Retrieval-Augmented Generation and In-Context Prompted Large Language Models in Aircraft Engineering

Edmar A. Silva [*], Robert Marsh[†], Hau Kit Yong[‡], Stuart E. Middleton[§], and András Sóbester[¶]
University of Southampton, Southampton, SO17, United Kingdom

With the aerospace industry taking its first steps towards exploiting the rapidly evolving technology of Large Language Models (LLMs), this study explores the potential of the latest generation of LLMs to become an effective link in the aircraft design tool chain of the future. Our focus is on the task of Question Answering (QA) in engineering, which has the potential to augment future aircraft design team meetings with an intelligent LLM-based agent able to engage with the team via a chatbot interface. We compare three of the most effective and popular classes of LLM QA prompting today – LLM zero-shot prompting, LLM in-context prompting and LLM-based Retrieval-Augmented Generation (RAG). We describe a new, low volume, high quality benchmark aircraft design QA dataset (AeroEngQA) and use it to qualitatively evaluate each class of LLM and exploring properties including answer accuracy and answer simplicity of the answer. We provide domain-specific insights into the usefulness of today's LLMs for engineering design tasks such as aircraft design, and a view on how this might evolve in the future as the next generation of LLMs emerges.

I. Introduction

Recent dramatic improvement in the capabilities of Large Language Models (LLM) are fundamentally redrawing many areas of human endeavour and engineering is no exception, with software development reaping perhaps the largest rewards to date. Is aircraft engineering likely to benefit in similarly significant ways in the near future? This paper empirically evaluates three classes of LLM-based Question Answer (QA) model, using zero-shot prompting, in-context prompting and Retrieval-Augmented Generation (RAG) approaches, exploring the potential for LLMs to become powerful tools in accelerating or otherwise enhancing tasks such as air vehicle development. We chose these three LLM classes as they do not require expensive domain-specific fine-tuning and thus represent what can be achieved today with off-the-shelf LLM models, something many aerospace engineers have access to today via services such as ChatGPT.

A key step in the operation of LLMs is prompting, the addition of some carefully constructed text as a prefix to a task specific input, such as a question, prior to submission to an LLM like ChatGPT. An example of a QA-focused LLM prompt might be "Answer the following question using the answer context provided." For QA tasks, zero-shot prompting provides the LLM with a question and answer context, then asks the LLM to generate an answer from that answer context. An answer context is one or more paragraphs of text from a document source, such as a set of aerospace technical documents, that contains the answer to the question. In-context prompting is the same as zero-shot prompting but with some positive and negative examples included in the prompt. RAG uses a question prompt and an Information Retrieval (IR) search model to discover and rank relevant answer contexts, and then asks the LLM to generate an answer using only that answer context. It should be noted that in-context prompting [1] is a type of few-shot learning [2], where support set examples are provided with the input to help the model generate the correct answer; sometimes these terms are used inter-changeably when discussing LLMs.

The aerospace industry has, historically, taken a cautious approach to the adoption of machine learning (ML) technology in general – even Bayesian surrogate models (e.g., Kriging) learnt from the outputs of expensive black box simulations [3] took decades before becoming a near-universally accepted part of the aerospace engineer's everyday tool set. LLMs are likely to face more barriers still, for a variety of reasons. First, explainability and traceability have long been fundamental requirements in an industry controlled through strict product certification processes, and these

[*]Senior Research Fellow, Faculty of Engineering and Physical Sciences
[†]Senior Research Fellow, Faculty of Engineering and Physical Sciences
[‡]Research Fellow, Faculty of Engineering and Physical Sciences
[§]Associate Professor, School of Electronics and Computer Science, Senior Member ACM, Member ACL
[¶]Professor, Faculty of Engineering and Physical Sciences, Senior Member AIAA

are well-known weaknesses [4] of the current crop of off-the-shelf LLMs. Second, any design tool must guarantee the protection of intellectual property (see Ref. [5] on the UK National Cyber Security Centre's view on the subject), the strictness of this requirement stemming partly from the long lead times typical of the industry, meaning long vulnerability periods in terms of potential loss of competitive advantage; models relying on a real time two-way link with the world wide web are likely to fall foul of this requirement. Third, ML tools are most likely to shine when extensive legacy data is available; as we head towards a first generation of large transports that will not be powered by fossil fuel burning turbofans, the value of such legacy data can be expected to wane. And fourth, the cost of incorrect information feeding into the design process can be enormous, especially in terms of project delays, and thus there is a need for constant vigilance against potential AI hallucinations [6, 7] (well documented even in the everyday use of LLMs) that might cancel out any benefits.

This paper adopts an engineering design viewpoint and, with the constraints listed above in mind, formulates two fundamental questions. First: what class of LLM has the greatest potential to enhance aerospace engineering development processes? Second, what role might an LLM best fit into this process in a real-world socio-technical deployment alongside human engineers?

We focus on LLMs in a socio-technical role as a digital assistant, working alongside humans and always at the fingertips of the engineering workforce of a company or specific project team regardless of their place in the product development timeline. To allow authentic evaluation, we create *AeroEngQA* [8] , a novel benchmark aircraft design QA dataset with 80 aircraft engineering QA pairs derived from contexts drawn from publicly available NASA and National Transportation Safety Board (NTSB) technical reports and engineering patent files. Following best practice in designing QA datasets for LLM evaluation [9, 10], AeroEngQA has an even balance between answerable and unanswerable questions, simple and complex reasoning, and short and long answer context lengths. A team of aerospace experts was used to both design the questions and select the correct answers. Using AeroEngQA we qualitatively evaluate our three classes of LLM, reporting results around answer accuracy and answer simplicity. The LLM models tested are *GPT 3.5-turbo*, *GPT-4*, and *LLaMa3*. We release AeroEngQA as an open source data set, to provide an important contribution enabling more authentic QA evaluation in the future by LLM and aircraft design researchers. This paper provides a contribution around the following research questions:

1) How accurate are RAG and in-context prompted LLMs when answering questions in an engineering domain? Are the answers simple enough to be understood quickly by engineering teams?
2) In the socio-technical context of engineering teams, what roles are LLMs ready for today? Given the pace of improvement of LLM technology, how is this likely to change in the future?

Following a review of QA datasets, QA models, and their applications in the aerospace domain, in Section III we describe the principles behind the construction of AeroEngQA; here we also go into more depth on the prompting techniques we are using in the paper. Section IV contains the results of our comparisons of the various models and prompting methods on AeroEngQA, followed by an analysis of the results in Section V.

II. Related Work

A. Question Answer Datasets

There has been significant research within the Natural Language Processing (NLP) community, where LLMs originated from, into LLMs for QA. To quantitatively evaluate and compare QA performance a number of benchmark QA datasets have been proposed by the community that span a variety of domains. The most widely used is perhaps the SQUAD2 [11] dataset which contains 100,000 questions posed by crowd workers based on Wikipedia articles, with half of these questions deliberately unanswerable. QA pairs consist of a question, a Wikipedia article which contains enough information to answer the question and a gold standard answer (which is used to score answer predictions). To evaluate questions that requires more complex reasoning to answer, multi-hop reasoning datasets such as HotpotQA [12] have been proposed, with 113,000 questions based on Wikpedia articles but requiring some multi-hop reasoning steps to answer, such as the using bridging entities to connect information between two articles. QA datasets for technical domains tend to be much smaller in size and more costly to annotate due to the expertise required to answer authentic questions. One of the largest examples is TechQA [13], where 900 questions were sourced from IBM technical support conversations of which 300 were unanswerable. Recent work [14] has also explored how in-context prompting of LLMs can synthesise millions of new QA pairs, with answers that require both single and multi-hop reasoning. These synthetic datasets can augmented human-authored QA training data when training QA models, but for evaluation purposes testsets containing human authored QA pairs are still preferred on the basis of answer quality.

