The Framework and Implementation of Using Large Language Models to Answer Questions About Building Codes and Standards

Isaac Joffe[1], George Felobes[2], Youssef Elgouhari[3], Mohammad Talebi Kalaleh[4], Qipei Mei, Ph.D., P.Eng.[5], and Ying Hei Chui, Ph.D., P.Eng.[6]

[1]Undergraduate Research Assistant, University of Alberta, Edmonton, Canada T6G 2R3. Email: ijoffe@ualberta.ca

[2]Research Assistant, University of Alberta, Edmonton, Canada T6G 2R3. Email: felobes@ualberta.ca

[3]Research Assistant, University of Alberta, Edmonton, Canada T6G 2R3. Email: algohary@ualberta.ca

[4]Ph.D. Student, University of Alberta, Edmonton, Canada T6G 2R3. Email: talebika@ualberta.ca

[5]Assistant Professor, University of Alberta, Edmonton, Canada T6G 2R3. Email: qipei.mei@ualberta.ca

[6]Professor, University of Alberta, Edmonton, Canada T6G 2R3. Email: yhc@ualberta.ca

ABSTRACT

Civil and structural engineering design projects are subject to strict regulations of relevant codes and standards to guarantee that certain standards of safety, reliability, and efficiency are met. However, ensuring that all engineering designs comply with the precise provisions of pertinent civil and structural engineering codes and standards is a complex and time-consuming task currently completed by professional engineers. Recent advancements in artificial intelligence have enabled large language models (LLMs) to complete abstract and complex tasks, such as answering questions based on provided context and summarizing text passages, with high accuracy. This work presents a novel framework to build an open-source and scalable LLM-based application allowing engineers

to quickly receive accurate answers to their codes-and-standards-related questions alongside corresponding citations simply by interacting in natural language with a ChatGPT-style chatbot. This work also presents a preliminary implementation of this framework using the National Building Code of Canada 2020. The system implemented achieves promising results, indicating that the proposed framework may be a useful tool to assist design engineers in efficiently and effectively completing their work and that this approach holds promise for applications to other domains.

PRACTICAL APPLICATIONS

New and powerful large language models (LLMs), such as ChatGPT, are rapidly changing the way in which professional work is done. However, these state-of-the-art LLMs alone lack the accuracy needed for truthfully answering detailed technical questions, limiting their utility in engineering contexts. Civil engineers often have to manually check designs for compliance with codes and standards, but this task can be difficult and time-consuming, especially for junior engineers. This paper investigates the use of modern LLMs alongside information retrieval techniques to enable engineers to receive accurate answers to their plain language questions about a particular code or standard, as well as a reference to the specific sections in the code or standard that support the generated responses. To achieve this, a general framework and an example system are introduced to produce a question-answering chatbot for these questions. Experiments on example questions show promising results, indicating that this technology has the potential to provide an efficient way for engineers to interact with codes and standards.

INTRODUCTION

Problem Background

Civil and structural engineering projects of all scales and scopes are strictly regulated to guarantee adherence to minimum standards of safety, reliability, and efficiency. These regulations take the form of building codes and standards. Building codes are typically extensive documents housing an abundance of precise provisions that outline the minimum requirements to be satisfied by all associated projects, while standards contain specific technical information describing the process

of fulfilling the requirements laid out by relevant codes. Application of these documents usually involves a professional engineer meticulously poring over large volumes of text to find minute but critical details affecting the acceptability of the design. Often, the interdisciplinary nature of engineering projects necessitates the coordinated application of several different codes and standards with unrelated scopes simultaneously (Ching and Winkel 2021). Despite the paramount importance of regulatory compliance, the aforementioned factors make this a slow and tedious process that consumes limited and expensive engineering hours, diverting resources that may be more judiciously allocated to other more complex and involved tasks.

