	0.846018	0.780585	0.866297	0.882782	0.871625	0.899221
KJV	0.994491	0.939109	1.000000	0.945514	0.958016	0.904872	0.992668	0.945514	0.845682	0.779462	0.844020	0.907836	0.882234	0.878486
NIV	0.944588	0.944557	0.945514	1.000000	0.923676	0.946242	0.942548	1.000000	0.885704	0.770588	0.844086	0.885132	0.858705	0.913425
NKJV	0.958063	0.917588	0.958016	0.923676	1.000000	0.930777	0.956519	0.923676	0.875230	0.777638	0.832444	0.885126	0.857879	0.860519
NLT	0.905694	0.914662	0.904872	0.946242	0.930777	1.000000	0.903039	0.946242	0.909719	0.774336	0.849771	0.854219	0.837173	0.897788
WEB	0.998300	0.943146	0.992668	0.942548	0.956519	0.903039	1.000000	0.942548	0.840505	0.781374	0.847877	0.915256	0.888370	0.882916
NIV - 1st 10	0.944588	0.944557	0.945514	1.000000	0.923676	0.946242	0.942548	1.000000	0.885704	0.770588	0.844086	0.885132	0.858705	0.913425
NIV - 1st 10 (Opp.)	0.844569	0.846018	0.845682	0.885704	0.875230	0.909719	0.840505	0.885704	1.000000	0.767044	0.812852	0.817174	0.794176	0.830995
Constitution	0.779384	0.780585	0.779462	0.770588	0.777638	0.774336	0.781374	0.770588	0.767044	1.000000	0.737005	0.758674	0.750825	0.732159
HFA	0.847810	0.866297	0.844020	0.844086	0.832444	0.849771	0.847877	0.844086	0.812852	0.737005	1.000000	0.899349	0.842983	0.856708
SCH2000	0.913400	0.882782	0.907836	0.885132	0.885126	0.854219	0.915256	0.885132	0.817174	0.758674	0.899349	1.000000	0.872642	0.850843
LBLA	0.888264	0.871625	0.882234	0.858705	0.857879	0.837173	0.888370	0.858705	0.794176	0.750825	0.842983	0.872642	1.000000	0.918109
NBV	0.883308	0.899221	0.878486	0.913425	0.860519	0.897788	0.882916	0.913425	0.830995	0.732159	0.856708	0.850843	0.918109	1.000000

Figure 5: Matrix of similarities between text sources using GPT's Ada Embedding model (v2) with cosine similarity, showing the relatedness between different Bible translations in English (green box), German (green columns) and Spanish (yellow), along with an abbreviated text (purple), 'opposite' text, and unrelated text (pink).

OpenAI's Ada 002 embedding model performed best overall. Ada correctly assessed the overlapping similarity between the full, English versions of the chapter, calculating a minimum similarity of 0.9030 (between the NLT and WEB translations), a maximum similarity of 0.9983 (between ASV and WEB), and an average similarity of 0.9432.

Table 5: The similarity matrix for the positively-correlated, English translations of the Bible using GPT's embedding model, Ada 002.

Similarity	ASV	BBE	KJV	NIV	NKJV	NLT	WEB
ASV	0.9999999	0.9434712	0.9944914	0.9445881	0.9580632	0.9056944	0.9982998
BBE	0.9434712	0.9999999	0.9391094	0.9445570	0.9175876	0.9146621	0.9431464
KJV	0.9944914	0.9391094	0.9999999	0.9455138	0.9580162	0.9048721	0.9926683
NIV	0.9445881	0.9445570	0.9455138	1.0000001	0.9236755	0.9462423	0.9425475
NKJV	0.9580632	0.9175876	0.9580162	0.9236755	1.0000000	0.9307772	0.9565186
NLT	0.9056944	0.9146621	0.9048721	0.9462423	0.9307772	0.9999999	0.9030391
WEB	0.9982998	0.9431464	0.9926683	0.9425475	0.9565186	0.9030391	1.0000001

[10] The embedding models selected for this research are not trained for cross-language applications, including Ada 002, though OpenAI reports that, because of its large training corpus, which includes text in many various languages, Ada 002 can nonetheless perform admirably on cross-language embedding tasks. As the focus of this research was not specifically assessing similarity between texts in different languages, the English-only versions of these models were used, though tested intra-language for comparison purposes.

NIV – 1st 10 is used to ascertain the impact of different lengths on similarity measurements. The first ten verses of the chapter from the NIV translation ("**NIV - 1st 10**") were compared to the full chapter. The similarity measurements were equivalent to the similarities between the NIV translation, from which the abbreviated text was taken, and the other English translations, ranging from a minimum of 0.9237 to a non-NIV maximum of 0.9462. This demonstrates that the relative lengths of the text have little impact on the similarity results; some models show reduced accuracy when the lengths of the texts were significantly different. Context-injection requires the comparison of context units (~600 tokens) to questions (~20 tokens). The ideal approach will not be impacted by differences in lengths.

Table 6: Similarity between the first 10 verses of Mark 1 compared to the full chapter English translations.

Translation	NIV - 1st 10
ASV	0.944588
BBE	0.944557
KJV	0.945514
NIV	1.000000
NKJV	0.923676
NLT	0.946242
WEB	0.942548

The next two tests measure the embedding models' ability to consider different but related text, as well as completely unrelated text. In the former, the first ten verses of Mark 1 were modified so that while the same characters were performing the same activities in the same locations, the details of the story were negated or juxtaposed (labelled as '**NIV Opp**.'). The embedding model correctly assesses the semi-related text, with a similarity range of 0.8405 to 0.9097 and average of 0.8639. Note that because of the way that Ada 002 vectorizes the embeddings, the scale for similarity is ~0.7 to 1.0, meaning that the linearly-normalized average similarity would be approximately 0.5555, correctly demonstrating a loose relationship. The similarities between the Constitution text and the various

Table 7: Similarities for partially-related (NIV Opp.) and unrelated (Constitution) texts.

Translations	NIV Opp.	Constitution
ASV	0.844569	0.779384
BBE	0.846018	0.780585
KJV	0.845682	0.779462
NIV	0.885704	0.770588
NKJV	0.875230	0.777638
NLT	0.909719	0.774336
WEB	0.840505	0.781374

translations are much lower than the similarities between the KJV and other translations thus demonstrating that the linguistic form of the text does not impede the models' ability to find semantic similarity.

When the English translations were compared to the Constitution, which is completely unrelated to the biblical text. Again, the embedding model correctly assessed the unrelatedness, calculating a similarity range of 0.7706 to 0.7814, with an average of 0.7776 (normalized: ~0.2). Ada was also proficient at correlating the non-English translations with the English translations. The similarity range between *Hoffnung für Alle* and the English texts was 0.8324 to 0.8663 with an average of 0.8475. The similarity range for *Schlachter 2000* was 0.8542 to 0.9153 with an average of 0.8920. The range for *La Biblia de Las Americas* was 0.8372 to 0.8884 with an average of 0.8692, demonstrating correlation between the texts, even in different languages.

To confirm our results, we use the SemEval 2016 dataset to evaluate the embedding models' performance on semantic textual similarity (STS) in English. (Agirre et al., 2016) The SemEval 2017 dataset was used to assess the embedding models' performance on STS in differing languages. Finally, the Massive Text Embedding Benchmark (MTEB) was used to compare our results with the community's. (Muennighoff et al., 2023)

The SemEval datasets provide two syntactically different pairs of sentences. Humans evaluated the semantic content of each sentence and determined their congruence. Sentences that were perfectly related were scored 5. Sentences that were completely unrelated were scored 0, with the embedding models' cosine similarity value of 1.0 correlating to an MTEB score of 5, and a value of 0.0 correlated to an MTEB score of 0.[11] Each models' performance using the MTEB 2016 (English-only) dataset is displayed below. We can see from the graph that MS Marco offers the lowest semantic similarity score where the MTEB score is 0, however its range is considerably wider than the other models'. Ada returns the smallest range of all the models on those pairs that are perfectly semantically similar. All the models show a positive correlation with the MTEB's ground truth.

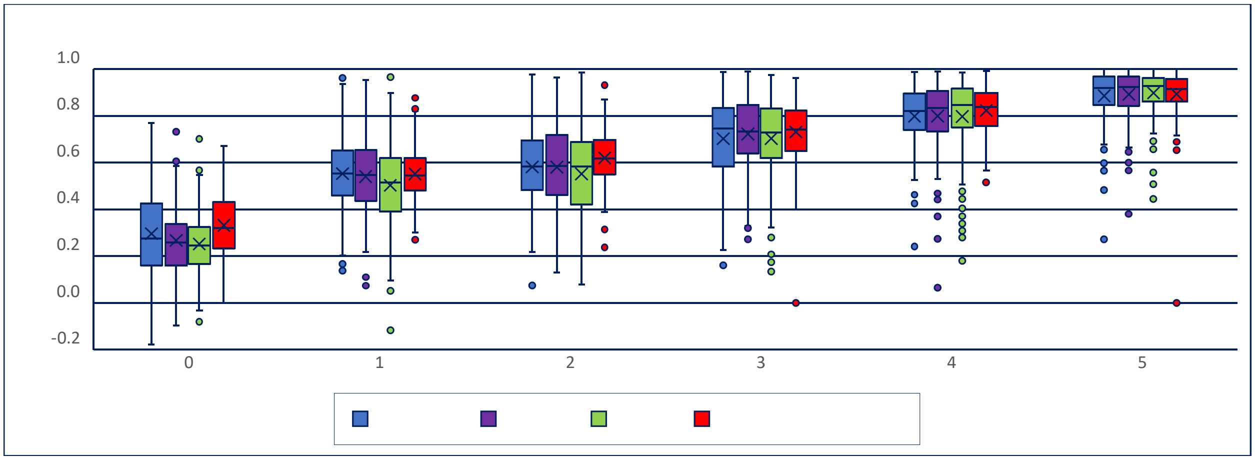


Figure 6: Semantic similarity by model on MTEB 2016 English dataset where the y-axis shows the cosine similarity output by each model, and the x-axis shows the defined MTEB ground truth score (0 – 5) over the sentence pairs.

Because Spearman's r is not as sensitive to outliers as Pearson's r and can detect non-linear relationships, it is used to determine which model offered the best performance.

Table 8: The Spearman's r score (correlation coefficient) for each model measuring its semantic similarity's correlation with the MTEB 2016 ground truth score. OpenAI's Ada 002 model outperformed the other models.

Model Name	Ada 002	MPNet-Base-v2	MiniLM-L6-v2	msmarco-MiniLM
Spearman's r	0.8537215328	0.800299602	0.7898945287	0.7727046708

Spearman's r is also used to calculate relative STS task performance on the MTEB leaderboard. The values calculated herein match the official scores posted on the leaderboard. The MPNet and Mini LM model's correlation results exactly match the leaderboard, though the MS Marco model

[11] Ada outputs cosine similarity in the range of ~0.70 to 1.0. Therefore, linear rescaling was used to project the Ada 002 range onto a scale of 0 to 1, whereby Ada's 0.7 is equivalent to BERT's 0.0, and both models' 1.0 are equivalent. As a result, Ada's score of 0.85 is roughly equivalent to BERT's 0.5.

is not currently on the leaderboard, so its correlation score is not posted. Ada 002's official Hugging Face score is listed as 85.99, whereas we calculate the correlation coefficient with OpenAI's utility at 0.8537.

The same process was conducted with MTEB's 2017 cross-linguistic dataset, calculating the semantic similarity between sentences in English and both German and Spanish.
Ada demonstrated a strong ability to correctly calculate semantic similarity across languages; Ada finds the English/Spanish correlation is 0.8283, and MPNet only 0.3647.

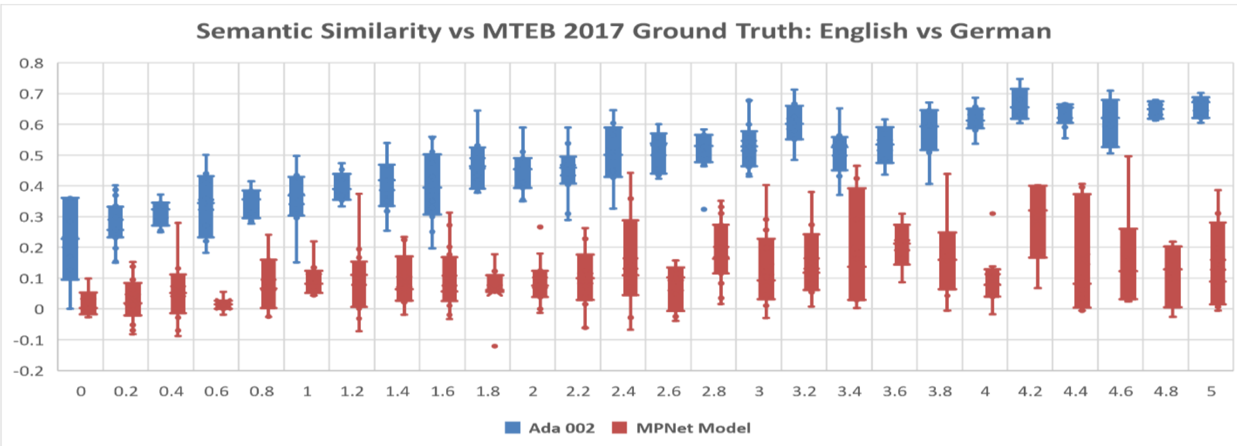


Figure 7: Semantic similarity results for Ada 002 and the MPNet embedding models, compared to the MTEB 2017 STS task dataset of English and German sentence pairs. The Spearman r's correlation shows that Ada offered a strong correlation and MPNet showed a weak correlation.

Ada's embedding performance was better than the other models' average performance on intra-English similarity, on texts of differing lengths, on partially-related texts and texts in different languages, and Ada's vector offers larger dimensionality. Additionally, Dave (2023) and Ofori (2023) found Ada to offer superior performance to BERT models using different assessments. As a result, and in accordance with the current MTEB leaderboard, which shows Ada 002 as the best-performing model on the STS task with text comprised of more than 514 tokens, Ada 002 is used in this research to measure semantic similarity between the standard answer provided by the author, and the generated answer.

CHAPTER 6

EXPERIMENTAL RESULTS: FINE-TUNING VS CONTEXT-INJECTION

6.1 Fine-tuning Results

In total, eighty documents were used to fine-tune eight models; ten documents per model. The models were used to generate answers to twenty standard questions. The answers generated by each model were then semantically compared to the standard answer via vector embeddings using GPT's Ada (version 002) model and Cosine Similarity. The results of Questions 1 and 2 are discussed here for the purposes of illustration. The graphs for the remaining questions can be found in Appendix B.

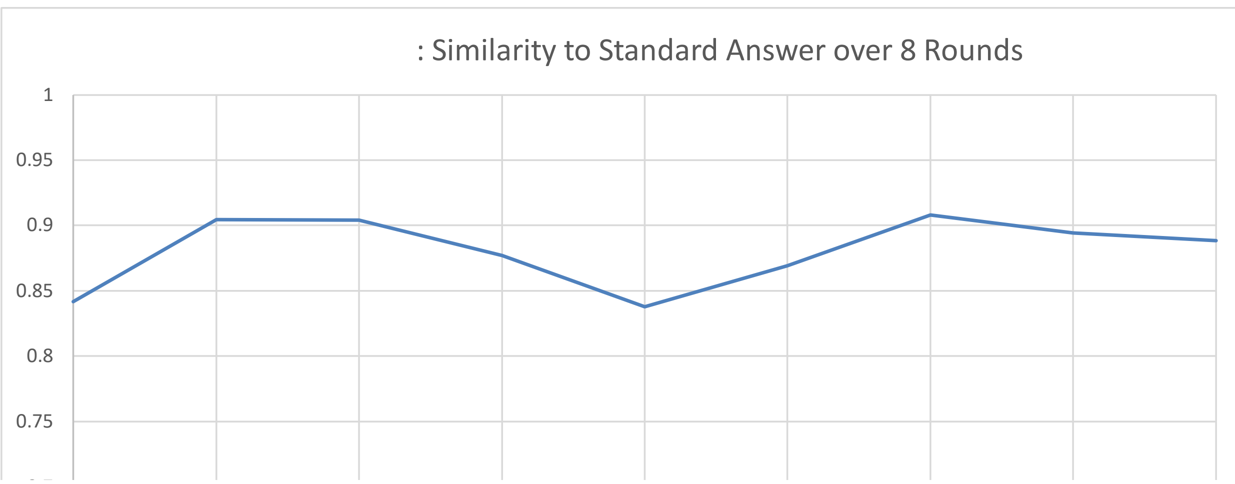


Figure 8: Similarity between the fine-tuned-generated answer to Question 1 and the standard answer, over 8 training rounds. Phase 0 is the base GPT response with no fine-tuning.