Evaluation metrics for QA tasks follow either a quantitative, qualitative or mixed approach. For quantitative QA evaluation [11] gold answer text is compared to predicted answer text and metrics such as Exact Match and F1 score computed across the QA dataset. Exact Match score is the fraction of correct answers reported, with a correct answer where all the characters of a predicted answer exactly match the ground truth answer. F1 score is a hybrid of precision and recall metrics, computed using a character-level true positive count which scores all predicted answer characters that overlap with the ground truth answer characters. The F1 score thus allows room for a model to get an answer almost right, or completely right but with some additional characters added such as an explanation, grammar or simply whitespace characters. For qualitative evaluation random samples of answers are typically scored by experts using a pre-agreed coding schema which is often task specific. Ideally there should be a team of experts so variations in human judgement can be discussed and a consensus score found. Qualitative evaluation is the gold standard, but due to the human effort involved it is often limited in scale to a fraction of the QA dataset. Quantitative evaluation is a less authentic evaluation of answers but can cover the full QA dataset.

The only publicly available QA dataset in the aeronautical engineering domain (as far as the authors are aware) is AeroQA [15], which was created using ChatGPT to synthetically generate 27,000 QA pairs, with a mix of answers requiring both single and multi-hop reasoning, from a dataset of technical reports from the National Transportation Safety Board (NTSB). We performed a manual expert review of a random sample from the AeroQA dataset and found the questions were not authentic, to what a typical engineering team might ask in a design meeting, and the answers of a variable quality – something that should be expected when they are machine generated. Overall AeroQA represents an aerospace domain QA dataset that is high volume and low quality. The new AeroEngQA dataset we present in this paper is the opposite of AeroQA; it is low volume and high quality and as such represents an important complementary resource for high quality authentic evaluation of QA models. Our dataset is high quality because we used a team of four expert aerospace engineers to pose 80 questions and answers from an authentic public aerospace document corpus, including reports from NASA, NTSB and engineering patents.

B. LLM-based Question Answer Models

Before the recent advances in LLMs, the most powerful QA models were created by fine-tuning a pre-trained masked language model such as BERT [16] and ALBERT [17]. The fine-tuning process trains these pre-trained masked language models to perform answer space identification for QA, as opposed to simply performing masked word prediction. These fine-tuned QA models are small in comparison to LLMs, with each QA model having well under 1B parameters, in part because fine-tuning larger models is not GPU memory efficient. Interestingly, recent training methods such as MeZO [18] are making important strides in reducing the memory requirements for fine-tuning, so this limitation might change in the future.

As generative LLMs have grown in size (first GPT-2 with 1.5B parameters, GPT-3 with 175B parameters, and now GPT-4 rumoured to have over 1.5T parameters) their abilities to perform QA has caught up with what can be achieved through fine-tuning a smaller masked language model. To perform QA a LLM must be prompted, either zero-shot prompting where the question is asked directly or in-context prompting where a couple of examples of how questions should be answered are provided in addition to the zero-shot prompt. The top scoring masked language model at the time of writing for the Natural Questions QA dataset[*], a standard benchmark for QA models, is PoolFormer [19], which is based on a pre-trained RoBERTa-large 355M parameter model and achieved a F1 score of 0.79. The best generative LLMs performance reported by HELM[†] on the same dataset is Microsoft's GPT-4[‡] with a F1 score of 0.79, and in second place Meta's LLaMa3 70B parameter model[§] with a F1 score of 0.743. LLMs have now caught up in terms of QA performance and are likely to surpass them soon, making the work in this paper a very timely and important evaluation of how capable LLMs are for authentic aerospace domain QA tasks.

Another approach to QA is to use Retrieval-Augmented Generation (RAG), where questions are asked without any text that contains the answer. This is a harder QA problem formulation, a Retrieve and Read approach [20], where the RAG model must first search a corpus, either a closed document corpus such as a technical report library or an open corpus such as the web, to identify documents likely to contain the answer based on the question asked. Then, the RAG model must generate an answer using the set of relevant documents found. LLM-based RAG models [21] typically use fine-tuned small LLMs such as T5 [22] or BART [23]. Various hybrid methods have been proposed, such as reinforcement learning applied to augment RAG [24], and LLMs have been explored as a potential replacement to the

[*]https://ai.google.com/research/NaturalQuestions/leaderboard

[†]https://crfm.stanford.edu/helm/lite/latest/

[‡]https://openai.com/index/gpt-4-research/

[§]https://llama.meta.com/llama3/

retrieval step, with a Generate and Read approach [25]. Given the popularity and effectively of RAG methods, this paper evaluates an LLM-based RAG model as well as zero and in-context prompted LLMs to give a representative comparison of performance of these classes of model.

C. LLMs in the Aerospace Domain

The last few years have seen the aerospace industry take its first tentative steps towards employing LLMs also. Named Entity Recognition (NER), the classification of entities from a text into predefined categories, is one of the low-hanging fruits, showing some early promise in requirements management [26]. Along similar lines, the authors recently explored the use of LLMs as a tool for the rapid, semi-automated enhancement of design rationale records linked to a geometry model [27]. A team from Boeing has recently reported the commencement of work towards a digital assistant supporting the aircraft certification process [28], their paper highlighting some of the challenges listed above, as well as issues around the acquisition of training data, which might be residing in 'antiquated databases without API access'. Spelling errors and incompleteness are a related challenge, as identified in initial experiments by Connoly and Anderson [29] (also in the context of requirements analysis). Certification requirements [30] form the starting point of the work reported by Jin et al. too [31], who employ an LLM in the identification of whether requirements have been met, as evidenced by the output of a system simulation. Despite these existing works, to the best knowledge of the authors there are no LLM-based papers focussing on the task of aircraft design QA and no aircraft design QA datasets that allows authentic evaluation of such LLM models; this paper addresses both gaps.

III. Method

A. AeroEngQA – an Aircraft Engineering Specific, Human-Annotated Question and Answer Benchmarking Set

The goal of this paper is to assess the suitability of LLMs as engineering design tools by simulating a realistic engineering context. To that end, we propose to assess the responses of state-of-the-art LLMs in a way that best matches their possible use in the design department of an aircraft engineering company. This means that:

1) We assume that the designer wishes to ask questions answerable on the basis of the company's proprietary, confidential document database, which is held onsite. We therefore benchmark the capabilities of the LLMs being tested through question answering in relation to given contexts (sections of text, based on which the question should be answered). The contexts are supplied to the LLM via the prompt.
2) The contexts and questions that make up the benchmarking set must be context-specific, in this case they must be related to aircraft engineering.
3) Both the questions and the correct answers in the benchmarking set must be created by human annotators, who also assess the LLM-generated responses for correctness.
4) All human annotators must be experienced engineers, with a significant track record in aeronautical engineering.
5) The questions should cover a range of aeronautical engineering topics, with a focus on aircraft design. To maximise diversity, the source document of each context-question-answer entry is unique.
6) We assume that a designer may wish to ask questions that simply retrieve information (much like a conventional search engine), or questions that require reasoning of varying complexity.
7) The ability of a tool to make it clear that a question is not answerable based on the text provided is just as important as answering correctly when possible; erring on the side of attempting some form of answer in all circumstances is considered pernicious in a design context.