Related Work

The predominantly textual nature of codes and standards has inspired previous research into the application of classical natural language processing (NLP) technologies to solving these problems (Fuchs 2021). Zhang and El-Gohary (2016) developed an NLP-based approach to automatically extract regulatory requirements from building codes. Their method used semantic role labeling and syntactic parsing to identify subject-predicate-object triples representing requirements. The extracted rules were represented in a logic-based format. While effective, the approach was limited to processing simple sentence structures. Beach et al. (2015) proposed a semi-automated method combining NLP and manual review to extract regulatory information from building codes. They used part-of-speech tagging and dependency parsing to identify key phrases, which were then manually reviewed and formalized. This hybrid approach improved accuracy but still required significant manual effort. Jiang et al. (2019) proposed an ontology-based framework for extracting regulatory requirements from building codes. They used NLP to identify key terms and relations, which were then mapped to concepts in a domain ontology. This enabled semantic querying of requirements, but the ontology development was time-consuming. Xu et al. (2019) developed a method to automatically construct a regulatory knowledge graph from building codes using NLP and machine learning. Their approach extracted entities and relations to populate the graph. While promising for knowledge representation, the method had limitations in capturing complex regulatory logic. Nawari (2019) developed an NLP-based system for automated code compliance checking.

The system used NLP to extract rules from building codes and translate them into a computer-processable format. While promising, it was limited to specific types of requirements and required manual review. Zhong et al. (2020) proposed a transformer-based model for retrieving relevant clauses from building codes given a query. Their approach outperformed traditional information retrieval methods but was limited to retrieval without interpretation of regulatory content. In general, in spite of the great progress, such classical computational NLP-based solutions have been limited by their accuracy, scalability, and level of abstraction. Furthermore, these works have did not produce an accessible user interface optimized for real-world use.

Recent advancements in artificial intelligence (AI) technologies have enabled computers to complete complex and abstract NLP tasks via large language models (LLMs) with ever-increasing proficiency (Zhao et al. 2023; Shanahan 2023). LLMs are modern systems that utilize deep learning, a technology based on the artificial neural network dating back to the 1950s (Wason 2018), alongside contemporary techniques, including the use of enormous datasets and immensely powerful computing hardware. The most important breakthrough in this field was the advent of the self-attention-mechanism-based Transformer architecture (Vaswani et al. 2017). This provided the foundation for the development of powerful LLMs capable of modeling human language at unprecedented levels (Islam et al. 2023). Subsequent works expanded upon this architecture, introducing the technique of unsupervised pre-training to facilitate the development of versatile foundational models such as BERT (Devlin et al. 2019) and GPT (Radford et al. 2018a). The rapid proliferation of such technologies has led to a relentless onslaught of new LLMs at a remarkable pace, with the amount of, the average size of, and both the academic and public interest in LLMs continually increasing at an exponential rate (Zhao et al. 2023; Naveed et al. 2023). Early landmark models such as GPT-2 (Radford et al. 2018b) and GPT-3 (Brown et al. 2020) furthered the state-of-the-art by exhibiting strong performance on a multitude of abstract tasks and producing text nearly indistinguishable from human composition, respectively. Chinchilla (Hoffmann et al. 2022) highlighted the importance of corpora size, not just LLM size, in optimizing LLM performance while balancing the consumption of computing resources. Other improvements have led to the

development of current state-of-the-art models, such as Google's LaMDA (Thoppilan et al. 2022), PaLM (Chowdhery et al. 2022), and PaLM-2 (Anil et al. 2023); Meta's LLaMA (Touvron et al. 2023b) and LLaMA-2 (Touvron et al. 2023a), and, perhaps most famously, OpenAI's GPT-4 (OpenAI 2023). The exceptional performance of these LLMs, which vastly outperform traditional methods (Min et al. 2023), has thoroughly captured the public's attention and permeated many aspects of daily life, sparking interest in their potential applications to industry-based, real-world problems in advanced fields like civil and structural engineering (Bommasani et al. 2022).