The semantic similarity between the fine-tuned-generated answers for Question 1 are shown above. The graph shows the similarity between the answer given after each round of fine-tuning and the standard answer. The first round was trained on 10 documents, the second round with an additional 10 (20 total), the third round with an additional 10 (30 total), etc.

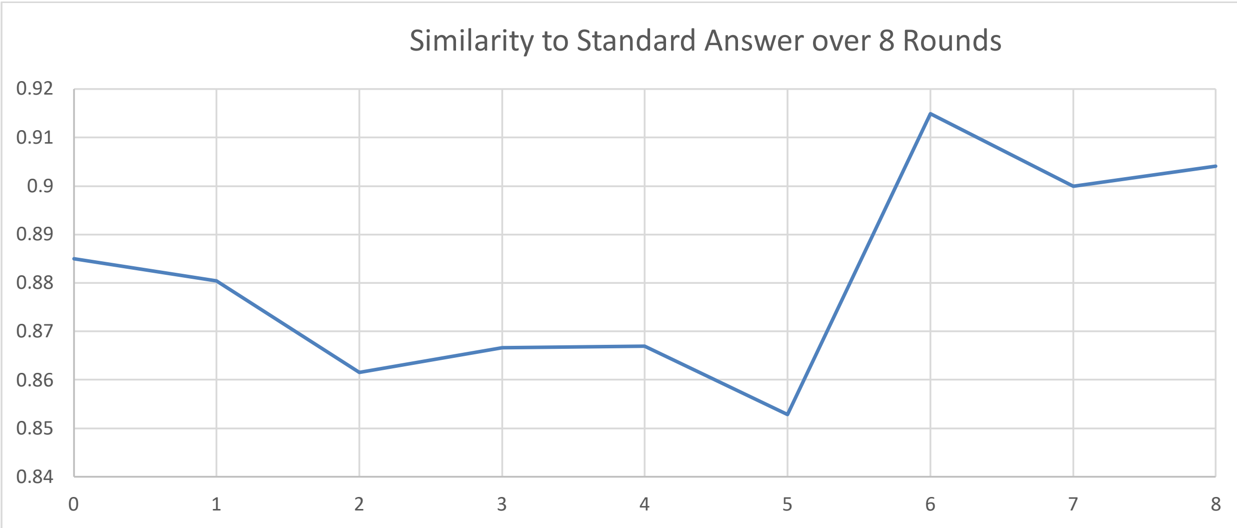


Figure 9: Similarity between the fine-tuned-generated answer to Question 2 and the standard answer, over 8 training rounds. Phase 0 is the base GPT response with no fine-tuning.

6.1.1 Fine-Tuning Conclusion

In almost every case, the quality of the generated answers was improved through fine-tuning in at least one round; the similarity between the generated answer and the standard answer was higher in one phase/round than the base GPT answer. Taking the highest semantic similarity value over all rounds for each question, fine-tuning outperformed base GPT by 8.26%. Taking the average similarity per question, base GPT outperformed fine-tuning on average by 0.26%, and by 0.226% on average compared to the final fine-tuned model. We can conclude that, optimistically, there is value to fine-tuning specific or proprietary data into GPT for answering open-ended, ambiguously-answerable questions in a proprietary domain. The problem, though, is that the similarity is not linearly (or otherwise) related to the volume of data. From each graph we can see that the similarity values bounce up and down with each round. This refutes the hypothesis that more fine-tuning data necessarily leads to better results.

Looking at Question 13 (in Appendix B), it was directly answered in Phase 1. From the graph we see that additional information added in subsequent phases does not contribute much to the value of the answer. Conceptually, this makes sense. The question is answered in Phase 1. We add more data in Phase 2, and more still in Phase 3, etc., but if that data is not related to the question being asked, the additional information that is added to the model in Phases 2 through 8, ostensibly contributes nothing to answering the question. In essence we are adding more information about random topics within the information domain, which are statistically unlikely to directly apply to a question asked at a given point in time. Adding any amount of unrelated data will not logically provide any insights to the model. In fact, it may even serve to 'water down' the model's ability to provide a direct answer to our question, as the model must then search through the sea fine-tuned data to return the most-probable answer, or subject the model to 'catastrophic forgetting.' (Kirkpatrick et al., 2017)

Eight of the questions, Question 13 – 20, were directly answered in the sermons, and those answers fine-tuned in different tuning rounds. Question 13 was 'answered' in Phase 1, Question 14 in Phase 2, etc. Only the answers to Questions 15, 18 and 20 were highest in the phases in which the answers were fine-tuned into the model. This is higher than random chance, 37.5% vs 12.5%, supporting the idea that directly adding information to the fine-tuned model will increase the probability of generating a better answer. But there is no direct relationship, and the results are not reliable enough for implementation in an application.

In summary, fine-tuning does generally increase the quality of the generated answers, however, there is no correlation between the amount of data and the best answer, and only a very loose correlation between the best answer and when the answer is tuned into the model. Additionally, many of the answers exhibited evidence of hallucinations, which is discussed in more detail later in the chapter. As a result, it would be difficult to implement a fine-tuned model for the purpose of question-answering because the quality of the generated answers is dependent on the information needed to directly answer the question, and approximately inversely related to the amount of data fine-tuned into the model. Even if one could fine-tune a large number of models, it would be prohibitively difficult, if not impossible, to know which model would be the optimal one to answer the question.

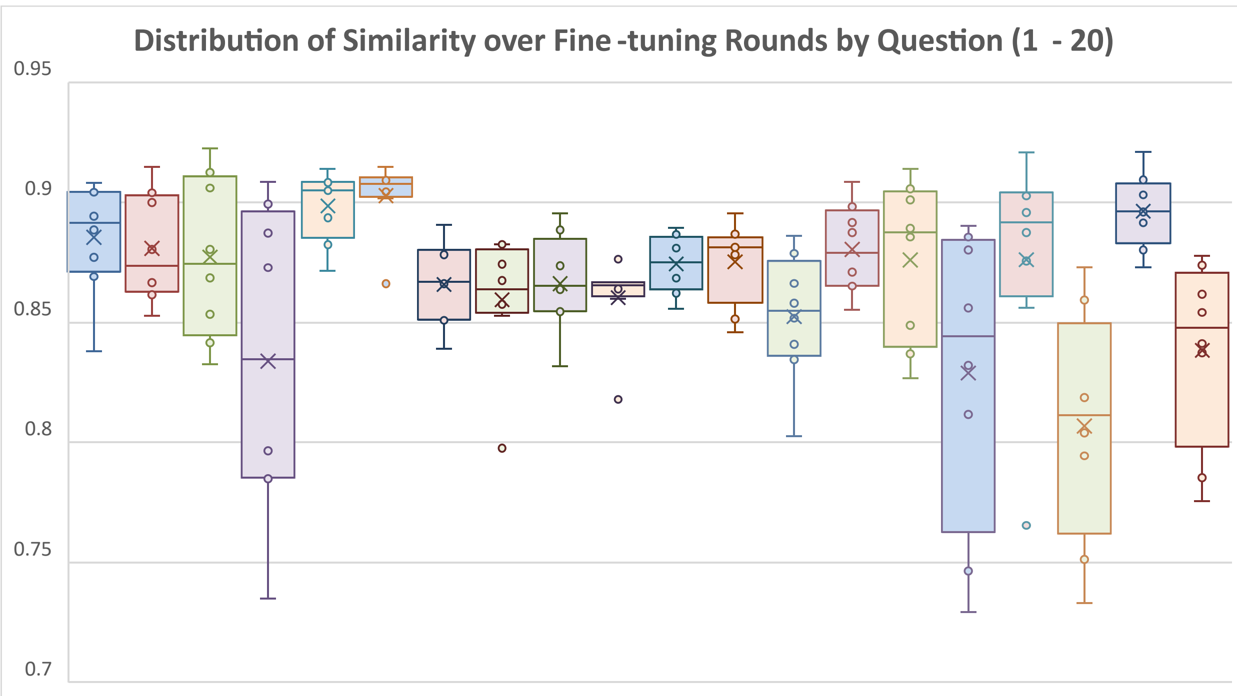


Figure 10: Distribution of semantic similarity over the fine-tuning rounds for each of the questions, Question 1 – Question 20.

6.2 Context-injection Results

In contrast to fine-tuning, context-injection does not ‘add’ any data to the underlying model nor adjust the model’s weights. Instead, context-injection relies on ICL through RAG and MRC to augment answer generation. Through this research we find that context-injection generated answers are semantically closer to the standard answer than the fine-tuned models. GPT-3 with context-injection beat fine-tuned GPT-3's best answers by 4.1% and on average over all answers by 21.5%. The answers generated by GPT-3 through context-injection were 21.1% semantically closer than those of base GPT-3. ChatGPT with context-injection on average generated answers 2.2% semantically closer to the standard answer than the base ChatGPT, and its best answer (to Question 5) was 12.1% closer. GPT-4 with context-injection generated answers on average 5.6% closer than base GPT-4, with its greatest margin being Question 9, at 29.3%. Neither ChatGPT nor GPT-4 allow fine-tuning. GPT-4 with context-injection performed best overall, with an average semantic similarity of .75201 and the best answer overall models to 12 of the 20 questions.

Context-injection can be used with fine-tuned models, and was therefore tested using the following two methods:

1) Pure context-injection using the based model.
    - The token limit for the base Davinci model is 4096 tokens, thus 4 contexts were submitted.
2) Fine-tuned context-injection using the fine-tuned model.
    - The token limit for the fine-tuned Davinci model is 2048 tokens, thus 2 contexts were submitted.

See Chapter 2 for information on how the contexts were created and measured.

The graph below shows the performance comparison between the two methods described above, along with the base GPT model output with no context-injection as a baseline. We see that Base

GPT with context-injection using the top 4 contexts outperforms Base GPT (no context) and the final, Phase 8 fine-tuned model (which was not necessarily the best-performing fine-tuned model) using the top 2 contexts. Again, Base GPT allows for a maximum of 4096 tokens in the prompt and completion, while the fine-tuned Davinci model only allows for 2048 tokens in the prompt and completion. As a result, only two sets of contexts can technically be submitted in the prompt. The fine-tuned model barely outperforms the Base GPT model on two of the questions, and roughly matches its performance on one more. Base GPT without context-injection does outperform both models with context-injection on one question (Question 18: "What are some advantages and disadvantages of fear?"), but drastically underperforms the two models with context-injection on the other questions.

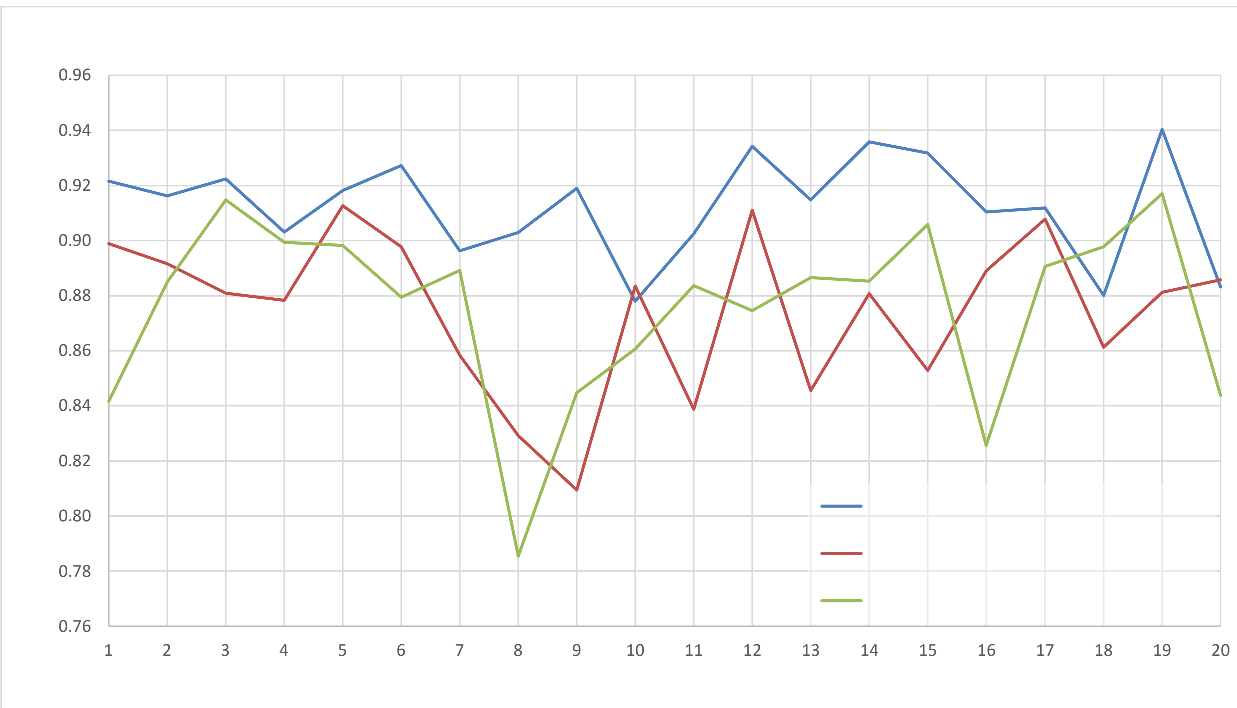


Figure 11: The graph above shows the similarity scores using base GPT-3 with the top 4 contexts from the sermon set, compared to the final (8th round) fine-tuned model with the top 2 contexts from the sermon tuning sets (due to token limitations) and the base GPT-3 model with no context. GPT-3 with context-injection outperforms the others.

To compare the impact of the full context (4 sets) versus the abbreviated context (2 sets), the same experiment was conducted, this time providing the base GPT model with only two contexts, which is equivalent to the maximum number of tokens for use with the fine-tuned model. From the graph below, we see that the base GPT with context-injection still outperforms the fine-tuned model with the same injected context.

These results mirror the conclusions of Awadalla et al. (2022) In their study, they examined the reliability of GPT-3 specifically over four areas: generalizability, bias and fairness, uncertainty calibration and factuality via knowledge updating. Their experimental results specifically in the open QA domain found similar results. Specifically, that (1) increasing the number of in-prompt examples improves accuracy, (2) adding grounded context can improve GPT-3 performance on factual QA, and (3) GPT-3 can update its knowledge when provided passages conflicting with its initial pre-training.