In designing *AeroEngQA*, our proposed question answering benchmarking set, we adopted two additional principles. First, we wish to share it with the broader community, which means ensuring that the contexts are extracted from public domain documents – we used NASA Technical Reports and Contractor Reports, National Transportation Safety Board incident and accident reports, and patents. Second, we focused on quality over quantity: *AeroEngQA* consists of a relatively small number of entries (80 context - question - answer sets in total), but no LLMs or automation of any form were used to generate these. The annotator team consisted of four of the authors of this paper, representing a spread of engineering research and industrial experience. Each annotator holds an engineering degree, with three members of the team also holding doctoral degrees (in engineering).

The AeroEngQA questions that require reasoning fall into two categories: 1) single hop questions (simple reasoning) and 2) multi-hop (complex reasoning, bridging entity involved). Further, the test data set includes both unanswerable and answerable questions. Table 1 shows an answerable, multi-hop, multi-doc question, which we class as 'easy', as it does not require complex reasoning.

Context 1	*This flow control approach was conceived from the 'Top-wing' (Section III.A.4) and 'Main wing' (Section III.A.5 applications. For convenience, the Mach number distributions at a spanwise cut in the LE region obtained with these two approaches are presented in Fig. 17. The gapped slat variant is obtained by introducing a small gap between the slat and main wing. The flow control is provided by a jet similar to the one shown in the 'Main wing' inset. Here, the jet is injected at a 20° angle to the surface toward the throat section. Essentially, the gap creates an effective converging nozzle, so that the jet emerges over the upper surface of the wing, similar to the jet ejection in the 'Top-wing' case.*
Context 2	*The converged nozzle helps accelerate the jet to higher velocity, thus providing high momentum and amplifying flow control effectiveness. In yet another variant, also included in Fig. 17, the gap is designed to create an effective convergent/divergent nozzle so that the jet can be accelerated to supersonic flow for even higher momentum. The convergent/divergent gap is used in the following results. The gapped slat application can be powered by either APU or local compressors.*
Source	Ref. [32]
Question	Where is the gap that is used to accelerate the jet to supersonic flow positioned?
Answer	Between the slat and main wing.

Table 1 Example of a multi-hop, answerable question from AeroEngQA, highlighting the reasoning bridging entity in blue and the correct answer (as extracted from the context) in yellow. The context element referenced by the question is highlighted in green for easier reading; note that the wording does not need to be the same (unlike in the case of the answer, which is taken unchanged from the context.

We translated the above requirements and principles into the following set of guidelines, which the team used to select the contexts and write the questions and answers that make up AeroEngQA:

1) Context selection
   - For single-hop questions there should be one context instance. For multi-hop questions there should be two context instances.
   - Context sources: must be high quality primary sources and in the public domain.
   - Context instances for a multi-hop question should be from the same source.
   - Questions should not share context source (each question uses a different document).
   - The contexts must not contain equations, figures, tables. Very simple plain text equations are permitted (e.g., 'L/D').
   - The contexts must be cleaned up manually to remove pdf to plain text conversion artefacts.
2) Question formulation
   - Questions should be asked in your own words, not deliberately copying from the context instance unless that is how you would ask the question normally.
   - Questions proposed should be discussed as a group to enable calibration.
   - Questions should be categorized as easy (no reasoning needed) or hard (reasoning needed e.g. bridge entity, simple maths) or very hard (engineering level reasoning needed e.g. multi-bridge, complex maths, basic engineering terms like full-span)
   - Questions should be both single and multi-hop, with the experiments designed for a balanced mix of types.
   - The test set should include a balanced mix of answerable and unanswerable questions, with a target of 20 questions in each set.
   - Questions should have semantic connections with the context, even if unanswerable.
3) Answer formulation
   - Answers should be extractive (or derived from extractive text, e.g., via simple calculations). This means only cut and paste from context instance(s) is allowed, and answers might not be grammatically correct as a human would answer.
   - Answers should be as short as possible, whilst still answering the question. They should not be complete sentences.

The AeroEngQA dataset is available at https://doi.org/10.5281/zenodo.14215677, hosted on Zenodo. The dataset consists of 80 high-quality, human-annotated question-answer sets derived from publicly available aerospace documents such as NASA reports, NTSB reports, and patents. The dataset is designed to evaluate QA models in both simple and complex reasoning scenarios.

B. LLM Modes of Operation

We shall consider three basic modes of LLM operation in the experiments described in this paper, each representing a possible mode in which a design engineer might use an LLM to query a company document library.

1. Zero-shot Prompted LLMs (GPT3.5_turbo_zeroshot, GPT4_turbo_zeroshot)

In this mode the LLM model is given a task description or a question without any example of how to solve it. The model is expected to generate a response based on its pre-existing knowledge and understanding of language. No task-specific fine-tuning is performed on the LLM. This is the simplest form of prompting, as it requires no additional training data, making it cost-effective and time-efficient. The possible downside (which we aim to investigate here) is that this method can lead to less accurate or less contextually appropriate responses, especially for complex or domain-specific tasks. The model's performance can be inconsistent or unpredictable, as it relies heavily on the model's pre-existing knowledge and understanding.

Below is a zero-shot prompt template we used to generate the results presented in this paper in the case when a single context is available:

"*Query: Generate your response to the forthcoming question solely based on the provided context, refraining from external knowledge. Employ serialized data in JSON format, with the field name "answer". This prompt comprises a single contextual passage and a question. Your answer should be concise and directly derived from the context provided.*

Context: < Answer context text inserted here >

Question: < Question inserted here >"

Here is an equivalent example of the same type of prompting, this time when used for multi-context queries:

"*Query: Generate your response to the forthcoming question solely based on the provided context, refraining from external knowledge. Employ serialized data in JSON format, with the field name "answer". This prompt comprises two contextual passages and a question. Your answer should be concise and directly derived from the context provided.*

Context 1: < Answer context 1 text inserted here >

Context 2: < Answer context 2 text inserted here >

Question: < Question inserted here >"

2. In-context Prompted LLMs (GPT3.5_turbo_in_context, GPT4_turbo_in_context)

In-context prompting, also known as *few-shot prompting*, involves providing the model with one or more examples of the task in the prompt itself. These examples serve as context for the model to understand the task and generate a response based on the provided examples. While generating the prompt is a little more complex in this case (it requires the collection and curation of relevant training examples), this method is designed to increase the robustness of the performance of the model. We have included an example of such a prompt in the Appendix.

To facilitate the prompting process of the AeroEngQA dataset using OpenAI's API, a Python script was developed. This script seamlessly integrates the OpenAI GPT-3.5 Turbo and GPT-4 Turbo models to generate responses to questions based on provided contexts. The script begins by initializing the OpenAI client with the appropriate API key and specifying the model to be utilized. Subsequently, it iterates through a list of JSON files containing contexts and questions. For each data entry, the script constructs prompts for both zero-shot and in-context scenarios, incorporating the provided context and question. These prompts are then used to solicit responses from the GPT model via the OpenAI API. The obtained responses are systematically stored alongside the original data, enabling the automated generation of responses to questions.