While recent models have exhibited remarkable performance on a variety of general-purpose language understanding and generation tasks, the vast pre-training corpora utilized and the inherent probabilistic nature of such LLMs has prevented them from being naturally adept at accurately answering questions, especially domain-specific questions with precise answers (Yang et al. 2023a; Krishna et al. 2021). Often, even state-of-the-art LLMs are poorly calibrated for question-answering (Jiang et al. 2021), meaning that the model's confidence in its response has little correlation with that response's correctness. This is a general limitation of LLMs (McKenna et al. 2023) related to the hallucination phenomenon, where LLMs confabulate factually incorrect responses that often seem plausible to unknowing users. This behavior poses a significant challenge to LLM use in civil and structural engineering, where the truthfulness of generated text holds critical significance for high-stakes, real-world construction projects. Furthermore, while existing LLMs have demonstrated some domain-specific knowledge, their generalized nature sacrifices specific knowledge in specialized fields, rendering them unable to accurately answer in-depth queries. Despite these challenges, the swift pace of improvement in modern NLP technologies indicates a strong potential for powerful applications across diverse fields (Bommasani et al. 2022), including civil and structural engineering. For example, recent enhancements to question-answering systems combining an information retrieval module with an LLM for context-aware text generation (Chen et al. 2017; Lee et al. 2019; Karpukhin et al. 2020; Chen et al. 2023b) have mitigated the hallucination problem; incorporating context into prompts reduces hallucination by as much as 50 times (Feldman et al. 2023). Datasets have also emerged to evaluate such systems (Zhuang et al. 2023).

Little exploration into the applications of modern NLP technologies to the realm of civil and structural engineering has occurred. Preliminary investigations, marred by the general limitations of language modeling including hallucination and bias (Head et al. 2023), have generally focused on the use of off-the-shelf LLMs as-is (Aluga 2023). Recent works have begun to use LLMs to interact with building codes and standards, but not in a conversational manner (Zheng et al. 2023). However, the rapid proliferation of powerful LLMs and their seemingly endless possibilities has brought about applications of these technologies to countless other disciplines (Bommasani et al. 2022); for example, recent works have explored applying LLMs to finance (Yang et al. 2023b), law (Douka et al. 2022), and medicine (Wu et al. 2023; Singhal et al. 2023; Guo et al. 2022). These systems almost universally apply fine-tuning techniques with vast domain-specific corpora alongside prompt engineering strategies to contemporary foundational LLMs, achieving promising results even beginning to rival human performance in medicine (Singhal et al. 2023). Other works have explored the related issue of using LLMs to understand and reason over highly structured datasets (Jiang et al. 2023). In the future, similar methods may be utilized here to understand the large tables and databases included in many engineering codes and standards, but textual data remains the scope of the present work. Numerous studies have considered the hallucination problem plaguing LLMs by building systems to better incorporate information sources into model responses. Some approaches employ retrieval-augmented generation to naturally attribute sources (Guu et al. 2020; Ram et al. 2023), like in the present work, while others embed references seamlessly in generated text (Menick et al. 2022).

Research Objectives and Contributions

The objective of this work is to design an automated framework for the construction of a custom LLM-based system enabling civil and structural engineers to quickly receive accurate answers to their questions concerning a particular engineering code or standard simply by interacting in natural language with a conversational chatbot. To ensure the tool is trustworthy enough for use in industrial settings with real-world repercussions, correct citations of the information sources used must be provided alongside all responses generated by the system. This feature will allow engineers

to determine and further investigate the source of any information used to generate a given response. This system must encapsulate a complete framework allowing any code or standard supplied to be converted into an interactive chatbot. Utilizing LLMs to interact with codes and standards in this innovative fashion will expedite this portion of the design process, ultimately improving overall workflow efficiency and enhancing engineering productivity.

Success of the system depends not only on the accuracy and speed of response generation, but also on the scalability, democratization, portability, and robustness of the system. Specifically, to ensure the system is scalable – meaning that it can be applied to many codes and standards, that it can be applied to very large codes and standards, that it can be used by many people concurrently, *etc.* – the entire pipeline must be entirely open-source. All LLMs and other tools utilized must be free, even for commercial use. Similarly, to limit the often prohibitively large amount of computing resources required to effectively run such technologies on consumer hardware, all LLMs employed should be relatively small. These factors will also facilitate democratization of the technology, allowing small and medium-sized enterprises to leverage this tool. Because countless engineering codes and standards, ranging from enormous building codes to small client-created documents, may be employed on a single project, framework portability and robustness are key; the system must be capable of strong performance across vastly different codes and standards.