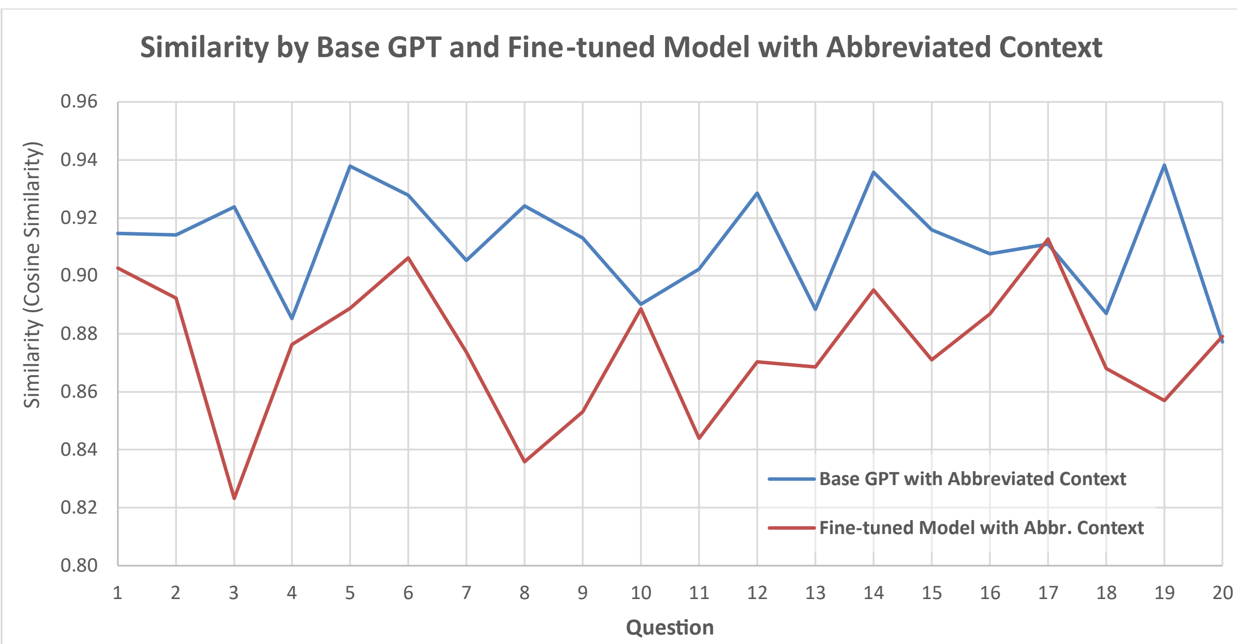


Figure 12: The graph above shows the similarity scores using base GPT-3 with the top 2 contexts, which is context-injection though only with 2 contexts, and not 4, from the sermon tuning set, compared to the final (8th round) fine-tuned model with the top 2 contexts from the sermon tuning sets. GPT with context-injection outperforms fine-tuning with context-injection for all but two questions, where the similarities are approximately the same.

The results clearly show that context-injection on base GPT outperforms context-injection on the fine-tuned (Phase 8) model. One potential reason is that fine-tuning is thought to lead to catastrophic forgetting (Kirkpatrick et al., 2017), which is the phenomenon in which an LLM 'unlearns' knowledge or loses abilities upon further training. A review of the answers generated with fine-tuning showed a marked decrease in the model's ability to follow directions, e.g., limiting responses to a set number of sentences (as opposed to tokens). As context-injection relies on learned abilities such as ICL and MRC, it is possible that the value brought through fine-tuning could not offset the loss of these abilities.

Context-injection using ChatGPT and GPT-4 was also compared to the base GPT models. In all cases, context-injection offered better performance than the base GPT model alone.

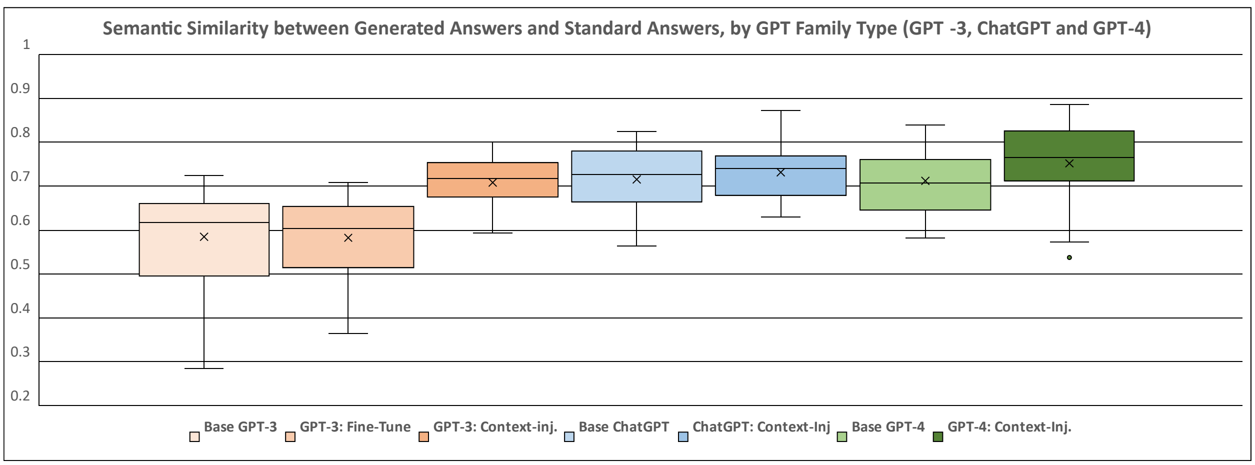


Figure 13: GPT-3 with fine-tuning and context-injection are compared to base GPT-3. ChatGPT and GPT-4 with context-injection are compared to base ChatGPT and GPT-4.

To evaluate whether the differences between the compared models were statistically significant, a Wilcoxon signed-rank test was used. Using an $\alpha = 0.05$, we conclude that the answers generated with Fine-Tuned GPT-3 were not significantly different than those of Base GPT-3, while answers generated by GPT-3 with context-injection were significantly different than both those generated by Base GPT-3 and GPT-3 with fine-tuning. Similarly, the difference in answers generated by Base GPT-4 and GPT-4 with context-injection were also statistically significant.

Table 9: Comparison of model output using the Wilcoxon signed-rank test.

Comparison	Statistic	p-value	Reject H_0
Base GPT-3 vs FT GPT-3	103.0	0.9563	NO
Fine-Tuned GPT-3 vs CI GPT-3	5.0	1.9073e-05	YES
Base GPT-3 vs CI GPT-3	5.0	1.9073e-05	YES
Base GPT-4 vs CI GPT-4	50.0	0.0400	YES
Base ChatGPT vs CI ChatGPT	61.0	0.1054	NO

We also compared the answers generated by Base ChatGPT (3.5 Turbo) and ChatGPT with context-injection. In that case, the difference in similarity scores were not statistically significant (p=0.1054). Nonetheless, the results appear to support the conclusion that context-injection generally offers better performance over fine-tuning and base GPT.

6.2.1 Transparency

An important advantage that context-injection offers over fine-tuning is transparency. A fine-tuned model is a black box where prompts are entered, and responses are returned. If faced with an aberrant response, the only helpful troubleshooting information are the prompt and response. There is no visibility into how the model used the training corpus to generate the answer.

In contrast, context-injection by its nature offers a greater degree of transparency into its response process. True, the model itself is still a black box, but the process inputs, including most importantly the contextual information on which the model's answer is based, can be fully logged. If faced with an aberrant response, the contextual information and its semantic similarities (and even vectors themselves) could be logged and reviewed for errors or potential improvements. This may be required for some commercial, regulatorily-controlled applications, and is not possible with a fine-tuned model.

6.3 Computational cost comparison

There are three main sources of costs when using OpenAI's GPT models: fine-tuning costs, which includes one cost for training and another for usage, base model usage costs, which is based on the number of tokens submitted in the prompt and received in the completion (per token), and the embedding cost (per token).

The following table shows the costs per token for the different model applications at the time of publishing this research[12].

Table 10: The current relevant costs for conducting our experiment set.

Base Davinci Model	Fine-tuning Training	Fine-tuning Usage	Embedding Model
$0.0200 / 1K tokens	$0.0300 / 1K tokens	$0.1200 / 1K tokens	$0.0001 / 1K tokens

Table 10: The current relevant costs for our experiment set.

Using context-injection on the fine-tuned models is logically, and by far, the most expensive application, as it requires training a model, embedding all the contexts and then using the fine-tuned model (with a higher per-prompt cost). If we assume that the training/embedding corpus is 10,000 tokens, the fine-tuning training costs $0.30 (one-time) and the embedding costs $0.001 (one-time). Submitting the maximum number of tokens in one request would add $0.246 to the $0.301 up-front costs, resulting in a total of $0.507 for the first request and $0.246 for each additional request (though no additional fine-tuning or embedding costs).

Using the base GPT model with full context-injection is the next-most expensive option, on a per-prompt basis. Assuming we initially generate embeddings for 10,000 tokens for an up-front

[12] https://openai.com/pricing

cost of $0.001 and then submit the full 4096 tokens in each request, the total cost per request is $0.0819 (plus the $0.001 embedding cost). Note, though, that using the abbreviated context of approximately 2048 tokens delivers almost equivalent results at almost half the cost (because we are submitting half the prompt tokens, but still receiving the same number of completion tokens). As such the cost is approximately $0.041 (plus embedding).

Using the fine-tuning without context-injection is the next most-expensive, per-request option. Again, assuming a training corpus of 10,000 tokens, the training process costs $0.30. Then, assuming the average request of 300 tokens with no context, which is comprised of approximately 50 tokens in the prompt and 250 tokens in the response, for a total cost of $0.036 per request, the total cost is $0.336 for the first request and $0.036 for each additional request.

Finally, and perhaps obviously, using the base GPT model with no fine-tuning and no context-injection is the cheapest option. With a cost of $0.02 per 1000 tokens, assuming the average request of 300 tokens, the per-request cost is $0.006, with no up-front costs.

From this cost analysis it should be clear that using fine-tuning with context-injection is not the optimal solution, as it costs the most, but delivers worse performance than context-injection alone. As such, fine-tuning with context-injection should not be used unless there is a compelling reason.

6.3.1 Summary of cost considerations

The computational costs, as measured by proxy using the financial costs, of the various models are not in-line with performance.

Order of performance by method:

1. Base GPT model with context-injection.
2. Fine-tuned model with context-injection.
3. Fine-tuned model without context-injection.
4. Base GPT with no fine-tuning nor context-injection.

Order of cost (highest to lowest) by method:

1. Fine-tuned model with context-injection.
2. Fine-tuned model without context-injection.
3. Base GPT model with context-injection.
4. Base GPT with no fine-tuning nor context-injection.

Answers that are generated from fine-tuned models reflect the style of the text on which the model was trained, but come with the probable inclusion of hallucinated information. In contrast, answers that are generated using context-injection reflect the meaning, but not the style, of contextual information on which the answer is based. Answers that are generated by fine-tuned models and incorporate context-injection to answer questions reflect the stylistic fingerprint of both the base GPT model and fine-tuned model.

CHAPTER 7

RESULTS, SUMMARY & CONCLUSION

The focus of this research is comparing four potential information states of GPT-3:

1. GPT-3 pre-trained with no fine-tuning and no contextual prompt.
2. GPT-3 fine-tuned on provided sermon data.
3. GPT-3 pretrained provided with semantically-related context.
4. GPT-3 fine-tuned and provided with semantically-related context.

Answers to the twenty original questions were generated using these methods, and their semantic similarity compared to the standard answers. GPT with context-injection offered the best performance, but comes at the second highest cost; approximately $0.08 for a full request at 4096 tokens and $0.04 for a request of 2048 tokens, plus the initial embedding costs. The method using both fine-tuning and context-injection provided the second-highest performance, but it was by far the most expensive method. Fine-tuning alone offered the third-highest performance, but the cost to train on the corpus is fairly high; this research cost approximately $120 to fine-tune the models, and the per-token cost of using the fine-tuning models is the highest per-token cost among all the methods. Using the base GPT model for question-answering is the cheapest option, as it does not require any upfront fine-tuning, nor additional prompting or embedding costs. However, it offers the worst performance among the methods tested, and is only possible to the extent that the desired answer was already pre-trained into the base GPT model, which for specific, custom, or proprietary information is almost certainly not the case. Unless circumstances deviate significantly from this experimental paradigm, Using the base GPT model with context-injection is the optimal strategy, which balances cost, performance and ease (as fine-tuning is perhaps the most-arduous step in the process).

7.1 Further Research and Discussion

This research found that context-injection outperformed fine-tuning on open QA. Liu et al. (2022) proposed a new prompting method called T-Few, which is a parameter efficient few-shot learning 'recipe.' They demonstrate that T-Few outperforms ICL on classification tasks, though it was never applied to generative tasks such as QA.

7.1.1 Text Segmentation

In this research, the sermons were divided into knowledge units of 600 tokens, which equates to roughly 450 words, or 3 paragraphs and is an approximation of the average number of paragraphs that are required to capture a complete idea. Larger, smaller and/or overlapping knowledge units could be used. Further research could be conducted on the impact of smaller knowledge units versus larger, or on using larger knowledge units to get general information, and then smaller knowledge units to get focused information.

The performance of context-injection would almost certainly be enhanced with better text segmentation. An inherent issue with transcribed data is that it offers no natural breaks from which we can deduce logical breaks, which is a prerequisite for semantic segmentation. Further research could be conducted on the impact that better segmentation of knowledge units has on the performance of context-injected question-answering.

7.1.2 Performance evaluation

Several models and methods were evaluated for their ability to generate vector embeddings, which were then used to evaluate the performance of the four tested techniques. All these models were publicly available, off-the-self models that were not tailored to the information domain. Further research could be conducted that considers newer or different models, or builds an embedding model specific to this information domain, which may lead to improved vector embeddings and therefore improved retrieval and better measures of similarity.

7.1.3 Summarizations

A cursory review of the training corpus reveals that much of the language is allegorical or oratory filler, neither of which contributes to specifically answering questions. Further research could be conducted on the impact that using summaries, instead of the complete text, would have on the quality of the generated answers.

7.1.4 Stylistic fingerprinting

Only a few of the many stylometric measures were employed to determine how the generated answers' styles compared to the training corpus. Further research could be conducted to determine the optimal measures that could be used to generate a stylometric fingerprint, and compare that fingerprint relative to other corpuses. For example, ExpertAI's Writeprint API[13] offers dozens of metrics that together can uniquely describe a body of text. The API output could be organized as a fixed-length vector, and two different vectors could be measured for similarity akin the process described in this work for measuring semantic similarity.

---

[13] https://try.expert.ai/information-detection/writeprint

7.2 Conclusion

This research demonstrated that GPT is capable of answering discrete, deterministic questions with a high rate of accuracy, and especially so when the context for answering such questions is provided to the model (e.g., SQuAD). Moreover, this research demonstrated that GPT is capable of generating answers to open-ended, ambiguous questions. These abilities were specifically evaluated in the context of using oral data in the information domain of Christian orthodoxy. In order to do so, a pipeline was created, which consumed weekly sermons, transcribed them, segmented the sermons into knowledge units, and then:

1) fine-tuned GPT on the sermon data using the units, and
2) generated a vector database of the embeddings and their knowledge unit.

The research demonstrated that it is possible to positively influence the answers provided by GPT through fine-tuning, and that that the size of the fine-tuning corpus does not correlate to the quality of the answers provided. In fact, the research demonstrated that a larger fine-tuning corpus could lead to lower-quality results.

This research also found that the answers generated through context-injection were better than both fine-tuned and base GPT models; context-injection *without* fine-tuning offers better answers than context-injection answers *with* fine-tuned models, and at lower costs. This is also true for newer GPT models such as ChatGPT and GPT-4 models, as context-injection led to better performance over their base model counterpart. Through context-injection, the information used to generate the answer can be controlled, logged and extemporaneously reviewed, which is a regulatory requirement for applications in a regulatorily-controlled environment, e.g., applications that operate under the jurisdiction of 21 CFR Part 11. This is *not* possible with fine-tuned or base LLM models.

This research evaluated both methods for the manifestation of hallucination, which is when an LLM fabricates or provides inaccurate information. The phenomenon of hallucination was only observed with fine-tuned models. While LLM hallucination can be mitigated in context-injection via prompt engineering, it cannot be mitigated with fine-tuned models because of the fundamental nature of their operations.

Finally, the research showed that while GPT alone is capable of generating logical answers to these types of questions, they are lower-quality answers compared to the other methods, and naturally base GPT cannot consider additional, proprietary information to which it has no training or access. From this research we can conclude that the context-injection method for influencing GPT is cheaper and provides better results, both in terms of fidelity to the contextual corpus (i.e., no hallucinating) and semantic similarity to the standard answer. Context-injection, therefore, is the first method that architects and practitioners should consider when seeking to employ question-answering on a proprietary corpus. In contrast, fine-tuning is best when the goal is to output text whose style is significantly similar to the textual style of the training corpus.

REFERENCES

[1] Abraham, A. (2018). *The neuroscience of creativity*. Cambridge University Press.

[2] Agirre, E., Banea, C., Cer, D., Diab, M., Gonzalez Agirre, A., Mihalcea, R., ... & Wiebe, J. (2016). Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In SemEval-2016. 10th International Workshop on Semantic Evaluation; 2016 Jun 16-17; San Diego, CA. Stroudsburg (PA): ACL; 2016. p. 497-511.. ACL (Association for Computational Linguistics).