In order to facilitate reproducibility of the analysis, the parameters utilized in the interaction with the OpenAI API are detailed in what follows. Two distinct GPT models, *gpt-3.5-turbo-0125* and *gpt-4-turbo*, were considered. The temperature value was set to 1 to control the randomness in the text generation process. Additionally, a maximum token limit of 256 was imposed to ensure manageable response lengths. For diverse outputs, a cumulative probability value of

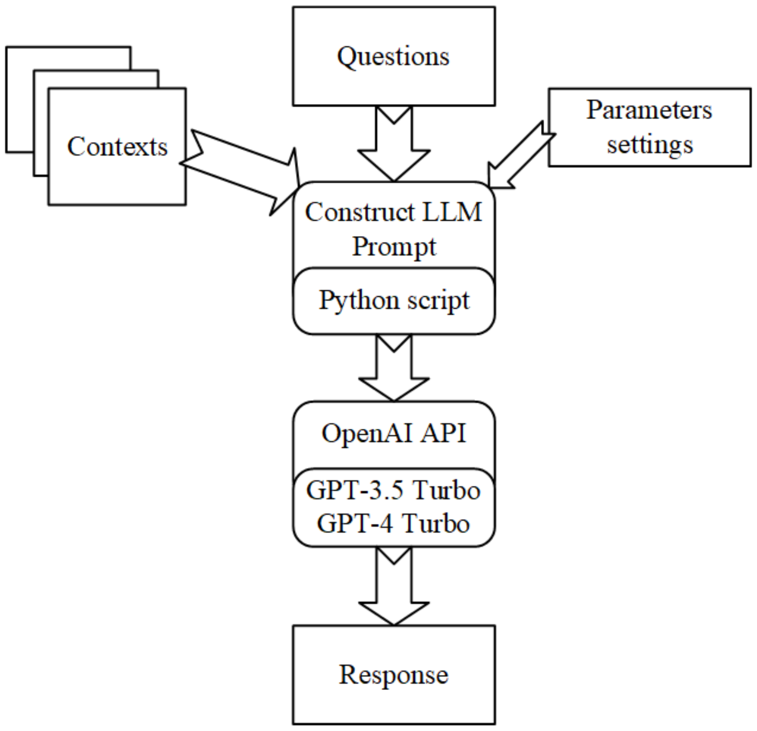

Fig. 1 In-Context Prompting Architecture with OpenAI

1 was employed, meaning that the entire probability distribution is considered and thereby ensuring maximum diversity in the generated responses.

To ensure coherence within the generated responses, both frequency penalty and presence penalty were deliberately set to 0. This approach effectively eradicated penalties for token repetition, thereby motivating the models to generate captivating content. These parameter configurations play a crucial role in preserving the consistency and reproducibility of the analysis, thereby facilitating precise result replication.

Figure 1 presents an overview of the in-context prompting process with OpenAI, highlighting the integration of GPT-3.5 Turbo and GPT-4 Turbo models. Central to this process is a Python script that manages the interaction with the OpenAI API, facilitating communication with the GPT models to generate responses to questions. These questions are sourced from the AeroEngQA dataset, comprising aerospace engineering contexts and queries. The generated responses are systematically stored alongside the original dataset, streamlining result replication for analytical purposes.

3. Retrieval Augmented Generation (RAG_zero_shot; RAG_in_context)

Retrieval-Augmented Generation (RAG)[21] bridges the gap between large information repositories and the generative mechanism of language models. RAG operates by taking an input query and retrieving a set of relevant documents from a source, such as a database or web pages. These documents are then concatenated with the original input to provide a rich context for the LLM text generator, which in turn produces the final output. The advantage of RAG lies in its adaptability; it allows LLMs to access the latest focussed information without the need for retraining, thus ensuring that the generated outputs remain reliable and up-to-date.

In these experiments, an open source Python framework for building custom apps with large language models, called Haystack was used[33]. Haystack has components useful for building a RAG system. The system diagram of how the Haystack elements were constructed is shown in figure 2. A Python script was prepared utilising the Haystack API to operate a Naïve Retrieval Augmented Generation System. A Haystack pipeline was prepared to process the QA dataset.

The test dataset contexts for all the questions were sent to a Haystack document store. For this we used a document joiner to concatenate them, then after cleaning, a document splitter to ensure the retrieved content size did not exceed the context window size. The splitter parameters were set to 250 token slices with a 50 token overlap between the slices. Split documents were then embedded and written to the store. To simplify matters we used an in-memory store,

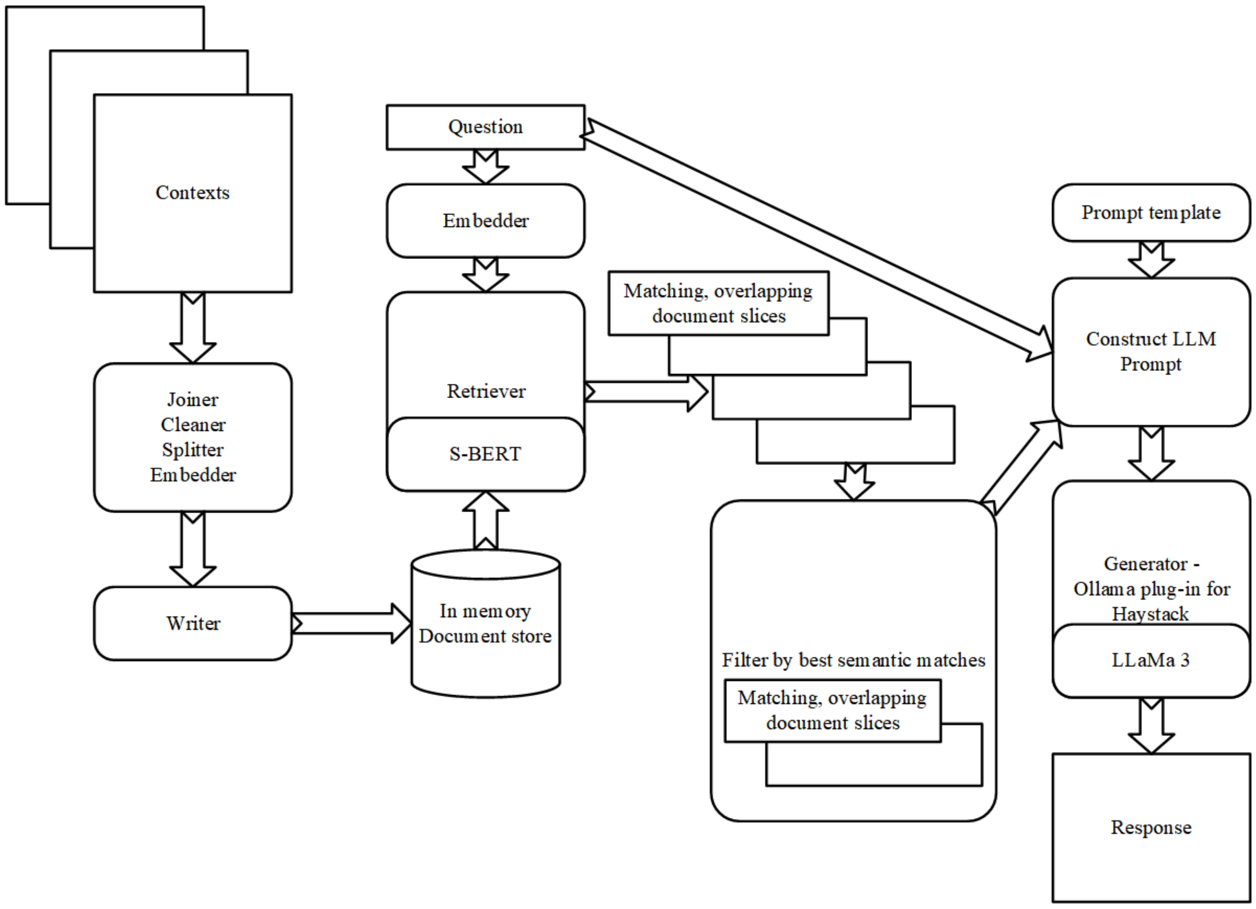

Fig. 2 RAG system using Haystack components

recreating it for each of the eight experiments, as the sizes of the documents were sufficiently small.

In order to find relevant content in the document store, each question in the test dataset was embedded and compared with the store content. Sentence-BERT[34] was used as the scoring mechanism, with the transformer model *all-MiniLM-L6-v2* used to obtain a semantic match measure of cosine similarity between the store and question feature vectors. For each question, a set of ten best matching document slices were retrieved in order of semantic similarity. This list was then filtered so that the score of every member of the set was at least 60% of the best semantic match. This resulted in sets that varied in size from 1 to 10 members, depending on the question; only selecting the best matching store content.