This paper presents the general framework to translate any code or standard into an interactive chatbot system and shares an implementation of such a system based on the National Building Code of Canada 2020 (NBCC) (Canadian Commission on Building and Fire Codes 2022) as a proof-of-concept. The main contribution of this paper is the application and adaptation of modern LLM-based NLP techniques specifically to answering questions about codes and standards. This high-stakes domain presents many important challenges, including a critical need for precision and accuracy and an emphasis on explainability and traceability. Unlike previous works (Zhang and El-Gohary 2016; Beach et al. 2015; Nawari 2019; Jiang et al. 2019; Xu et al. 2019; Zhong et al. 2020), our approach does not require any additional manual work from the engineer at runtime. Additionally, the proposed system leverages LLMs to interpret the retrieved information, producing

easily-readable answers to user questions that may be expressed freely in natural language. The use of LLMs brings about many advantages over traditional NLP techniques, primarily related to performance (Min et al. 2023). This work combines advanced information retrieval techniques with the recent computing method of LLM-based text generation to create a more complete system to assist engineers in the application of codes and standards acting as a novel ChatGPT-style chatbot. Furthermore, another main contribution of this paper is demonstrating the feasibility of applying LLMs to lengthy documents with technical and professional content, such as engineering codes and standards. Overall, the framework presented in this paper uniquely seeks to create a user-friendly experience, allowing engineers to effortlessly interact with these complex technical documents in plain language.

Subsequent sections of this paper are organized as follows:

- Section 3 outlines the methodology of the general framework designed,
- Section 4 describes the methodology of the specific system implemented,
- Section 5 presents the empirical results of this system with accompanying analysis, and
- Section 6 summarizes the key takeaways of this work and provides direction for future research.

GENERAL FRAMEWORK

Overview

The general architecture utilized by the system is a retrieval-augmented generation (RAG) pipeline. In this robust and secure approach, user inputs are used to search a knowledge database encompassing all information present in the code or standard of interest. Relevant context information is retrieved with accompanying references and embedded into the user input prior to generating natural language responses. By informing system answers with the actual source information from the code or standard, accuracy is improved and practical citations are guaranteed. Incorporating search enables accurate answers relaying precise information in a manner that, at present, the use of LLMs alone cannot provide. Because of the capability for generalization of advanced search

algorithms and LLMs, this methodology can be applied to any corpora housing technical and precise information and, thus, holds promise for diverse scientific domains and industrial applications (Gao et al. 2024).

This design requires the development of a framework composed of two distinct parts (Fig. 1). First, the code or standard of interest is parsed and transformed into searchable knowledge databases (the "data engineering pipeline"). Second, these knowledge databases are integrated with an LLM to produce a natural language interface (the "chatbot application"). The former part executes only once per code or standard, but the latter segment executes repeatedly at application runtime as it encompasses the dynamic behavior of the system.

Data Engineering Pipeline

The first stage in the framework involves converting the code or standard at hand from its original, unprocessed format into a structured, predictable format that may be effectively searched while maintaining data privacy and integrity. A step-by-step data engineering pipeline completes this task (Fig. 2).

First, the relevant code or standard is subject to preliminary preprocessing algorithms and various cleaning procedures producing a uniform and coherent corpus. The details and extent of this task vary widely, depending wholly on the characteristics and quirks of the original data format, including the file format, organizational hierarchy, and more. More structured file formats, such as XML, HTML, or TeX, are best because they naturally organize information and integrate valuable metadata, but codes and standards are typically distributed in visually-based file formats, like PDF or DOCX. These formats introduce an array of data quality issues – like sporadic insertion or deletion of whitespace characters, hidden text, improper line formatting, and improper figure and table formatting – that are largely resolved by custom file-parsing algorithms and regular expressions. More general cleaning procedures, particularly those related to file formatting, may be universally applied to ensure some scalability of this process to a wide array of codes and standards, but certain document-specific corrections for unique issues may still be required. This is especially the case for PDF files; other formats obtained directly from the document publisher,

such as XML or HTML, simplify this preprocessing stage and improve scalability. Additionally, because LLMs exclusively analyze textual data, only raw text is extracted; figures and tables are ignored. These procedures must be carefully designed so that all document content is retained, ensuring the comprehensiveness of the extracted data. This stage produces a cleaned, purely textual file containing all the information present in the code or standard at hand. Given data in this preprocessed format, the framework is fully scalable to any document; the scope of this paper is to prove that constructing a complete pipeline is possible, even though this process may rely on some document-specific preprocessing algorithms.