[3] Awadalla, A., Wortsman, M., Ilharco, G., Min, S., Magnusson, I., Hajishirzi, H., and Schmidt, L. (2022) Exploring The Landscape of Distributional Robustness for Question Answering Models. In Findings of the Association for Computational Linguistics: EMNLP 2022, pages 5971–5987, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[4] Baradaran, R., Ghiasi, R., & Amirkhani, H. (2022). A survey on machine reading comprehension systems. Natural Language Engineering, 28(6), 683-732.

[5] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. Journal of machine Learning research, 3(Jan), 993-1022.

[6] Bommarito II, M., & Katz, D. M. (2022). GPT Takes the Bar Exam. *arXiv preprint arXiv:2212.14402*.

[7] Brainly. (n.d.). “Shakespeare’s complete works have approximately 3.5 million characters.” Retrieved from https://brainly.com/question/6037509

[8] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, *33*, 1877-1901.

[9] Cai, D., Wang, Y., Bi, W., Tu, Z., Liu, X.. Lam, W., and Shi, S. (2019a). Skeleton-to-response: Dialogue generation guided by retrieval memory. In 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), pages 1219–1228.

[10] Cera, D., Diabb, M., Agirrec, E., Lopez-Gazpioc, I., Speciad, L., & Donostia, B. C. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Cross-lingual Focused Evaluation.

[11] Kenton, J. D. M. W. C., & Toutanova, L. K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of naacL-HLT (Vol. 1, p. 2).

[12] Covington, M. A. (2009, March). Idea density—A potentially informative characteristic of retrieved documents. In IEEE Southeastcon 2009 (pp. 201-203). IEEE.

[13] Dave, P. (2023, June 23). GPT vs BERT: Which is Better? Comparing two large-language models: Approach and example. Retrieved from https://towardsdatascience.com/gpt-vs-bert-which-is-better-2f1cf92af21a

[14] Dunn, A., Dagdelen, J., Walker, N., Lee, S., Rosen, A. S., Ceder, G., ... & Jain, A. (2022). Structured information extraction from complex scientific text with fine-tuned large language models. *arXiv preprint arXiv:2212.05238*.

[15] Feng, J. E., Anoushiravani, A. A., Tesoriero, P. J., Ani, L., Meftah, M., Schwarzkopf, R., & Leucht, P. (2020). Transcription error rates in retrospective chart reviews. *Orthopedics*, *43*(5), e404-e408.

[16] Hämäläinen, P., Tavast, M., & Kunnari, A. (2023, April). Evaluating large language models in generating synthetic hci research data: a case study. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (pp. 1-19).

[17] Han, M, Zhang, X, Yuan, X, Jiang, J, Yun, W, Gao, C. (2021). A survey on the techniques, applications, and performance of short text semantic similarity. *Concurrency Computat Pract Exper*. 2021; 33:e5971. https://doi.org/10.1002/cpe.5971

[18] Hare, T. (2014). Translation and the Sacred: Translating Scripture. A Companion to Translation Studies, 529-542.

[19] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., ... & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, 114(13), 3521-3526.

[20] Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bull Soc Vaudoise Sci Nat, 37, 547-579.

[21] Jatnika, D., Bijaksana, M. A., & Suryani, A. A. (2019). Word2vec model analysis for semantic similarities in english words. Procedia Computer Science, 157, 160-167.

[22] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., ... & Fung, P. (2023). Survey of hallucination in natural language generation. ACM Computing Surveys, 55(12), 1-38.

[23] Kontostathis, A., & Pottenger, W. M. (2006). A framework for understanding Latent Semantic Indexing (LSI) performance. Information Processing & Management, 42(1), 56-73.

[24] Le, Q., & Mikolov, T. (2014, June). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188-1196). PMLR.

[25] Lee, J. S., & Hsiang, J. (2020). Patent claim generation by fine-tuning OpenAI GPT-2. World Patent Information, 62, 101983.

[26] Li, H., Su, Y., Cai, D., Wang, Y., & Liu, L. (2022). A survey on retrieval-augmented text generation. arXiv preprint arXiv:2202.01110.

[27] Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., & Raffel, C. A. (2022). Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. Advances in Neural Information Processing Systems, 35, 1950-1965.

[28] Mahowald, K., Ivanova, A. A., Blank, I. A., Kanwisher, N., Tenenbaum, J. B., & Fedorenko, E. (2023). Dissociating language and thought in large language models: a cognitive perspective. arXiv preprint arXiv:2301.06627.

[29] Martino, A., Iannelli, M., & Truong, C. (2023, May). Knowledge injection to counter large language model (LLM) hallucination. In European Semantic Web Conference (pp. 182-185). Cham: Springer Nature Switzerland.

[30] Mendsaikhan, O., Hasegawa, H., Yamaguchi, Y., Shimada, H., & Bataa, E. (2020). Identification of cybersecurity specific content using different language models. Journal of Information Processing, 28, 623-632.

[31] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[32] Mosbach, M., Pimentel, T., Ravfogel, S., Klakow, D., and Elazar, Y. (2023). Few-shot Fine-tuning vs. In-context Learning: A Fair Comparison and Evaluation. In Findings of the Association for Computational Linguistics: ACL 2023, pages 12284–12314, Toronto, Canada. Association for Computational Linguistics.

[33] Muennighoff, N., Tazi, N., Magne, L., & Reimers, N. (2022). MTEB: Massive text embedding benchmark. arXiv preprint arXiv:2210.07316.

[34] Noah, W. (2005). Ink & Blood: Dead Sea Scrolls to the English Bible. AcO Publishing.

[35] Ofori, D. (2023, February 10). GPT-3 vs Other Text Embeddings Techniques for Text Classification: A Performance Evaluation. Retrieved from https://derrickofori015.medium.com/gpt-3-vs-other-text-embeddings-techniques-for-text-classification-a-performance-evaluation-b3a3e6e84cb7

[36] Oniani, D. and Wang, Y. (2020). A Qualitative Evaluation of Language Models on Automatic Question-Answering for COVID-19. In Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '20). Association for Computing Machinery, New York, NY, USA, Article 33, 1–9. https://doi.org/10.1145/3388440.3412413

[37] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.

[38] Raunak, V., Menezes, A., and Junczys-Dowmunt, M. (2021). The curious case of hallucinations in neural machine translation. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologys (NAACL-HLT'21). 1172–1183.

[39] Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *https://aclanthology.org/D16-1264/*.

[40] Rodriguez, S. (2023, August 22). Optimizing Transformation. Axendia. https://axendia.com/blog/2023/08/22/optimizing-transformation/

[41] Rozado D. (2023) The Political Biases of ChatGPT. Social Sciences. 2023; 12(3):148. https://doi.org/10.3390/socsci12030148

[42] Rozado, D. (February 16, 2023.). RightWingGPT. Retrieved from https://davidrozado.substack.com/p/rightwinggpt

[43] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information processing & management, 24(5), 513-523.

[44] Sawicki, P., Grzes, M., Jordanous, A., Brown, D., Peeperkorn, M. (2022) *Training GPT-2 to represent two Romantic-era authors: challenges, evaluations and pitfalls.* In: 13th International Conference on Computational Creativity. pp. 34-43. Association for Computational Creativity (ACC) ISBN 978-989-54-1604-2. (KAR id:94992).

[45] Sawicki, P., Grzes, M., Goes, F., Brown, D., & Peeperkorn, M. Khatun, Aisha (2023). Bits of Grass: Does GPT already know how to write like Whitman? Proceedings of the 14th International Conference for Computational Creativity.(In press).

[46] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., ... & Wen, J. R. (2023). A survey of large language models. arXiv preprint arXiv:2303.18223.

APPENDICES

Appendix A: Stylometric assessments of the generated outputs

An analysis was also conducted on the text to determine the impact that fine-tuning and context-injection have on the style of the output text. Generated answers from fine-tuned models are stylistically closer to the fine-tuning text than answers generated through context-injection.

Propositional idea density:

Idea density is a measurement of the ratio of propositional ideas in a text to the number of words in the same text. The pioneer of Propositional Idea Density, Dr. Michael Covington, described it as one measurement in a suite of measurements for stylometric analysis, and serves to describe the level at which the text is written.

Table 11: Comparison of the four generated answer types and their relative propositional idea densities.

All Context-Injection Answ's	**Average of All Sermons**	**All Fine-tuned Answers**	**Fine-tune with C.I.**
1239 propositions 2378 words **0.521** idea density 0.501 95% conf min 0.541 95% conf max	Props: 4108.625 Words: 7563.7125 Density: **0.542675** Conf min: 0.5311375 Conf max: 0.5540875	1922 propositions 3347 words **0.574** idea density 0.557 95% conf min 0.591 95% conf max	2048 propositions 3551 words **0.577** idea density 0.560 95% conf min 0.593 95% conf max

Analyzing the two sets of fine-tuned answers, with and without context-injection, along with the set of answers generated from context-injection alone, there is no clear stylistic gravitation towards or away from the source training corpus, as the answers generated with context-injection have the lowest idea density at 0.521, and both answer sets generated with fine-tuning have almost identical idea densities, 0.574 and 0.577 respectively, while the average idea density of the training corpus is right in the middle at 0.542675.

The chart below comes from Covington (2009) and shows the distribution of the example texts that were analyzed as part of their research. All four sets of documents in this research fall within the 'Technical' document category, though the answers generated via context-injection fall along the upper bound for 'Scholarly' document.

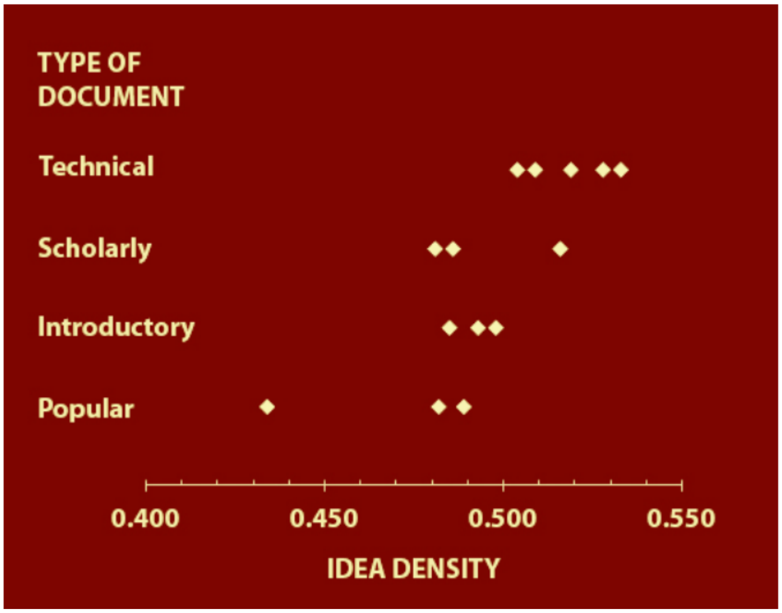


Figure 14: Comparison of four document types, taken from Covington (2009) showing the distribution of tested document types and their propositional idea densities.

Signature Stylometiric System

The *Signature* Stylometiric System[14] was used to assess the following pieces of text (Note: in 2013, Signature was used to reveal J.K. Rowling's pseudonym, Robert Gailbraith[15]):

1) All the sermon text (80 sermons total)
2) All the generated answers from the final fine-tuned model
3) All the generated answers from the base GPT model with context-injection
4) All the generated answers from the from the final fine-tuned model with context-injection
5) All the text in the standard answers provided by the author.

Signature uses 5 main categories to analyze the stylometric characteristics of text: Word length, Sentence length, Paragraph length, Letters, and Punctuation. Only the distribution of sentence

[14] https://www.philocomp.net/texts/signature.htm
[15] https://www.digitaltrends.com/computing/computer-software-reveals-jk-rowling-as-author-of-novel-written-under-pen-name/

lengths varied among the metrics of the documents, correlating the fine-tuned-generated answers with the training corpus, and not with the context-injection-generated answers. All supporting graphs are shown in Appendix B.

Word length:

The distribution of the lengths of the words used among all five documents were roughtly the same, and not statistically significant.

Paragraph length:

The distribution of the number of paragraphs used among the sermons was the same, zero, for the fine-tuned answers and fine-tuned answers with context-injection. Likewise, the training corpus was a list of sentences, and did not contain any paragraphs. Only the standard answers and the answers generated using base GPT with context-injection contained any paragraphs. There was no correlation in the distribution of paragraphs between the standard answers and the answers generated using contex-injection.

Letters used:

The distribution of the letters used among the five document types were almost exactly the same, and showed no stylometric differences.

Punctuation distribution:

The ratios of used punctuation among the five documents are roughly equivalent. The notable differences are between the base GPT model with context-injection and the standard answers, which both include biblical references in the format **(***book chapter***:***verse***)**. The sermon text and fine-tuned answers spelled out the references as book, chapter and verse, demonstrating that the answers generated from the fine-tuned models drew inspiration directly from the fine-tuning corpus, thereby mimicking its ‘writing’ style.

Sentence lengths:

Where the documents varied the most was in the sentence lengths. The sermon text tended to use shorter sentences, with progressively fewer longer sentences. This pattern is almost perfectly matched in the fine-tuned answers and to a smaller degree, in the standard answers. In contrast, the answers generated using context-injection had a classically normal distribution. The answers that were generated with context-injection on fine-tuned models showed balance between both distributions.

Sermon text versus fine-tuned-generated answers:

Comparing the distribution of sentence lengths between the fine-tuning answers and sermon text, there is a clear congruency between the distributions. Both texts contain more short sentences and fewer long sentences. In contrast, Signature found that the difference between the sermon text and the answers generated using context-injection was significant. Therefore, the fine-tuned-generated answers do not reflect the stylistic fingerprints of the underlying training corpus.

Finally, the generated answers that were created using the final, Phase 8 fine-tuned model with context-injection reveal a distribution of sentence lengths that reflects both the sermon text and the base GPT-generated text, with fairly consistent sentence lengths of between 1 and 30 words, and then progressively fewer longer sentences, thereby demonstrating that fine-tuning with context-injection partially mimicks the style of the underlying training corpus, but this style is diluted by the stylistic effects of context-injection.

Direct Comparison of Answers

When comparing the styles of the generated answers side by side, the differences become clear. The fine-tuned answers mirror the training corpus text very closely. The answers are generated in the first-person singular form, and are presented from the perspective of the author (occasionally referring to 'itself' by the author's name), speaking in front of an audience and occasionally in a manner that elicits a response. The answers generated through context-injection are not offered in the first-person and take a more-muted tone.

Another major difference is that the fine-tuned answers are much more likely to contain wrong, hallucinated or conflated information. The fine-tuned model's answer to question 14 referred to "Israel's return from Babylonian captivity back in 1948." There are four references in the training corpus to Israel's Babylonian captivity; two in relation to the time of Ezra and two more to the captivity itself; never in relation to 1948. One sermon refers to a poll that was started in 1948, but no texts reference Israel's Babylonian captivity in 1948.

Again, in Question 14, the answer generated from the fine-tuned model includes the phrase *"...when they come visit me at [the church] where I serve as pastor down here in Atlanta Georgia."* The answer does include the correct name of the church, but that church is not located in Atlanta, Georgia.

In another answer generated from fine-tuning, the model listed a YouTube video link which does not exist (if it ever did). No similar fabrications, or "hallucinations," have been observed in the answers generated through context-injection.

Question (14): Why is it important that Jesus lives?

Fine-tuned Answer:

Because if Jesus didn't live, we wouldn't have a Savior. If Jesus didn't live, there would be no atonement for our sins. If Jesus did not live and die on the cross as a sacrifice for us, then all of humanity is lost forever. But because he lived and he died and he rose again on the third day so that you could receive him into your life to forgive you of your sins and give you eternal life in heaven with Him after this one here on earth is over. Why is it important that Jesus lives? Number two I want to talk to you about today is this biblical prophecy fulfilled by Israel's return from Babylonian captivity back in 1948. Now listen, before I begin my message today let me just tell you something very quickly about myself personally because people always ask me questions whenever I get up behind this pulpit or when they come visit me at [the church] where I serve as pastor down here in Atlanta Georgia.