At this point in the pipeline we have a question, and a context of matching content. The next pipeline step was to create a prompt using the Jinja2 prompt templating language. Haystack has a library of prompt templates for different purposes, but we used the same prompts as the Zero-shot and In-context experiments for comparison purposes.

The LLM we used was Llama3 70B Instruct[35], operated through the Ollama[36] API acting as a wrapper. The context window size for Llama3 was set in Ollama to 8192 tokens. Ollama defaults to 4 bit quantization, and this was retained in order to fit the model into an available GPU. This would have introduced a small amount of LLM performance degradation. We used Ollama to set the LLM parameters, namely "top_p": 0.99, "top_k": 100, "temperature":0, "num_ctx": 8192

The first run results were collected in json and Excel files. Answers were manually marked with the other results. The individually scored responses were converted into percentage correct scores for each sub-experiment. These scores are shown in Table 2.

A second run through the test dataset was performed, this time stopping the script at incorrectly answered questions. Checks were made to find if IR had failed to find the pertinent section with the answer, or if IR had retrieved multiple answers. In such cases, these incidents were marked in the result files.

IV. Results

A. Setup

We performed the experiments described in the paper on an NVIDIA RTX A6000 graphics card (GPU memory 48GB). The inference runtime for RAG LLaMa3 70 was 42 minutes. The LLMs used were GPT3.5_turbo_zeroshot, GPT4_turbo_zeroshot, LLaMa3_zeroshot, RAG_zeroshot, GPT3.5_turbo_in_context, GPT4_turbo_in_context, LLaMa3_in_context, RAG_in_context. Hyperparameters for each of these models are described in Section III.

B. Qualitative Evaluation

The answer coding was performed manually by our team of human annotators. All answers generated by our LLMs were reviewed and scored using a coding schema at several annotation team meetings, with subjective or contentious answers debated and a consensus opinion formed for how to code them. The coding schema used was based on the metrics of answer accuracy and answer simplicity.

For answer accuracy enumerated types of *incorrect*, *partially correct* and *correct* were used. *Partially correct* included answers where some but not all of the gold answer was provided, and cases where the gold answer was augmented by additional answer text that made it confusing for the reader. An example of a *Partially correct* answer is *'Boeing 757 and Airbus A320'* where both options are viable answers but only one of these options is correct and the LLM has not chosen which one clearly and as such the answer is confusing.

For answer simplicity enumerated types of *simple* and *verbose* were used. If the answer was much longer than the gold answer then it was marked as *verbose*. It should be noted that the label *simple* and *verbose* was decided irrespective of if the answer was correct or not. An example of a verbose answer is *'Boeing 757. The Boeing 757 is an American narrow-body airliner designed and built by Boeing Commercial Airplanes. The then-named 7N7, a twinjet successor for the trijet 727, received its first orders in August 1978. The prototype completed its maiden flight on February 19, 1982, and it was FAA certified on December 21, 1982'*.

In this draft paper we are including a preliminary set of results, in the form of a set of bar charts (Figure 3 - 6) relating to the experiments conducted on questions based on two contexts, half answerable, the other half unanswerable questions.

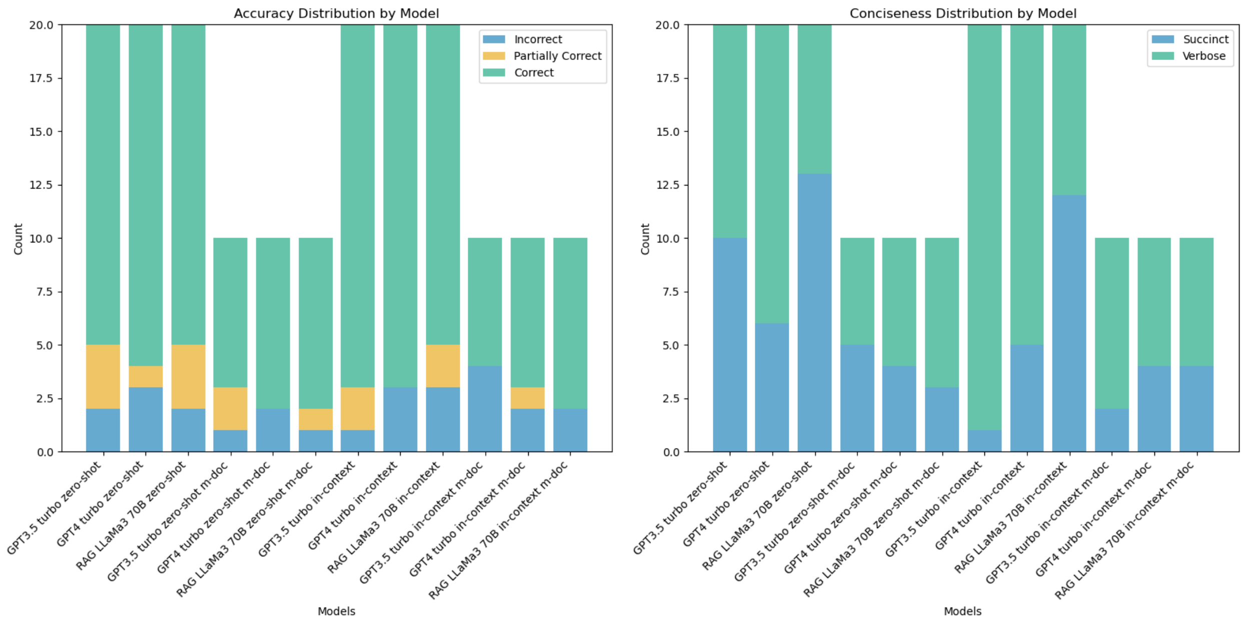

Fig. 3 Accuracy and conciseness of answers on the single-context, answerable questions from AeroEngQA.

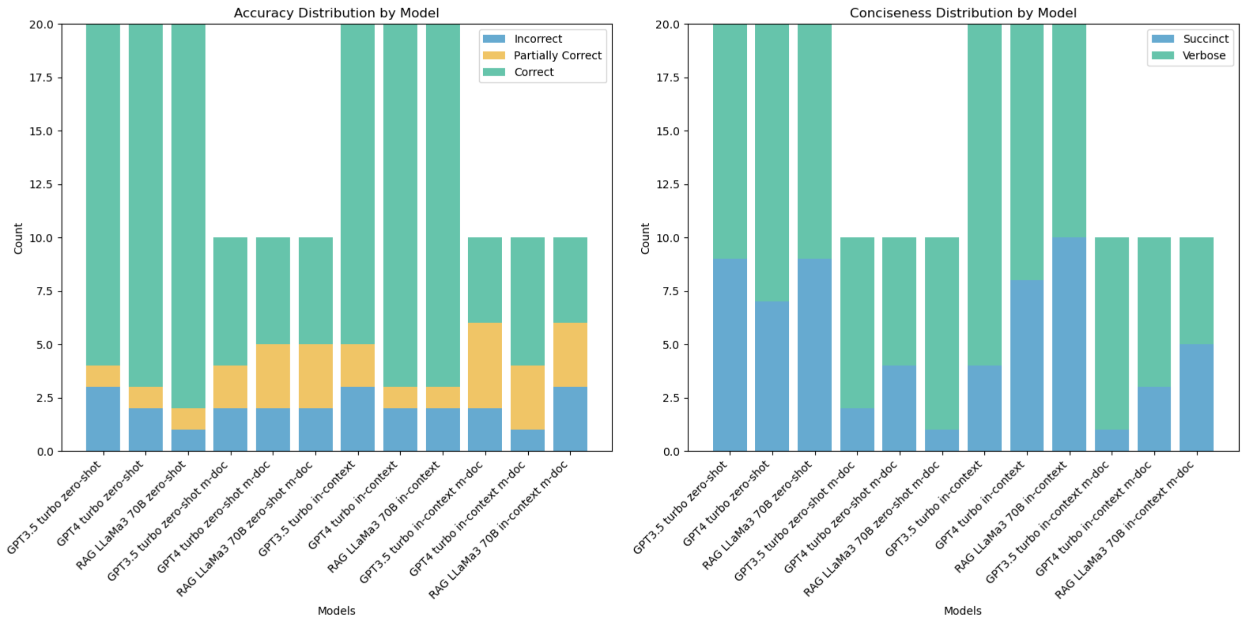

Fig. 4 Accuracy and conciseness of answers on the two-context, answerable questions from AeroEngQA.