Second, this refined textual data is partitioned into a mutually exclusive but collectively exhaustive set of small "documents," each representing a small fraction of the information present in the code or standard of interest. Each document should thoroughly embody exactly one idea (in this case, a provision). Documents containing an incomplete idea pose problems once retrieved; such a document may not contain all the information needed to comprehensively answer a relevant question when retrieved, and the other documents containing this missing information may contain other, unrelated ideas, preventing them from being retrieved concurrently. However, documents encapsulating multiple ideas are also problematic; unnecessary information will be presented to the LLM, delaying and potentially distracting generated responses, and the additional sentiments present in the document may prevent it from being retrieved at all. Therefore, for best results, the content of the code or standard should be split naturally around the original document structure; that is, on the basis of sections and subsections, clauses and subclauses, paragraphs and sentences, and the like. Structured file formats are advantageous again because they automatically encode this information. Pertinent metadata may be extracted from unstructured formats through the use of regular expressions, and content may be split by a simple word or sentence count rule if such metadata is unavailable. In addition to the content of each document, a citation localizing it is stored for accurate referencing once retrieved. These may be constructed from certain metadata – headers, footers, page numbers, section numbers, *etc.* – naturally present in structured file formats or extracted from unstructured file formats. Once utilized, these metadata are removed from

the document content as they carry little useful meaning. This stage creates a secure database of compact but information-dense documents, each dedicated to one particular topic, alongside corresponding citations. This text parsing and cleaning task is extremely important to system success.

Third, this preliminary document collection is further preprocessed to enable and optimize the search algorithms employed. Two main classes of search algorithm, lexical and semantic, are utilized. The former is a keyword-based search that returns those documents with the most words in common with some input, while the latter is an embeddings-based search that retrieves those documents with the most similar vectorized mapping to some input. To optimize the lexical search, all raw text across the database is made uniform; this involves decapitalization, the removal of certain punctuation (such as periods, apostrophes, and quotation marks), and the removal of common stopwords carrying little or no actual meaning (such as "the" and "a"). This process uses simple string-parsing algorithms to assemble a set of significant words for each document, facilitating fast and accurate search. To enable semantic search, each document in the database is transformed into a vectorized format capturing its meaning; this involves systematically applying a neural-network-based machine-learned model. The resulting vector for each document may be effortlessly compared to any others by a simple mathematical function. While these processed data formats are utilized by each search algorithm, the original text and citation are still stored alongside these representations for retrieval. This stage produces coherent, searchable databases.

Chatbot Application

Relying on the existence of properly-formatted knowledge databases generated by the data engineering pipeline, the chatbot application implements the main RAG architecture to enable users to interact with the code or standard of interest through an LLM. A step-by-step cycle executes each time the user enters a question (Fig. 3), producing an advanced chatbot user interface.

First, the user input is used to search the knowledge databases embodying the code or standard of interest. Both lexical and semantic search approaches have strengths and weaknesses, but semantically searching theoretically better accounts for the unpredictable and varied nature of

natural language questions. This is because the exact words supplied do not have to identically match the terms present in the database documents; the overall meaning behind the words present is the basis of the search. For effective searching, this input question is preprocessed in the exact same fashion as the data in the database utilized was. For example, if searching lexically, the question is decapitalized and has its punctuation and stopwords removed, and if searching semantically, the question is converted into a vector of the same cardinality by the same text embeddings model used previously. The search algorithm selected then compares the preprocessed question with each document in the relevant database and ranks them based on their computed similarity scores. A predetermined or automatically-computed optimal quantity of top documents may be designated for retrieval. In the latter strategy, the top similarity scores computed are analyzed in relation to each other. This way, many documents with a similarly high score will all be retrieved concurrently, but one document with a score significantly higher than all others will be retrieved exclusively. These most similar portions of the code or standard to the user input query are assumed to be the portions most likely to hold the answer to the question at hand and, thus, are supplied to the LLM. Thus, the quality of user input affects system accuracy; poorly worded or unintelligible questions may not be answered correctly because the irrelevant context information retrieved by the search algorithm will not give the LLM the facts it needs to correctly answer the question. This limitation is common to other NLP systems (Mishra and Jain 2016). Applying the same preprocessing procedures to the user input as were applied to the database limits this issue, but users should take care to ensure precise language matching the code or standard is used for best results. Robustness to unanswerable questions – beyond the ability to verify the truthfulness of responses by exploring the citation provided – is outside the scope of this work; the user must supply enough information in their input question for the system to behave properly. Because the precision and consistency of user-generated input cannot be controlled, strong performance cannot be guaranteed in all cases. Future work may be undertaken to overcome this limitation. Despite these factors, the system remains user-friendly. The system is intended for use by trained and experienced professional engineers. These users will have prior knowledge about and a basic understanding of the code