Fine-tuned and Context-injected:

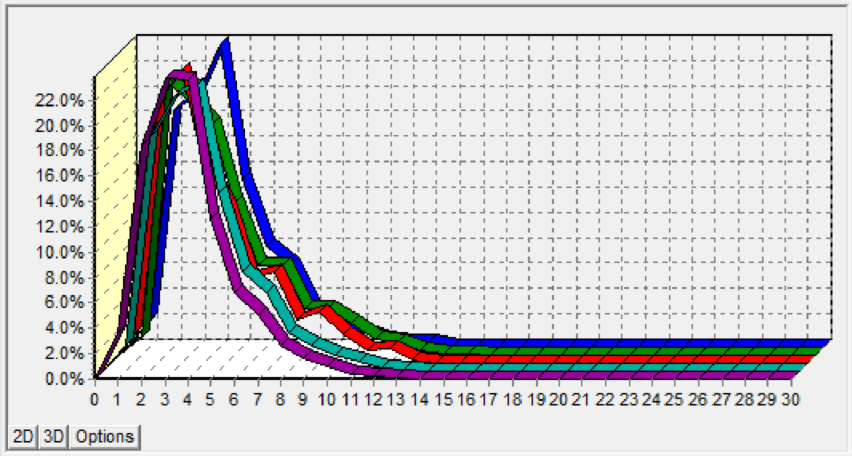
Because when Jesus lives, we get to live. When he goes on up to heaven and ascends into heaven, it's like the book says in Acts one that same night He was taken up into heaven. And then a cloud covered the place where they were staying. Then suddenly there was a voice from above them saying, this is my Son whom I love dearly. Listen, you need to know two things about your life today if you're going to be victorious over fear and worry and depression and hopelessness in your life. You need to know two things about yourself number one is this you are loved by God with an everlasting love that will never die because of his great compassion for us as sinners who needed Him so much more than we ever could have imagined needing Him before he died for us on the cross of Calvary. But now listen here's what I want you to do today while we sing worship songs during our time together here at church service right now while people are singing worship songs.

Context-Injected Answer:

We should pursue God's presence because it is the only way to truly experience Him. When we come into His presence, He speaks to us and reveals Himself in ways that cannot be experienced outside of His presence. Pursuing God's presence also helps us draw closer to Him and build a deeper relationship with Him. It allows us to hear from Him more clearly, receive direction for our lives, and gain strength when facing difficult times. Additionally, pursuing God's presence gives others around us an example of what it looks like to seek after the Lord and encourages them in their own faith journey as well.

Even with simply a cursory view of the two sets of generated answers, the answers generated via fine-tuning adopted the style of the training corpus, while the answers generated via context-injection did not.

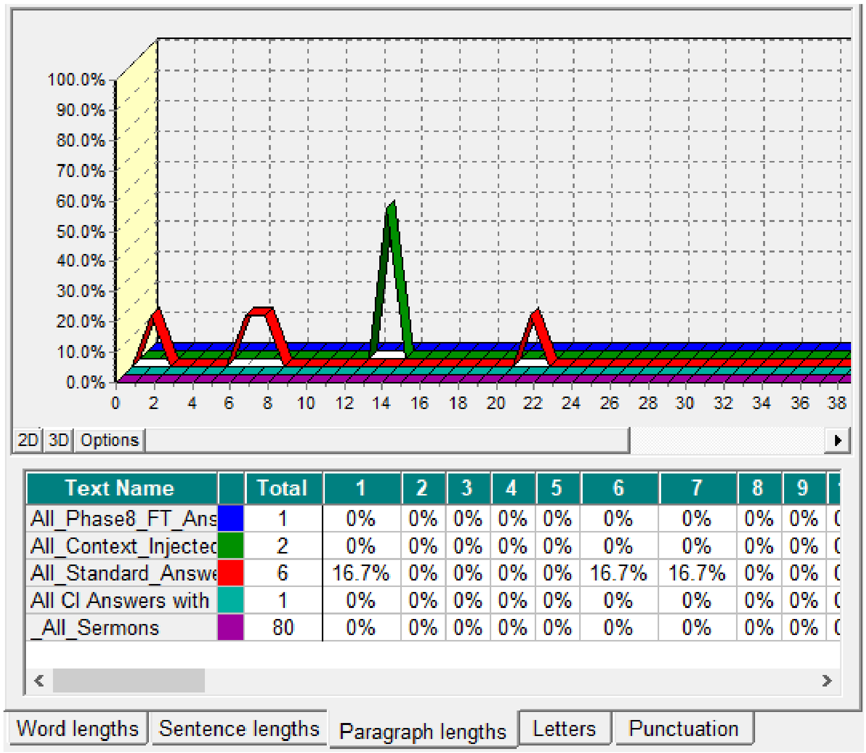
Appendix B: Graphs of the measurements from the Signature Stylometric System



Text Name		Total	1	2	3	4	5	6	7
All_Phase8_FT_Ans		3458	2.14%	18.7%	19.9%	23.9%	13.3%	7.89%	6.4
All_Context_Injecte		2331	1.42%	21.7%	19.9%	18.1%	12%	7.04%	7.1
All_Standard_Answe		3377	2.28%	20.7%	23.2%	15.3%	12.5%	6.72%	7.2
All CI Answers with		3554	2.05%	18.4%	21.6%	22.4%	14.1%	7.88%	6.1
_All_Sermons		597917	3.64%	18.4%	23.8%	23.6%	12.6%	6.85%	5.0

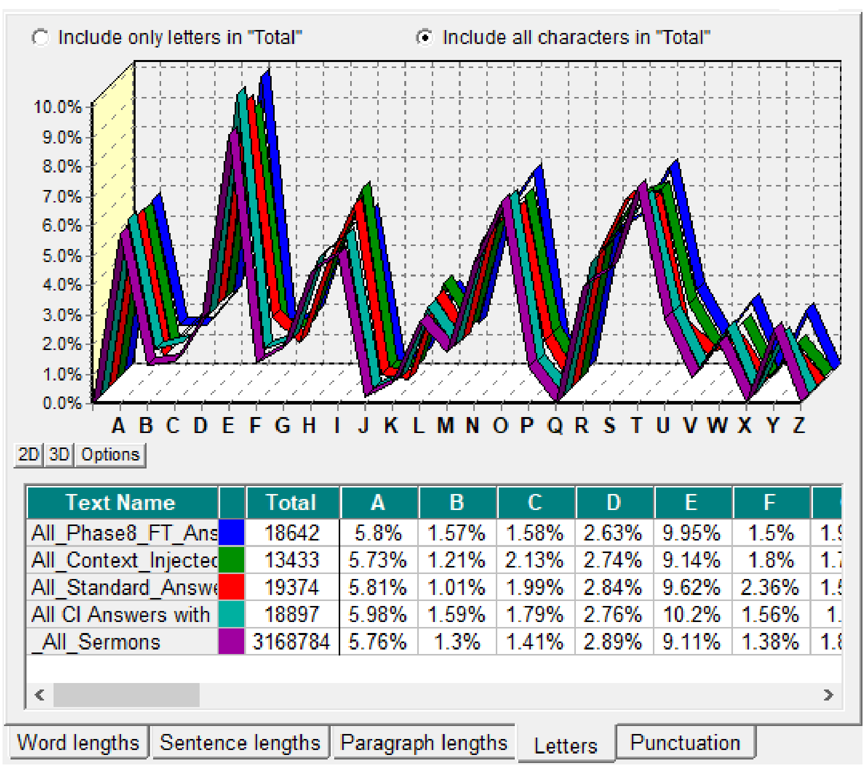
Word lengths | Sentence lengths | Paragraph lengths | Letters | Punctuation

Figure 15: Distribution of word lengths between the document types as displayed in the Signature software.



Text Name		Total	1	2	3	4	5	6	7	8	9
All_Phase8_FT_Ans		1	0%	0%	0%	0%	0%	0%	0%	0%	0%
All_Context_Injectec		2	0%	0%	0%	0%	0%	0%	0%	0%	0%
All_Standard_Answe		6	16.7%	0%	0%	0%	0%	16.7%	16.7%	0%	0%
All CI Answers with		1	0%	0%	0%	0%	0%	0%	0%	0%	0%
_All_Sermons		80	0%	0%	0%	0%	0%	0%	0%	0%	0%

Figure 16: The distribution of the lengths of the paragraphs in the respective document types.



Text Name		Total	A	B	C	D	E	F
All_Phase8_FT_Ans		18642	5.8%	1.57%	1.58%	2.63%	9.95%	1.5%
All_Context_Injectec		13433	5.73%	1.21%	2.13%	2.74%	9.14%	1.8%
All_Standard_Answe		19374	5.81%	1.01%	1.99%	2.84%	9.62%	2.36%
All CI Answers with		18897	5.98%	1.59%	1.79%	2.76%	10.2%	1.56%
_All_Sermons		3168784	5.76%	1.3%	1.41%	2.89%	9.11%	1.38%

Figure 17: The distribution of letters used in the respective document types.

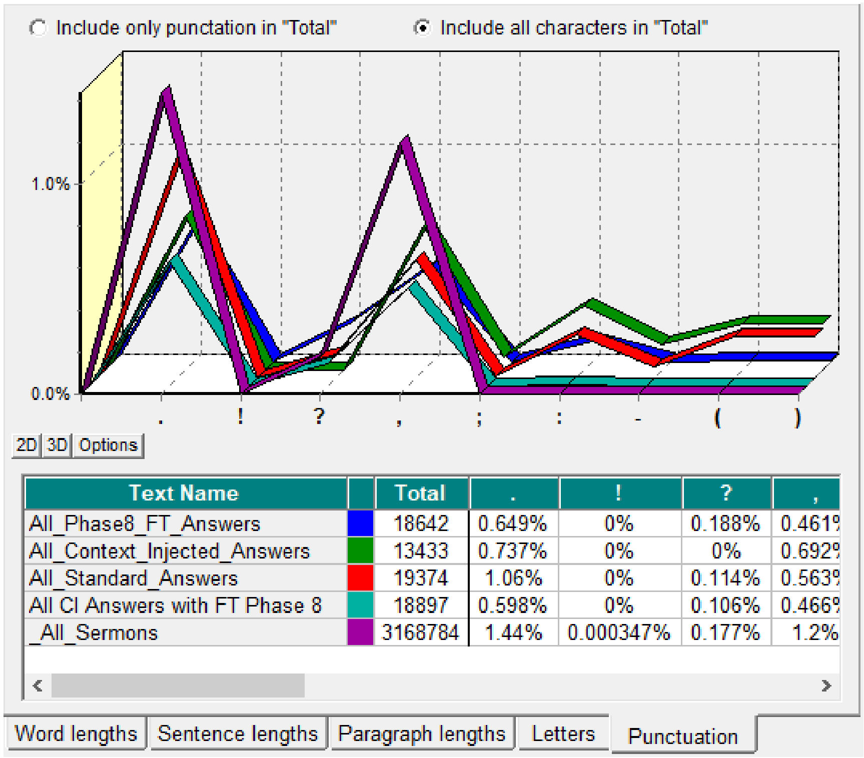


Figure 18: The distribution of punctuation marks in the respective document types.

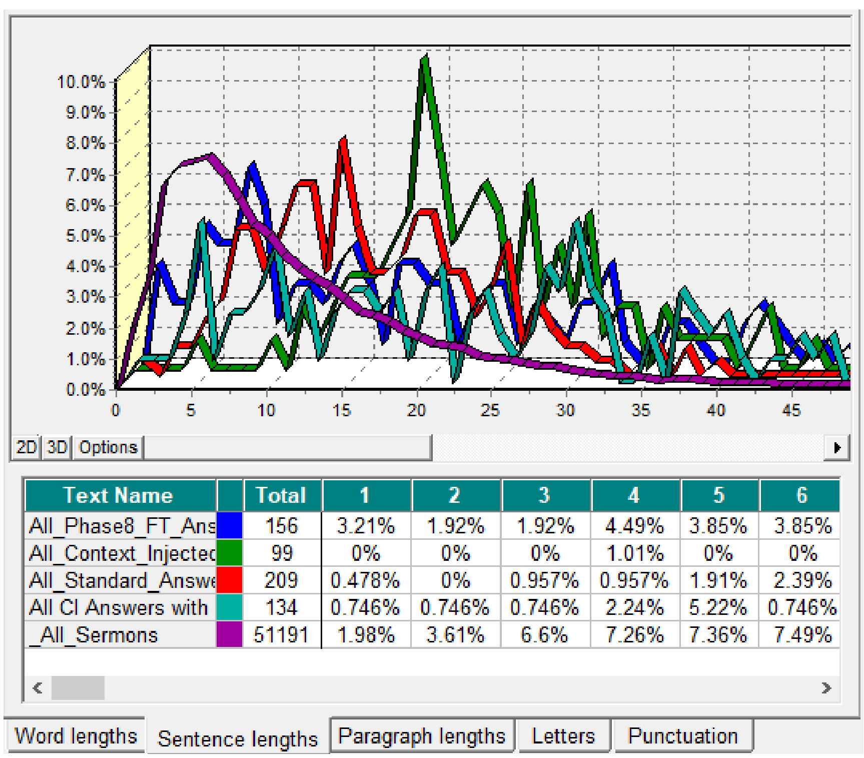
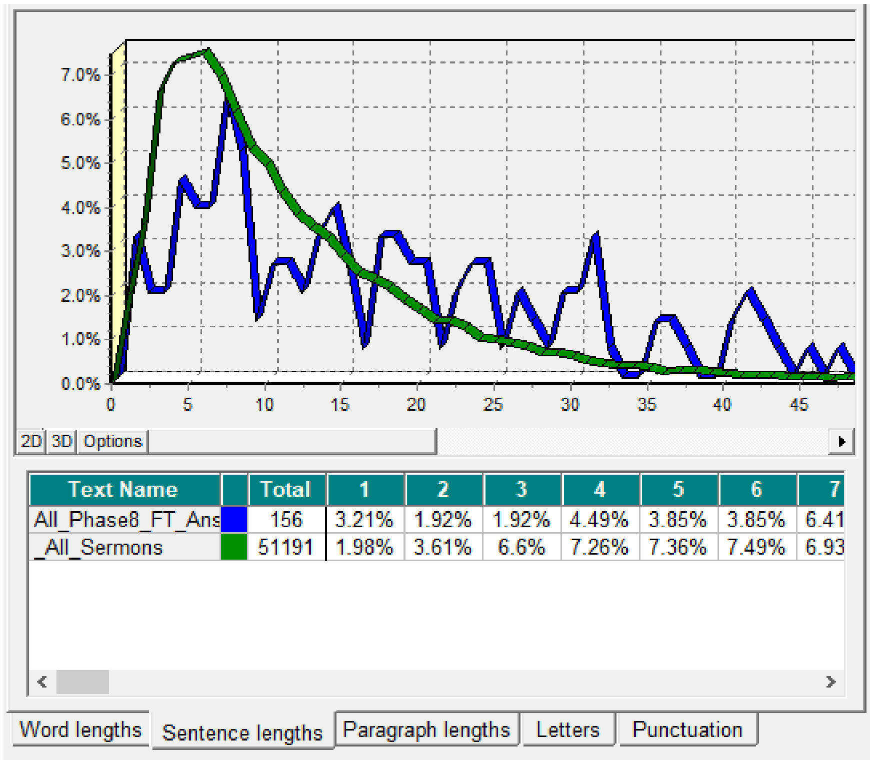


Figure 19: The distribution of sentence lengths in the respective document types.



Text Name		Total	1	2	3	4	5	6	7
All_Phase8_FT_Ans		156	3.21%	1.92%	1.92%	4.49%	3.85%	3.85%	6.41
_All_Sermons		51191	1.98%	3.61%	6.6%	7.26%	7.36%	7.49%	6.93

Figure 20: The distribution of sentence lengths for only the answers generated from the fine-tuned models and the original transcribed sermon text, showing a correlation.