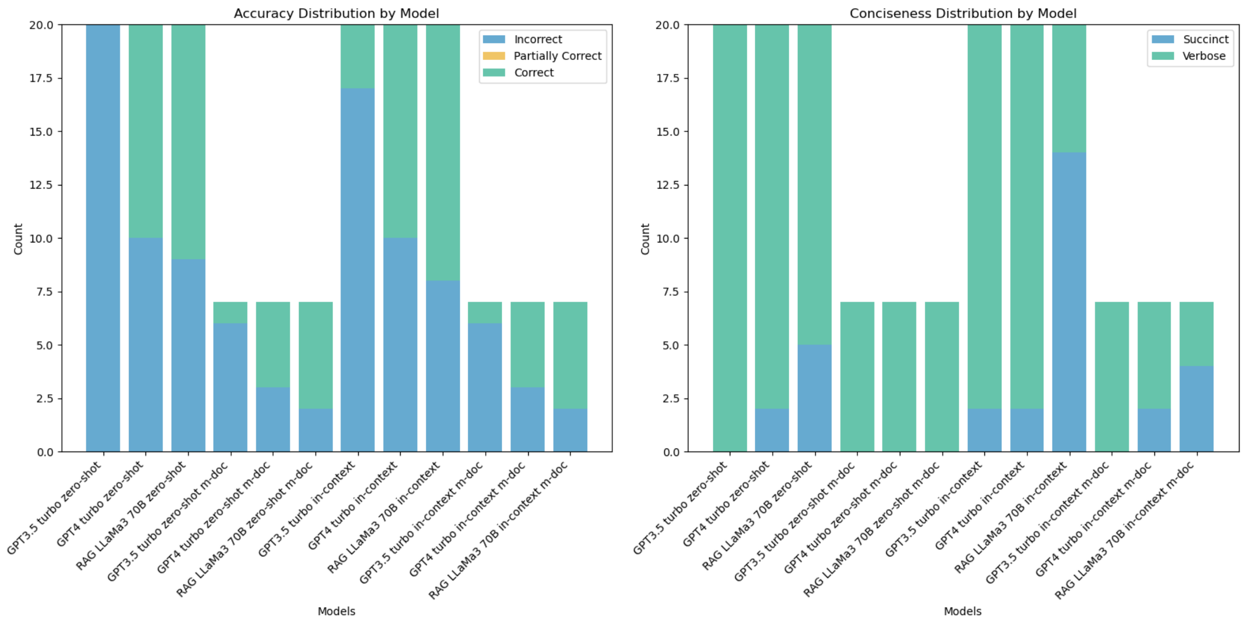

Fig. 5 Accuracy and conciseness of answers on the single-context, unanswerable questions from AeroEngQA.

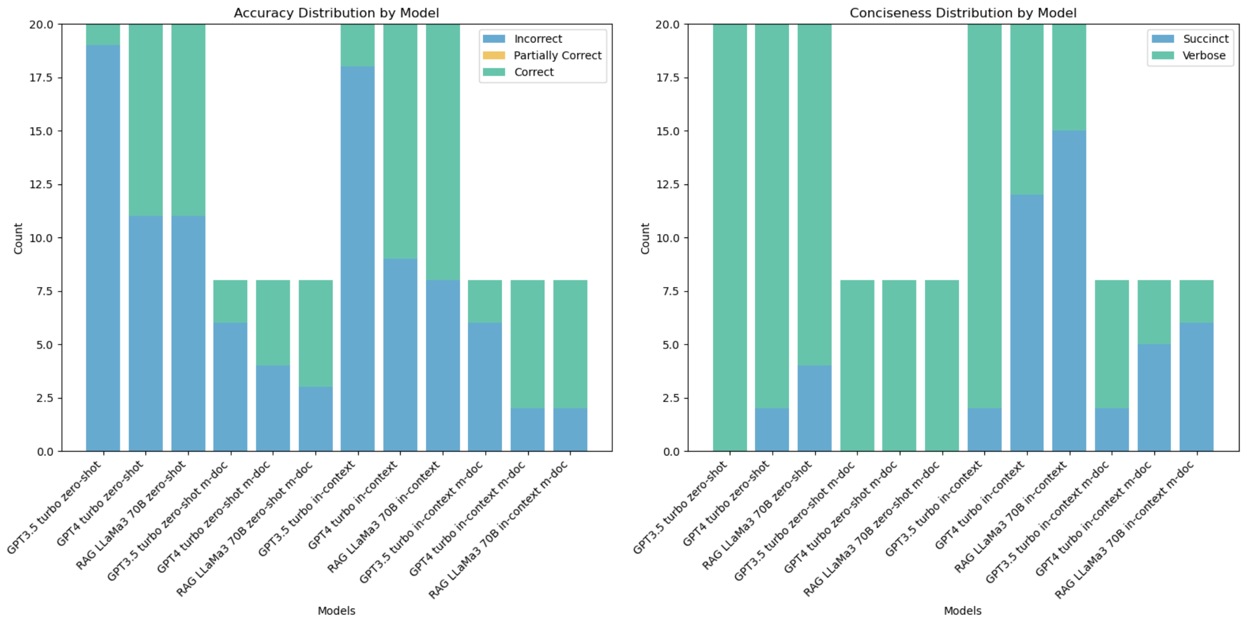

Fig. 6 Accuracy and conciseness of answers on the two-context, unanswerable questions from AeroEngQA.

V. Conclusions

The results from the answerable and unanswerable questions experiments in section IV provide some interesting insights into how well LLMs are able to support authentic engineering-focused QA tasks. In the context of the small dataset size of AeroEngQA, these conclusions should be taken as indicative results only that, whilst interesting, require additional larger scale experimentation to fully confirm.

In terms of accuracy of answers generated, prompting LLMs with examples via in-context prompting seems to improve the overall accuracy by around 5 to 10 percent on average (in particular on the unanswerable questions). As has been reported by other LLM research outside the engineering domain, LLM size seems to improve performance and we see GPT4 delivering more accurate answers than the smaller LLMs.

Our results are similar between single and multi-hop reasoning QA pairs, with only a small reduction in accuracy when trying to generate the more difficult multi-hop answers. This suggests that reasoning difficulty might not be the bottleneck for LLM performance for engineering QA tasks. When constructing the AeroEngQA dataset our human experts found that a question's technical ambiguity played an important part in how difficult that question was to answer, which suggests this is something that should be explored in more detail in future larger experiments.

Interestingly, the LLaMa3 70B RAG model accuracy was comparable with the non-RAG models, even though the RAG task is a more difficult one as the answer context must first be discovered. For multi-hop the RAG method actually out-performed the non-RAG models. This suggests that the RAG approach is well suited to working with local specialized technical corpora and thus represents a promising direction for engineering-focused LLM research.