or standard they are interacting with and, thus, are likely to include the proper industry-standard terminology in their queries. Furthermore, the system is more user-friendly than its contemporary counterparts (Zheng et al. 2023) because it uses an LLM to build an abstract user interface. Since LLMs can understand diverse text, users can express their question freely; this, along with the strength of the search algorithms utilized, allows variations in the structure of and exact language used in the user input to be inconsequential. While the system is quite robust, users are encouraged to apply best practices for prompting LLMs and abide by the following guidelines for best results:

- State questions briefly and simply.
- Use precise terminology consistent with the code or standard at hand.
- Adhere to a binary (yes-or-no) or interrogative (who, what, where, when) question structure.

Additionally, the LLM produces an easy-to-understand response that directly answers the question in plain language. The system may be further enhanced in the future through the use of strict question or keyword templates (Sneiders 2002; Fabbri et al. 2020). The reference localizing the relevant snippet within the code or standard is also retrieved, but is not forwarded to the LLM. Errors at this step will propagate and cause the entire system to generate an incorrect answer.

Second, a log of all previous user inputs and system responses in the conversation thus far is retrieved. Incorporating this text into future prompts is crucial in producing chatbot-esque system behavior. This feature allows users to reference prior interactions, enabling the system to better respond to rephrased or related follow-up questions. This also grants the system the ability to handle complex multi-step reasoning problems by generating responses incrementally (Wei et al. 2023).

Third, all required input information, including the system directive, conversation history, relevant context, and the actual question, is amalgamated through prompt engineering techniques to produce a cohesive and coherent but minimal prompt. This is critical to the quality of generated responses, since the LLM must consider all factors accordingly, and the rate at which they are generated. The prompt is the combination of several distinct components:

(a) A system directive is provided, giving basic context about the role and demeanor of the system.

(b) Conversation history is listed in a consistent convention where all previous user inputs and system outputs are prefixed by a distinct identifier. To prevent the LLM from confabulating both sides of the conversation, these identifiers are designated as special stopping tokens.

(c) The current user prompt – prefixed accordingly – is supplied. The most relevant documents previously identified in the searching stage are integrated with the current user input to form a clear, consistent, and coherent in-context prompt that is easily understood by the LLM.

(d) The system response identifier is appended, signaling to the LLM that it should generate a response from the proper perspective.

Because important steps, such as defining the system's role and the task at hand, are already completed by the carefully-designed system prompt, the system prompt itself acts as a general template that can be filled in with any question. This enables the user to interact with the system relatively freely. More complex prompt engineering techniques (Chen et al. 2023a; Sahoo et al. 2024) may be applied to improve system performance in the future, but such experimentation is outside the scope of this paper. Testing the system's conversation chaining capability is particularly difficult because of the lack of such a dataset for meaningful analysis. Therefore, the simple and standard prompt structure is applied herein. In-context learning enables sufficiently powerful off-the-shelf LLMs to quickly grasp and adapt to this format. Performance may be improved in the future by fine-tuning the LLMs used – adjusting their model weights based on exposure to new, domain-specific training data – on data following this format. However, fine-tuning is outside the scope of this paper and such techniques are not utilized herein.

Fourth, the system generates and returns a response to the user. Modern LLMs are extensively pre-trained on expansive public text-based datasets to acquire a strong general understanding of language. This enables them to answer questions based on context with reasonable accuracy. Again, the application of fine-tuning techniques may improve performance on this task, but this is outside the scope of this paper. The LLM response generated, which should contain the answer

to the user's question, is stored to inform future responses in the conversation and automatically combined with the citations returned by the searching stage. This attribution of source information used substantiates all responses generated and establishes search algorithm transparency, allowing users to fact-check all answers obtained.