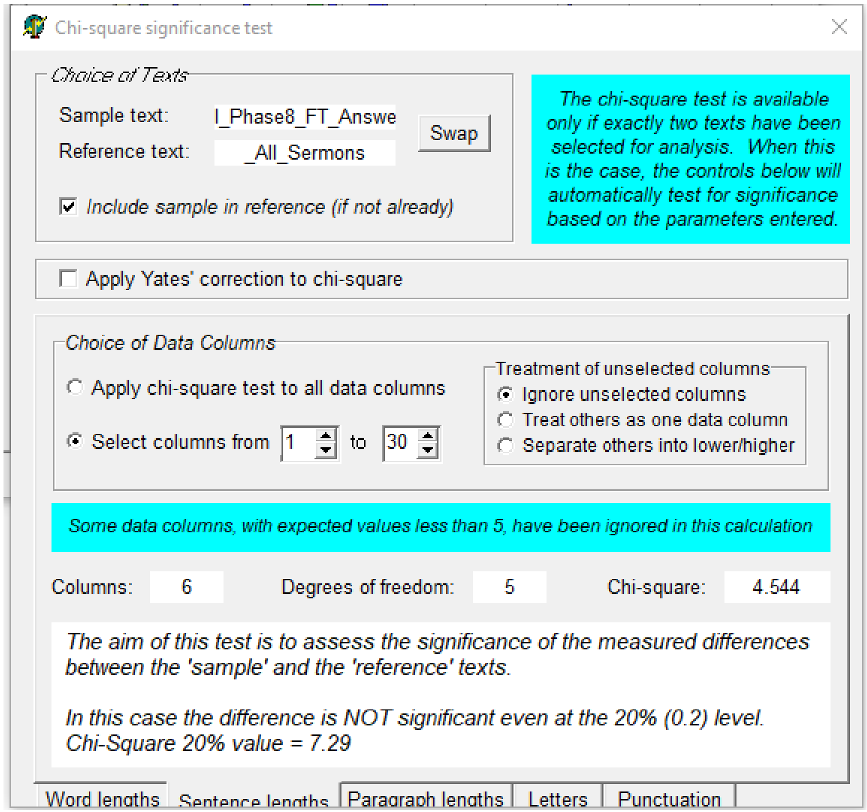


Figure 21: Signature found no significant difference between the sentence lengths of the answers generated by the fine-tuned *models and the original transcribed sermon text.*

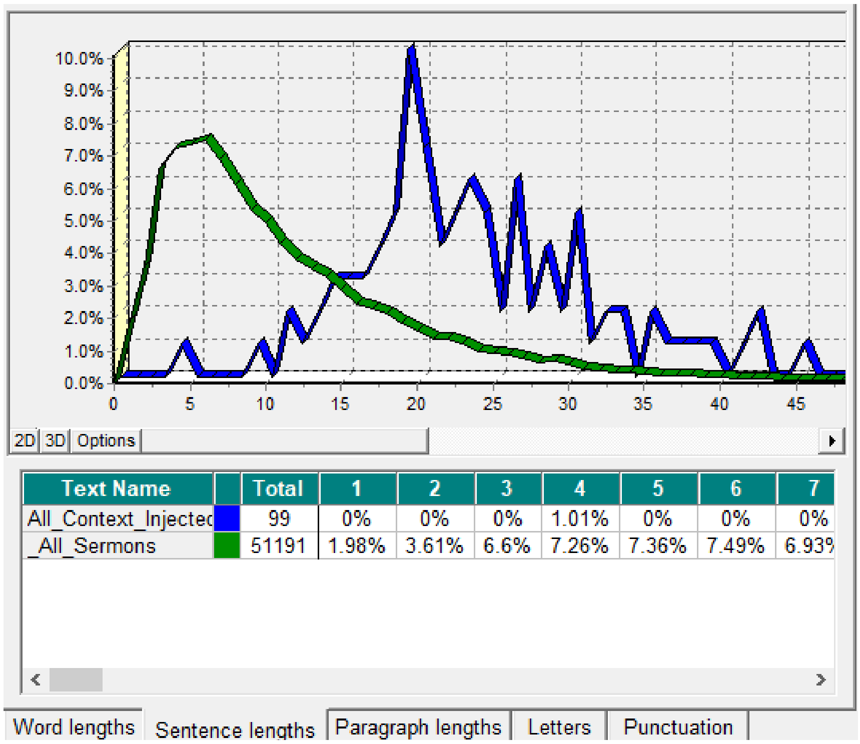


Figure 22: Distribution of sentence lengths for only the answers generated using context-injection and the original transcribed sermon text.

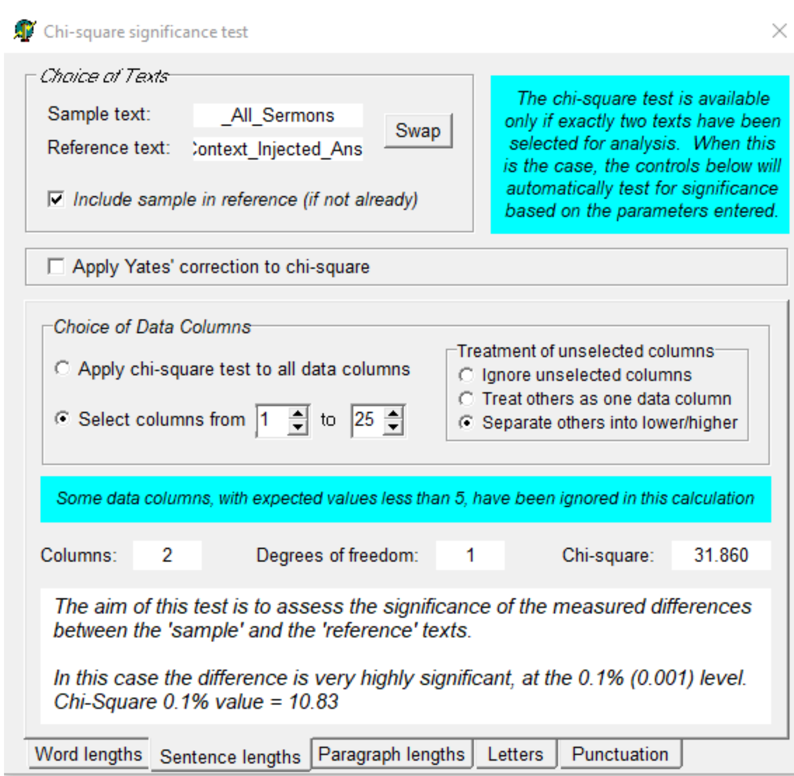


Figure 23: Signature found a significant difference between the sentence lengths of the answers generated with context-injection and the original transcribed sermon text.

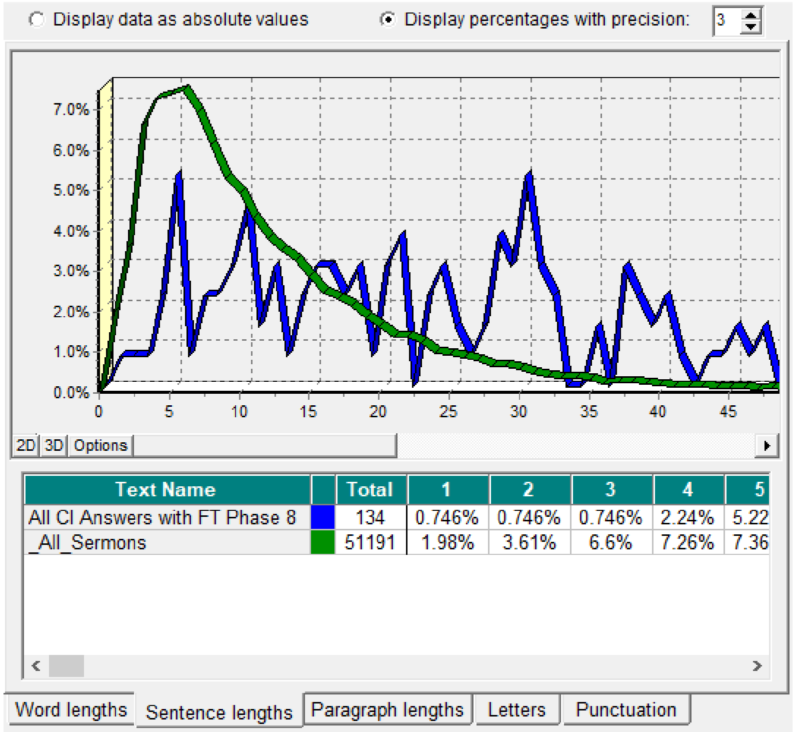
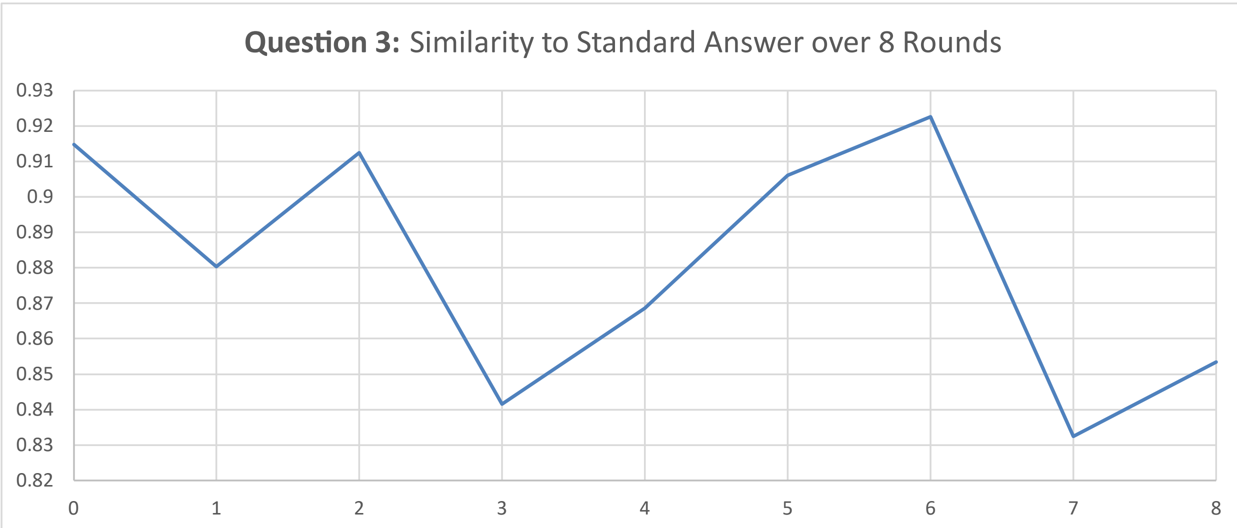
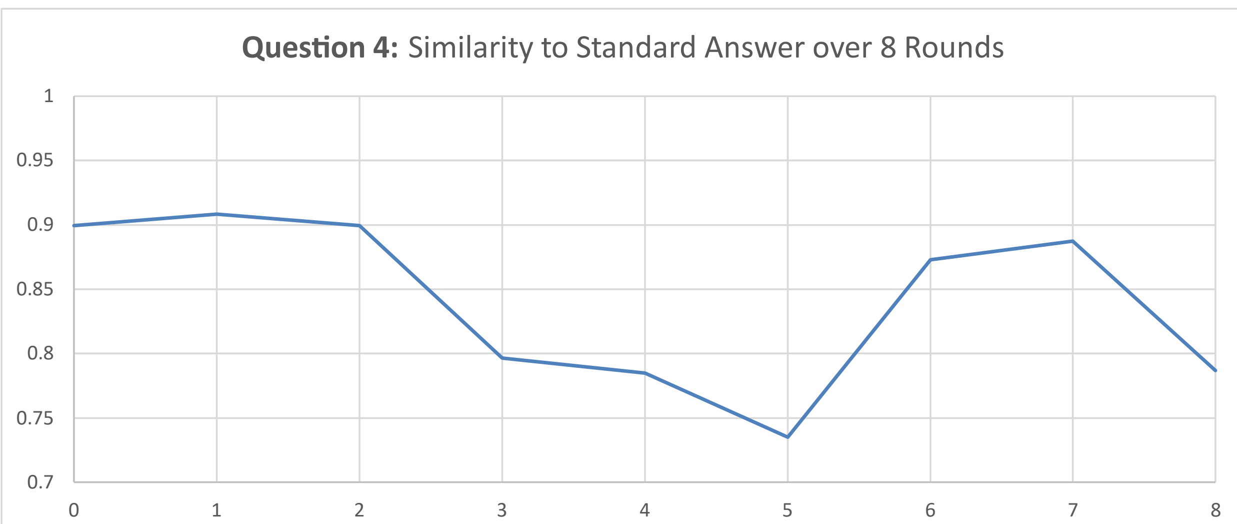


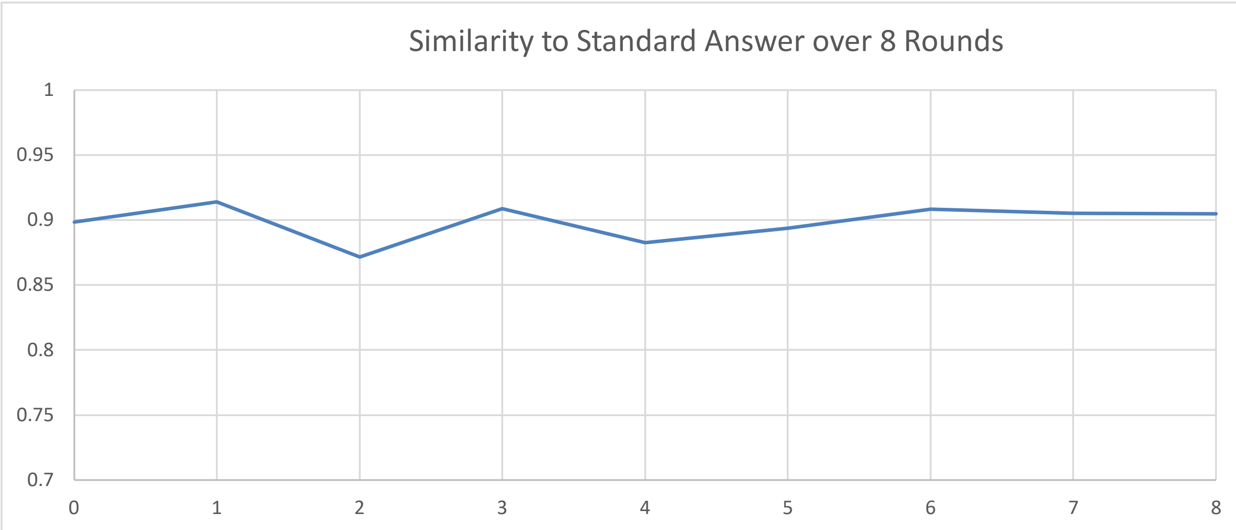
Figure 24: Distribution of sentence lengths for the answers generated with context-injection using the fine-tuned models, and the original transcribed sermon text.