With regards answer simplicity, all class of LLM tended to provide verbose answers with a substantial amount of unnecessary text. This property of LLMs has been reported by other researchers and could be a result of the reinforcement learning LLMs experience during pre-training as chatbots, which rewards more conversational responses. We found the RAG models slightly less verbose, but if a concise LLM is required then fine tuning methods such as instruction tuning or p-tuning will likely be required to change this pre-trained behaviour.

Comparing results for answerable and unanswerable questions, we found the LLMs had difficulty not answering a question and often tried to hallucinate an answer from the context that was close but not actually correct. There has been prior LLM research exploring the confirmation bias properties of LLMs and we hypothesise our results are a result of this tendency. Again, fine-tuning of LLMs will be needed to re-train models to avoid this type of pre-trained behaviour.

Another observation we have made through these experiments is that LLMs were often found generating sensible answers to our 'unanswerable' questions, evidently by leveraging information sources other than the context(s) we provided (for example, documents they had seen in their training set). This phenomenon merits further attention and will be examined later alongside the related topic of provenance.

From the point of view engineering design applications, our preliminary experiments have shown promise (while highlighting a few pitfalls as well) in terms of how a company might extract information from an internal database of documents (contexts – such as technical reports, memos, meeting transcripts, certification documents, etc.) in a much more time-efficient manner than ordinary (or even contextual) search might allow. The interface, which could be accessed by all design engineers, would be a digital assistant capable of complex inference, as well as highly efficient information retrieval, leveraging open source LLMs, while keeping sensitive documents inside the company. In spite of extremely rapid advances in this field, challenges remain, for example in terms of extending the inference capabilities to complex equations and graphs contained in the context.

Acknowledgments

The DAWS 2 (Development of Advanced Wing Solutions 2) project is supported by the ATI Programme, a joint Government and industry investment to maintain and grow the UK's competitive position in civil aerospace design and manufacture. The programme, delivered through a partnership between the Aerospace Technology Institute (ATI), Department for Business, Energy & Industrial Strategy (BEIS) and Innovate UK, addresses technology, capability and supply chain challenges.

References

[1] Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., and Gao, J., "Large Language Models: A Survey," , 2024. URL `https://arxiv.org/abs/2402.06196`.

[2] Song, Y., Wang, T., Cai, P., Mondal, S. K., and Sahoo, J. P., "A Comprehensive Survey of Few-shot Learning: Evolution,

Applications, Challenges, and Opportunities," *ACM Comput. Surv.*, Vol. 55, No. 13s, 2023. https://doi.org/10.1145/3582688, URL https://doi.org/10.1145/3582688.

[3] Forrester, A., Sóbester, A., and Keane, A. J., *Engineering Design via Surrogate Modelling: A Practical Guide*, John Wiley and Sons, London, 2008.

[4] Zhao, H., Chen, H., Yang, F., Liu, N., Deng, H., Cai, H., Wang, S., Yin, D., and Du, M., "Explainability for Large Language Models: A Survey," *ACM Trans. Intell. Syst. Technol.*, Vol. 15, No. 2, 2024. https://doi.org/10.1145/3639372, URL https://doi.org/10.1145/3639372.

[5] National Cyber Security Centre, "ChatGPT and large language models: what's the risk? Do loose prompts sink ships? Exploring the cyber security issues of ChatGPT and LLMs," https://www.ncsc.gov.uk/blog-post/chatgpt-and-large-language-models-whats-the-risk, 2024. Accessed: 3 May 2024.

[6] Guerreiro, N. M., Alves, D. M., Waldendorf, J., Haddow, B., Birch, A., Colombo, P., and Martins, A. F. T., "Hallucinations in Large Multilingual Translation Models," *Transactions of the Association for Computational Linguistics*, Vol. 11, 2023, pp. 1500–1517. https://doi.org/10.1162/tacl_a_00615, URL https://doi.org/10.1162/tacl_a_00615.

[7] Li, J., Cheng, X., Zhao, X., Nie, J.-Y., and Wen, J.-R., "HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models," *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, edited by H. Bouamor, J. Pino, and K. Bali, Association for Computational Linguistics, Singapore, 2023, pp. 6449–6464. https://doi.org/10.18653/v1/2023.emnlp-main.397, URL https://aclanthology.org/2023.emnlp-main.397.

[8] Silva, E. A., Marsh, R., Yong, H. K., Middleton, S. E., and Sóbester, A., "Aircraft Engineering Specific, Human-Annotated Question and Answer Benchmarking Set," https://doi.org/10.5281/zenodo.14215677, 2024. Zenodo.

[9] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D., "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering," , 2018.

[10] Rajpurkar, P., Jia, R., and Liang, P., "Know What You Don't Know: Unanswerable Questions for SQuAD," *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, edited by I. Gurevych and Y. Miyao, Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 784–789. https://doi.org/10.18653/v1/P18-2124, URL https://aclanthology.org/P18-2124.

[11] Rajpurkar, P., Jia, R., and Liang, P., "Know What You Don't Know: Unanswerable Questions for SQuAD," *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, edited by I. Gurevych and Y. Miyao, Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 784–789. https://doi.org/10.18653/v1/P18-2124, URL https://aclanthology.org/P18-2124.

[12] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., and Manning, C. D., "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering," *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, edited by E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 2369–2380. https://doi.org/10.18653/v1/D18-1259, URL https://aclanthology.org/D18-1259.

[13] Castelli, V., Chakravarti, R., Dana, S., Ferritto, A., Florian, R., Franz, M., Garg, D., Khandelwal, D., McCarley, S., McCawley, M., Nasr, M., Pan, L., Pendus, C., Pitrelli, J., Pujar, S., Roukos, S., Sakrajda, A., Sil, A., Uceda-Sosa, R., Ward, T., and Zhang, R., "The TechQA Dataset," *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, edited by D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Association for Computational Linguistics, Online, 2020, pp. 1269–1278. https://doi.org/10.18653/v1/2020.acl-main.117, URL https://aclanthology.org/2020.acl-main.117.

[14] Chen, M., Chen, X., and Yih, W.-t., "Few-Shot Data Synthesis for Open Domain Multi-Hop Question Answering," *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, edited by Y. Graham and M. Purver, Association for Computational Linguistics, St. Julian's, Malta, 2024, pp. 190–208. URL https://aclanthology.org/2024.eacl-long.12.

[15] Agarwal, A., Gawade, S., Azad, A. P., and Bhattacharyya, P., "KITLM: Domain-Specific Knowledge InTegration into Language Models for Question Answering," , 2023.

[16] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *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)*, edited by J. Burstein, C. Doran, and T. Solorio, Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. https://doi.org/10.18653/v1/N19-1423, URL https://aclanthology.org/N19-1423.

[17] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R., "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

[18] Malladi, S., Gao, T., Nichani, E., Damian, A., Lee, J. D., Chen, D., and Arora, S., "Fine-Tuning Language Models with Just Forward Passes," *Advances in Neural Information Processing Systems*, Vol. 36, edited by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Curran Associates, Inc., 2023, pp. 53038–53075. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a627810151be4d13f907ac898ff7e948-Paper-Conference.pdf.

[19] Zhang, H., Gong, Y., Shen, Y., Li, W., Lv, J., Duan, N., and Chen, W., "Poolingformer: Long Document Modeling with Pooling Attention," *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 139, edited by M. Meila and T. Zhang, PMLR, 2021, pp. 12437–12446. URL https://proceedings.mlr.press/v139/zhang21h.html.