The proposed framework is designed such that any LLM may be used as the basis of the chatbot application, enhancing system extensibility and robustness. This design protects the architecture from the blazingly fast progress in this field – what were state-of-the-art LLMs a few years ago are now obsolete, far outpaced by brand-new LLMs – because new models may be easily incorporated. This versatile architecture also permits the use of locally- or remotely-run LLMs, increasing system portability and scalability. Models may even be used via API access, enabling system integration with powerful commercial LLMs, like OpenAI's most recent GPT models, and existing open-source language modeling frameworks, such as HuggingFace (Wolf et al. 2020). These frameworks prove valuable for the application of custom and readily available open-source LLMs and for enabling the integration of diverse NLP techniques.

SYSTEM IMPLEMENTATION: A CASE STUDY

Data Engineering Pipeline

The NBCC (Canadian Commission on Building and Fire Codes 2022), which presents detailed and comprehensive requirements for the design, construction, alteration, and demolition of all buildings in Canada, is accessible as a PDF, requiring extensive cleaning procedures. The unprocessed text present in the file is extracted and separated by page before being broken down by subsection such that each provision is isolated. These partitioned provisions are then reconstructed to form the document databases while ensuring that the constituent documents are of consistent length; if a certain provision is very long, it is broken down into multiple documents, and if a certain provision is too short, it is combined with neighboring documents. These databases incorporate page numbers, division numbers, and precise section numbers, enabling accurate localization of each information snippet within the NBCC. The exact number of documents in the database can be tuned easily, but 6238 documents are contained in the system presented.

Both search algorithms work by returning the documents with the highest calculated similarity score to the question asked. The specific lexical search mechanism implemented is BM25 (Robertson and Jones 1976; Trotman et al. 2014), a powerful bag-of-words algorithm rooted in term frequency-inverse document frequency principles (tfi 1988). This algorithm scores documents containing the exact words present in the query higher, accounting for extraneous factors such as document length and word frequency across the document database. The similarity score S_l for each document D containing l_D tokens in an N-document database where the average document contains l_{avg} tokens and query Q containing n unique tokens q_i where each token appears f_{q_i} times in D and in n_{q_i} documents in the database is given by Eq. (1). The particular semantic search algorithm chosen is doc2vec (Le and Mikolov 2014; Dai et al. 2015), an extension of word2vec (Mikolov et al. 2013) that computes text embeddings vectors representing long strings of text. The similarity score S_s for each document D with an embeddings vector $\boldsymbol{D}$ and query Q with an embeddings vector $\boldsymbol{Q}$ is given by their cosine similarity (and, thus, is bounded by -1 and 1), as shown in Eq. (2). Both of these approaches are scalable, democratized, portable, and robust solutions accessible via open-source software libraries. In particular, the Gensim library (Rehurek and Sojka 2011) provides the preprocessing function used to standardize data when creating the BM25-searchable database as well as the interface and machine-learned model frameworks to create the doc2vec-searchable database. The doc2vec-based text embeddings model is trained using Gensim on all raw text within the NBCC as well as all the textual data from articles on English Wikipedia tagged as being related to civil engineering, structural engineering, or construction (in total, more than 1.1 million words). This process (Fig. 4) presents a large volume of domain-specific text to the model, teaching it to understand the meaning of words particular to relevant engineering contexts.

$$S_l = \sum_{i=1}^{n} \ln\left(\frac{N - n_{q_i} + 0.5}{n_{q_i} + 0.5}\right) \cdot \frac{f_{q_i} \cdot (k+1)}{f_{q_i} + k \cdot (1 - b \cdot (1 - \frac{l_D}{l_{avg}}))} \tag{1}$$

$$S_s = \frac{\boldsymbol{D} \cdot \boldsymbol{Q}}{\|\boldsymbol{D}\| \|\boldsymbol{Q}\|} \tag{2}$$