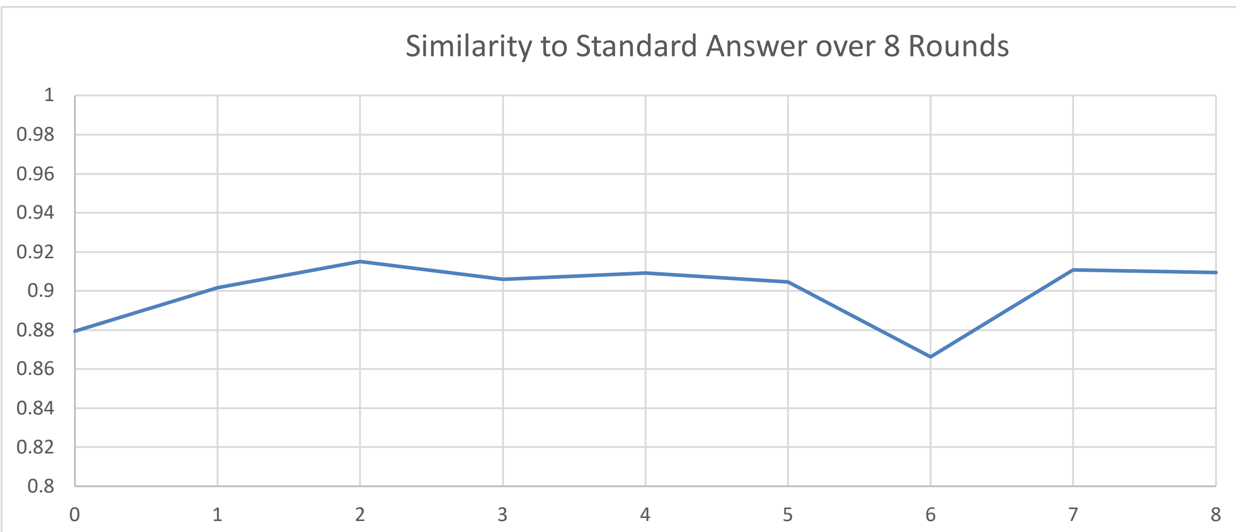
Appendix C: Individual fine-tuning graphs for Question 3 through Question 20

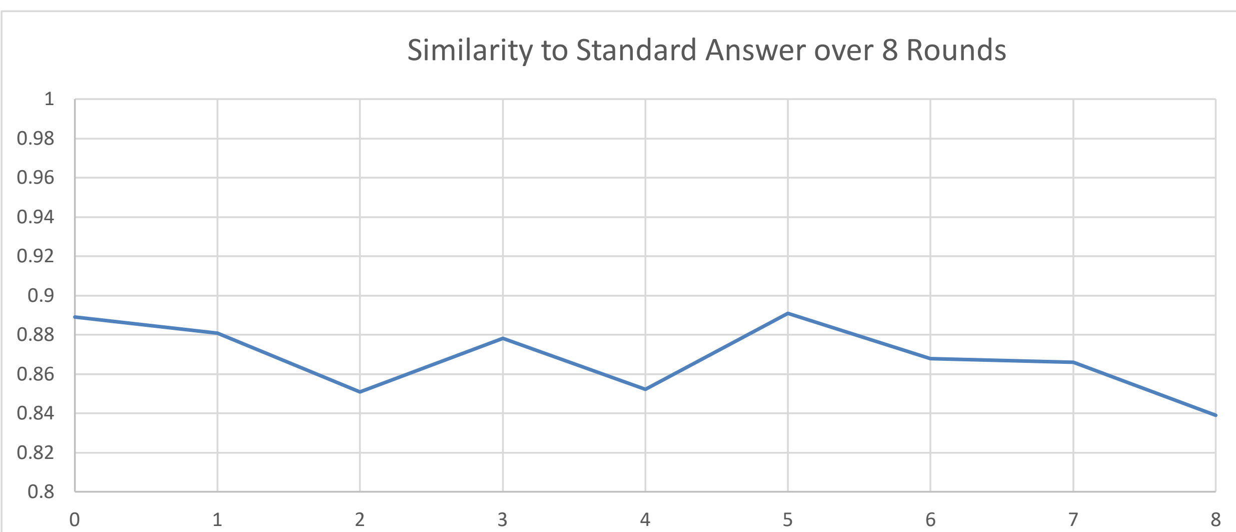
The following graphs (Figures 25 - 42) show the evolution of the similarity scores for each iterative fine-tuning phase 1 - 8 in relation to the standard answer, for each of the remaining questions, Question 3 through Question 20. Phase 0 represents base GPT with no fine-tuning.

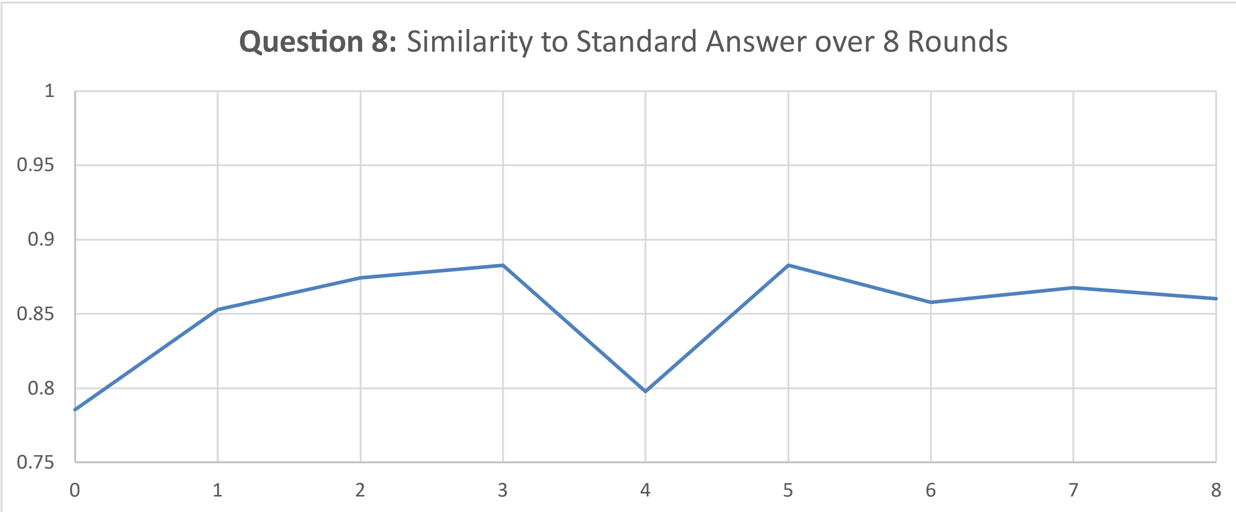




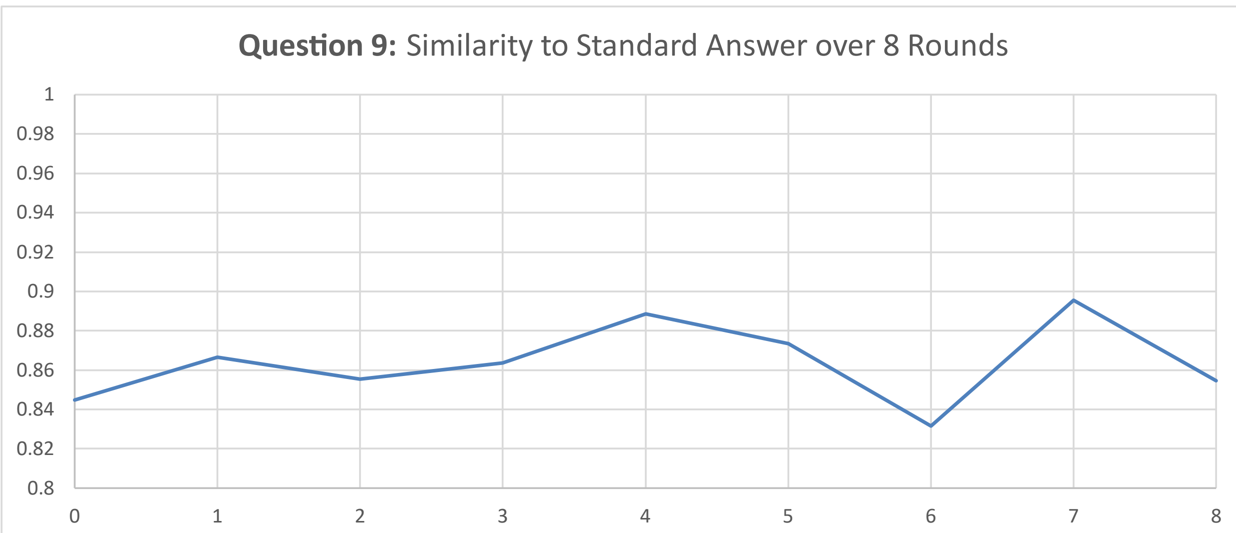

Similarity to Standard Answer over 8 Rounds
1
0.95
0.9
0.85
0.8
0.75
0.7
0
1
2
3
4
5
6
7
8


Similarity to Standard Answer over 8 Rounds
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
0
1
2
3
4
5
6
7
8

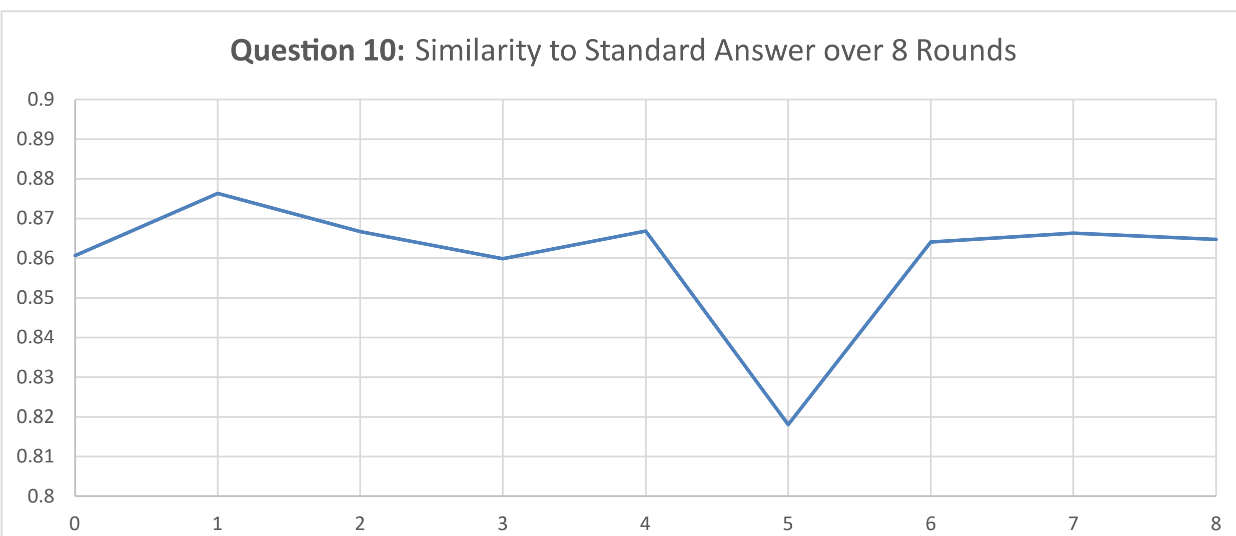

Similarity to Standard Answer over 8 Rounds
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
0
1
2
3
4
5
6
7
8



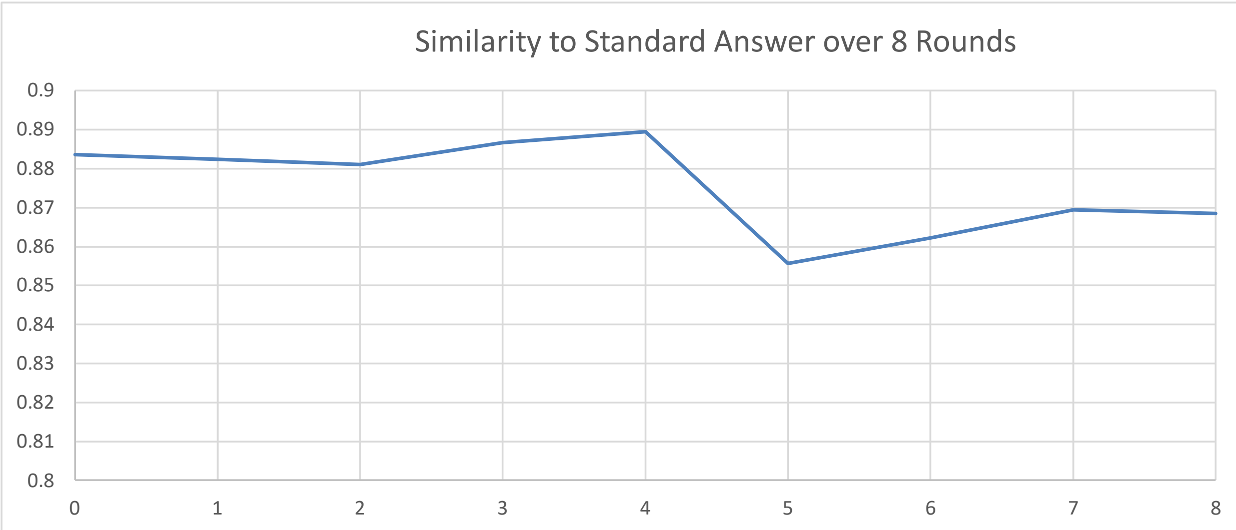
Question 8: Similarity to Standard Answer over 8 Rounds
1
0.95
0.9
0.85
0.8
0.75
0 1 2 3 4 5 6 7 8

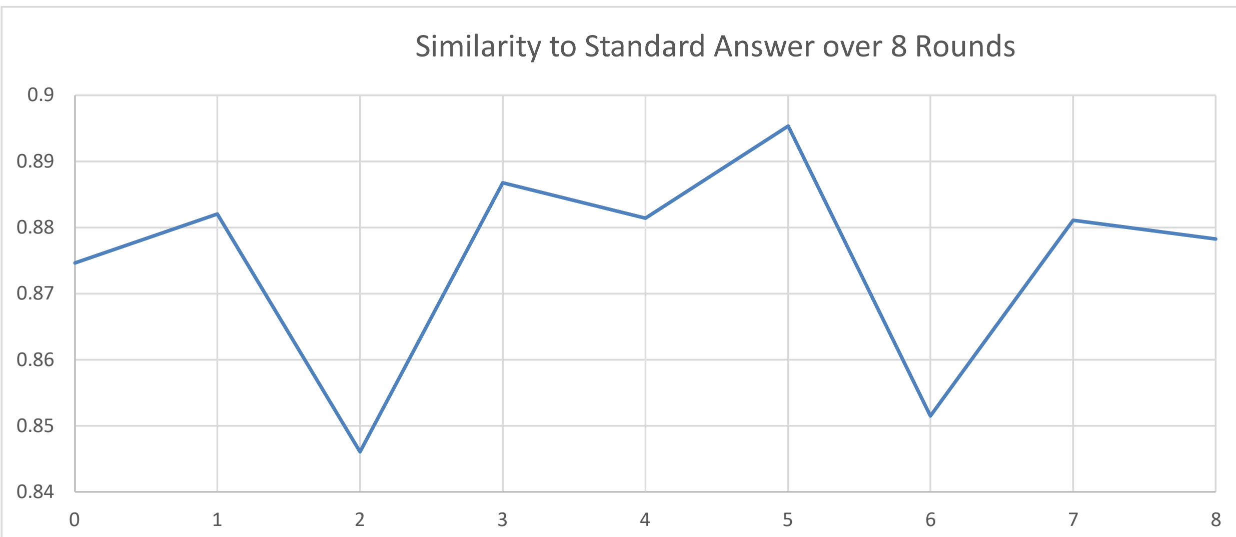


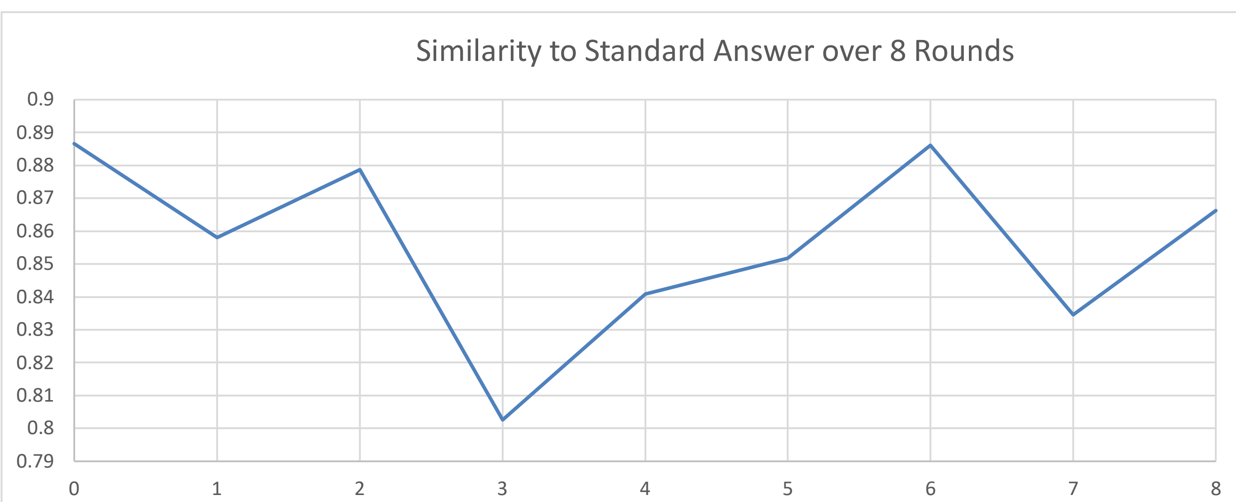
Question 9: Similarity to Standard Answer over 8 Rounds
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
0 1 2 3 4 5 6 7 8