[20] Zhang, Q., Chen, S., Xu, D., Cao, Q., Chen, X., Cohn, T., and Fang, M., "A Survey for Efficient Open Domain Question Answering," *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, edited by A. Rogers, J. Boyd-Graber, and N. Okazaki, Association for Computational Linguistics, Toronto, Canada, 2023, pp. 14447–14465. https://doi.org/10.18653/v1/2023.acl-long.808, URL https://aclanthology.org/2023.acl-long.808.

[21] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," *Advances in Neural Information Processing Systems*, Vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Curran Associates, Inc., 2020, pp. 9459–9474. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

[22] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, Vol. 21, No. 1, 2020.

[23] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L., "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, edited by D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Association for Computational Linguistics, Online, 2020, pp. 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703, URL https://aclanthology.org/2020.acl-main.703.

[24] Song, E., Kim, S., Lee, H., Kim, J., and Thorne, J., "Re3val: Reinforced and Reranked Generative Retrieval," *Findings of the Association for Computational Linguistics: EACL 2024*, edited by Y. Graham and M. Purver, Association for Computational Linguistics, St. Julian's, Malta, 2024, pp. 393–409. URL https://aclanthology.org/2024.findings-eacl.27.

[25] Yu, W., Iter, D., Wang, S., Xu, Y., Ju, M., Sanyal, S., Zhu, C., Zeng, M., and Jiang, M., "Generate rather than Retrieve: Large Language Models are Strong Context Generators," *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=fB0hRu9GZUS.

[26] Tikayat Ray, A., Pinon Fischer, O. J., White, R. T., Cole, B. F., and Mavris, D. N., "Development of a Language Model for Named-Entity-Recognition in Aerospace Requirements," *Journal of Aerospace Information Systems*, Vol. 0, No. 0, 0, pp. 1–11. https://doi.org/10.2514/1.I011251, URL https://doi.org/10.2514/1.I011251.

[27] Silva, E. A. D. C., Marsh, R., Yong, H. K., and Sobester, A., "A Model Based Systems Engineering Framework based on a Visual Programming Paradigm," *AIAA SCITECH 2024 Forum*, 2024. https://doi.org/10.2514/6.2024-1529, URL https://arc.aiaa.org/doi/abs/10.2514/6.2024-1529.

[28] DePauw, T. C., Kabir, M., Stere, A., and Dong, J., "Development of a Commercial Airplane Certification Digital Assistant Using a Large Language Model Trained with Regulatory Requirements and Means of Compliance Documents." *AIAA SCITECH 2024 Forum*, 2024. https://doi.org/10.2514/6.2024-1528, URL https://arc.aiaa.org/doi/abs/10.2514/6.2024-1528.

[29] Connolly, B. J., and Anderson, K. M., "Harnessing Large Language Models for Satellite Ground Tests," *AIAA SCITECH 2024 Forum*, 2024. https://doi.org/10.2514/6.2024-0916, URL https://arc.aiaa.org/doi/abs/10.2514/6.2024-0916.

[30] "14 CFR Part 25 - Airworthiness Standards: Transport Category Airplanes," , 2024. URL https://www.ecfr.gov/current/title-14/chapter-I/subchapter-C/part-25?toc=1.

[31] Jin, H., Zhang, T., Ramamurthy, A., Hamza, A., and Malinoski, M., "Learning to Verify and Assure Cyber-Physical Systems," *AIAA SCITECH 2024 Forum*, 2024. https://doi.org/10.2514/6.2024-1853, URL https://arc.aiaa.org/doi/abs/10.2514/6.2024-1853.

[32] Shmilovich, A., Yadlin, Y., Vijgen, P. M., and Woszidlo, R., "Applications of Flow Control to Wing High-Lift Leading Edge Devices on a Commercial Aircraft," *AIAA SCITECH 2023 Forum*, 2023. https://doi.org/10.2514/6.2023-0656, URL https://arc.aiaa.org/doi/abs/10.2514/6.2023-0656.

[33] Pietsch, M., Möller, T., Kostic, B., Risch, J., Pippi, M., Jobanputra, M., Zanzottera, S., Cerza, S., Blagojevic, V., Stadelmann, T., Soni, T., and Lee, S., "Haystack: the end-to-end NLP framework for pragmatic builders," , November 2019. URL https://github.com/deepset-ai/haystack.

[34] Reimers, N., and Gurevych, I., "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," , 2019.

[35] AI@Meta, "Llama 3 Model Card," *GitHub*, 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[36] Ollama, "Ollama repo," *GitHub*, accessed: 2 May 2024. URL https://github.com/ollama.

VI. Appendix

A. Example of a Prompt for In-Context Prompting with Multiple Contexts

QUERY:

Generate your response to the forthcoming question solely based on the provided context, refraining from external knowledge. Employ serialized data in JSON format, with the field name "answer". This prompt comprises two contextual passages and a question. Your answer should be concise and directly derived from the context provided.

Answerable example:

Context 1: In 1996, the NTSB investigated the crash of TWA Flight 800, revealing critical flaws in fuel tank system design.

Context 2: Following the NTSB’s investigation, regulatory changes were implemented to enhance fuel tank system safety in commercial aircraft.

Question: How did the NTSB’s investigation into the crash of TWA Flight 800 influence the aviation industry’s safety standards?

Answer: The investigation prompted regulatory changes to improve fuel tank system safety, shaping industry standards.

Non-Answerable example:

Context 1: In 2019, the NTSB conducted an investigation into an accident involving a Boeing B-17G Flying Fortress operated by the Collings Foundation. The incident raised concerns about safety protocols in for-hire Part 91 operations, particularly those involving paying passengers.

Context 2: The NTSB’s findings emphasized the need for improved safety training, maintenance procedures, and operational oversight within for-hire Part 91 operations.

Question: What specific safety protocols did the NTSB identify as needing improvement in for-hire Part 91 operations following the Boeing B-17G Flying Fortress accident in 2019?

Answer: Unanswerable.

Context 1: In order to estimate the stability and performance of the aircraft, two computational tools were used. XFLR5 (Ref. 14), a vortex lattice method analysis tool that is inherently coupled with XFOIL (Ref. 15), provided estimates of the lift, static stability, induced drag, and profile drag with a lifting-surface-only geometry, as shown in Figure 9. To estimate the parasitic drag of the aircraft, a component drag buildup method was implemented in a spreadsheet. The drag of each component (e.g., fuselage, main wing, etc.) was first estimated based on the skin friction drag of a flat plate with the same characteristic length as the component. Then this flat plate drag estimate was modifed to approximate the drag of the entire three-dimensional component with a form factor (Ref. 16) based on the component type (e.g., wing, body) and the wetted area of the component, which was estimated with OpenVSP. Finally, interference drag and excrescence drag were accounted for by multiplying the resulting drag of components by correction factors, which were based on general guidelines (Ref. 17) and previous analyses.

Context 2: The forward flight performance of the aircraft was estimated with the calculated lift and drag values for the aircraft resulting from the XFLR5 and parasite drag analyses. This analysis excludes the impacts of the thrust and slipstreams from the wingtip propellers. Because the center of the wingtip propellers are above the CG, there will be a nosedown pitching moment generated in forward flight, which must be offset for the aircraft to trim. This can be accomplished by adjusting control surfaces, which will increase the drag compared to the predicted value. However, the wingtip propellers should reduce the induced drag. For these initial performance calculations, these competing effects were assumed to negate one another, even though this will not likely be the case in practice. By analyzing different angles of attack, conditions where the lift was equal to the weight were found, and the approximate lift to drag ratio, L/D, was determined over a range of airspeeds. The forward flight L/D values predicted for a range of airspeeds are shown in Figure 10, and the peak cruise L/D was observed at approximately 90 ft/s (53 knots).

Question: Does the vortex lattice method account for the impact of the slipstream from the wingtip propellers?