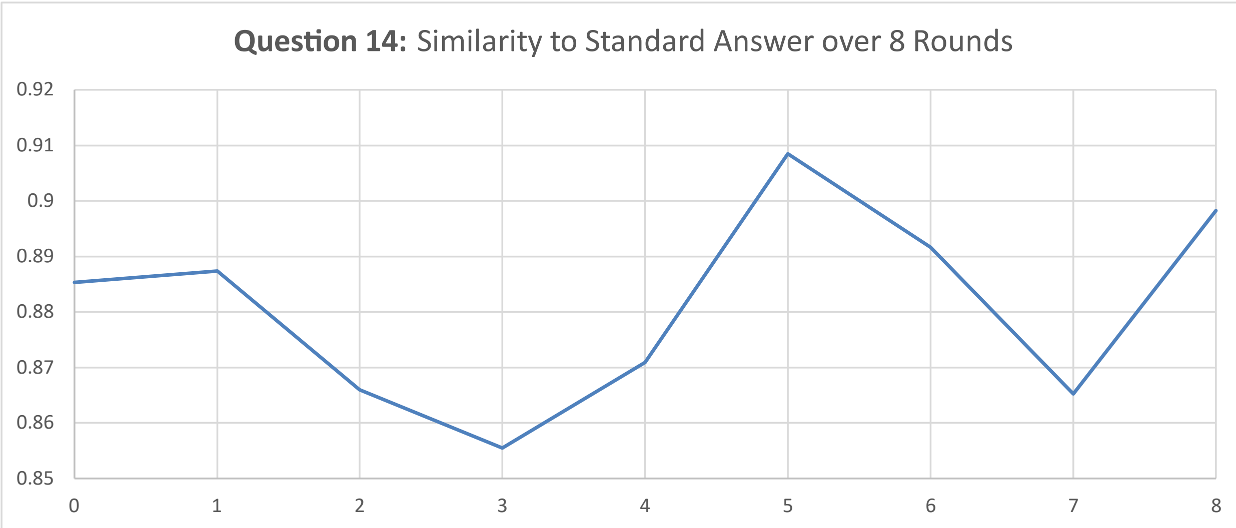


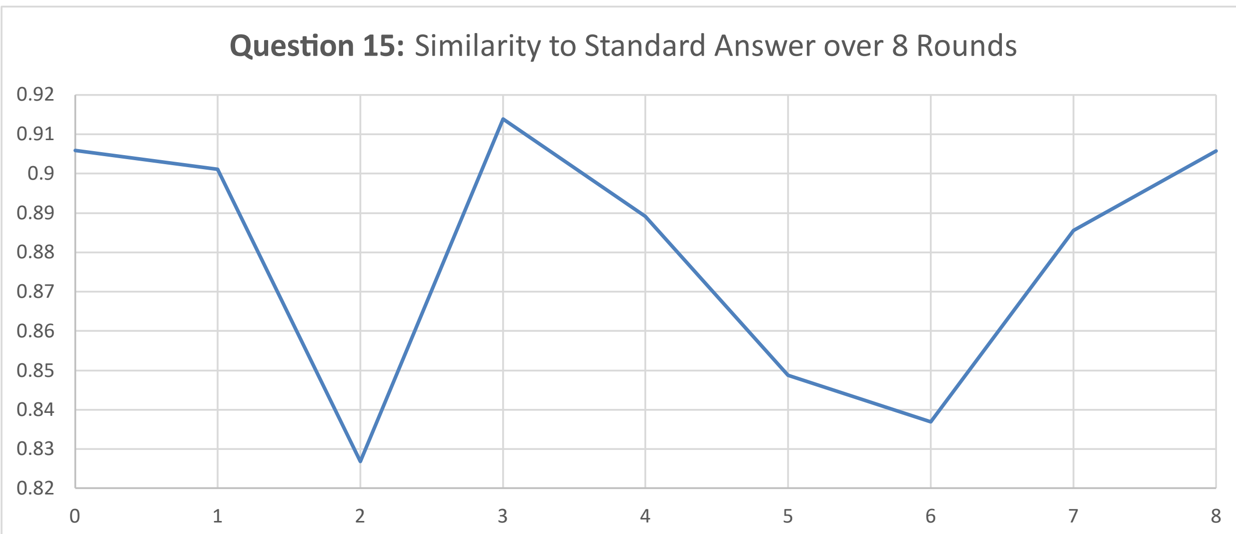
Question 10: Similarity to Standard Answer over 8 Rounds
0.9
0.89
0.88
0.87
0.86
0.85
0.84
0.83
0.82
0.81
0.8
0 1 2 3 4 5 6 7 8

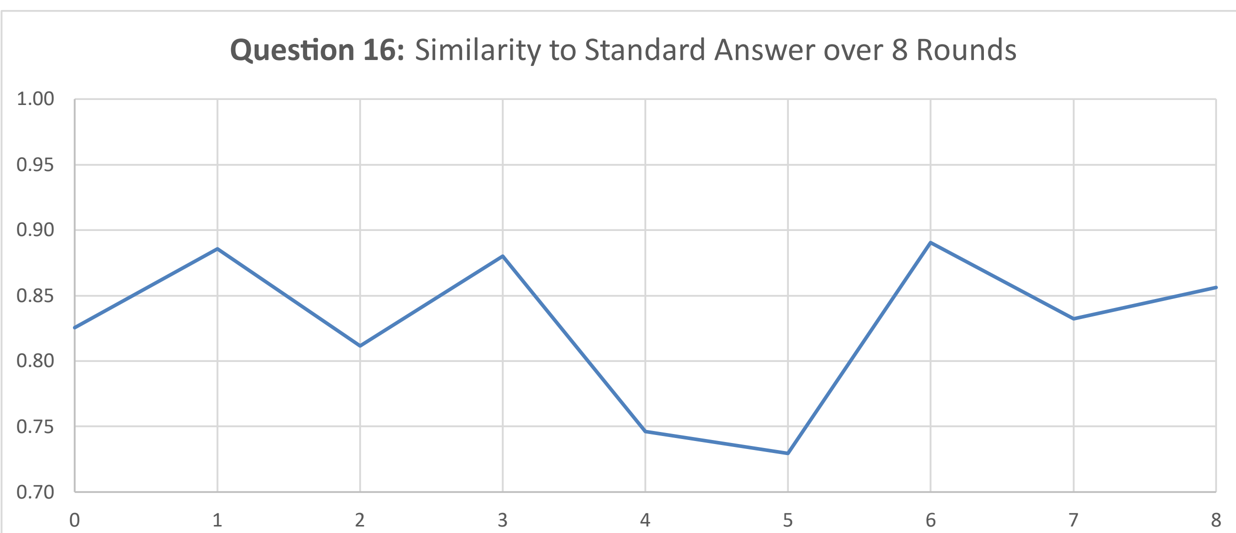

Similarity to Standard Answer over 8 Rounds
0.9
0.89
0.88
0.87
0.86
0.85
0.84
0.83
0.82
0.81
0.8
0
1
2
3
4
5
6
7
8

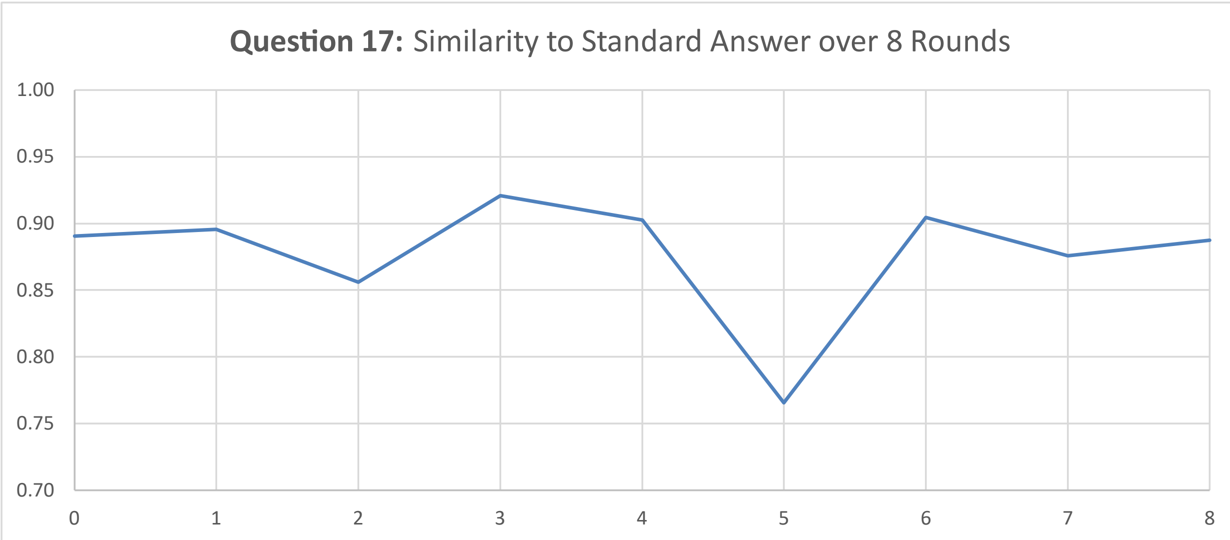

Similarity to Standard Answer over 8 Rounds
0.9
0.89
0.88
0.87
0.86
0.85
0.84
0
1
2
3
4
5
6
7
8


Similarity to Standard Answer over 8 Rounds
0.9
0.89
0.88
0.87
0.86
0.85
0.84
0.83
0.82
0.81
0.8
0.79
0
1
2
3
4
5
6
7
8

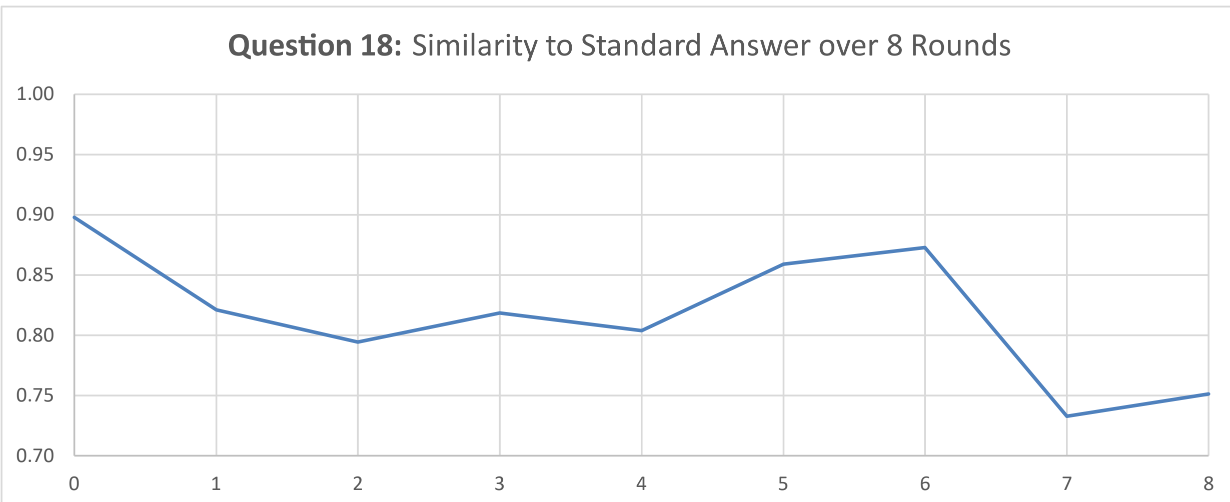

Question 14: Similarity to Standard Answer over 8 Rounds
0.92
0.91
0.9
0.89
0.88
0.87
0.86
0.85
0
1
2
3
4
5
6
7
8


Question 15: Similarity to Standard Answer over 8 Rounds
0.92
0.91
0.9
0.89
0.88
0.87
0.86
0.85
0.84
0.83
0.82
0
1
2
3
4
5
6
7
8

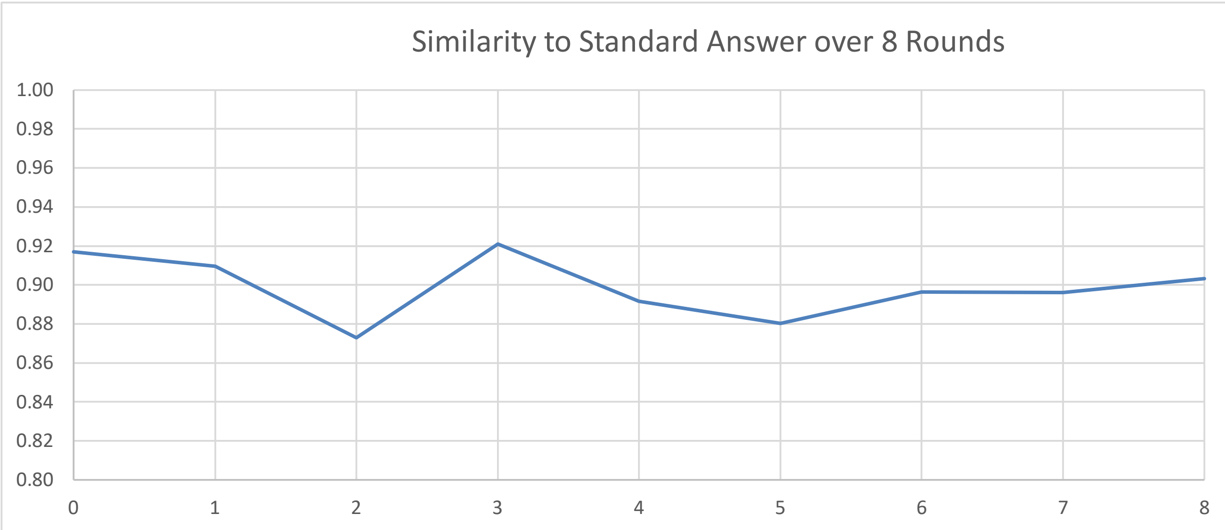

Question 16: Similarity to Standard Answer over 8 Rounds
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
1
2
3
4
5
6
7
8

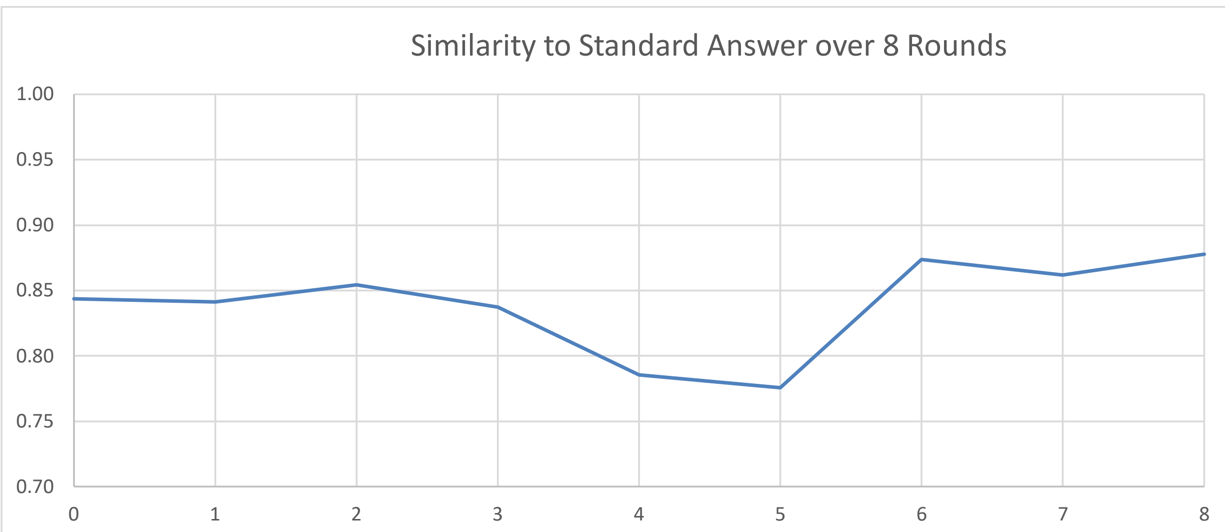


Question 17: Similarity to Standard Answer over 8 Rounds
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0 1 2 3 4 5 6 7 8



Question 18: Similarity to Standard Answer over 8 Rounds
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0 1 2 3 4 5 6 7 8


Similarity to Standard Answer over 8 Rounds
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0
1
2
3
4
5
6
7
8


Similarity to Standard Answer over 8 Rounds
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
1
2
3
4
5
6
7
8