Chatbot Application

Various LLMs are integrated with the system in an effort to meaningfully compare the capabilities of several state-of-the-art models for this use case. The specific LLM families implemented are LLaMA-2 (Touvron et al. 2023a) and Falcon (Penedo et al. 2023). These powerful model families, both released in 2023, represent the state-of-the-art amongst entirely open-source LLMs (Beeching et al. 2023; Gao et al. 2021; Clark et al. 2018; Zellers et al. 2019; Hendrycks et al. 2021; Lin et al. 2022), making them scalable, democratized, portable, and robust solutions as well. Several versions of both of these model architectures are integrated via the open-source HuggingFace library (Wolf et al. 2020), ensuring that the entire pipeline remains completely open-source and free. Additionally, the recent GPT-3.5 (Brown et al. 2020) and GPT-4 (OpenAI 2023) models are incorporated into the system to provide a meaningful comparison between performance when using open-source models and when using commercial LLMs.

RESULTS AND DISCUSSION

General Approach

The primary measure of the success of the system is the accuracy of responses generated. For the system to be useful in high-stakes, real-world scenarios, relevant and correct answers must be consistently produced. The two primary factors affecting the correctness of system responses are the accuracy of the search algorithms chosen on natural language queries and the in-context question-answering ability of the LLMs selected on domain-specific examples.

The truthfulness of system outputs is directly limited by the relevance of the documents retrieved – if an irrelevant document is retrieved, the LLM has no chance of generating the correct answer to the question at hand. Thus, the strength of the search mechanisms utilized is paramount to system success. This is evaluated by determining the accuracy of the search algorithm in recovering the corresponding context document for sample natural language questions relevant to the NBCC.

The native question-answering capability of the LLM employed also greatly affects system performance. Even if the proper context document is retrieved, if the LLM employed is unable to understand the natural language question, locate the relevant information within the context

supplied, and produce a truthful natural language answer reflecting this information, the system will fail. Thus, the in-context question-answering ability of the LLM is critical to maximizing system performance. However, this capacity is difficult to quantify; at present, there is no widely accepted method of evaluating open-ended question-answering systems (Kamalloo et al. 2023). As such, model evaluation is best carried out by testing on a dataset formatted as multiple choice questions and answers. This format allows comparisons between model-generated and pre-defined responses to be made, enabling direct classification of responses as correct or incorrect. Without many possible answers to compare and contrast between, deducing when an open-ended question has been answered correctly is very difficult.

Testing Dataset

Because there has been no prior research into this specific area, no codes-and-standards-based dataset to meaningfully evaluate system performance exists. Thus, an original dataset is generated and utilized for meaningful analysis of system performance. Based on the approach taken to produce Alpaca (Taori et al. 2023), existing off-the-shelf LLMs are used to generate this dataset, including the labels. This process begins with a brief manual inspection of the knowledge database to identify those documents belonging to pages of the NBCC containing only well-formatted provisions; for example, documents taken from pages containing information relating to document structure and format, tables, and figures are discarded. Each of these high-quality documents (the `{}`) is then integrated into a meticulously crafted prompt (Fig. 5) passed into an LLM to generate a single open-ended multiple choice question accompanied by one correct and three incorrect, but plausible, answers based directly on the document content. The labeled correct answer among the four possibilities is selected wholly by the LLM. A set of 1285 data points is produced. This generated dataset is easily coupled into the search algorithms of interest and is readily converted into coherent system prompts of the same style that the system encounters at runtime. This complex and abstract task necessitates the use of a large, powerful LLM. The primary LLM employed for this task is the 70-billion-parameter chat version of LLaMA-2 (Touvron et al. 2023a), keeping the entire pipeline open-source and free.

This dataset covers a wide range of topics among the textual-based portions of the NBCC, with questions coming from both volumes, four divisions, and nine sections of the document (Fig. 6). Thorough manual investigations of dataset quality are not feasible because of the dataset's size, but preliminary human inspections by the authors indicate that the questions and possible answers produced closely mirror the content in the NBCC. When presented with 25 questions alongside their context and possible answers (Fig. 7) from the 1285 present in the dataset in an examination format, three trained engineers selected the correct response in 72.0%, 68.0%, and 68.0% of cases for a 75-point average of 69.3%. Subsequent deeper investigations revealed that 80.0% of the 25 samples are excellent (that is, the question is coherent and the answer labeled as correct by the LLM that generated the sample is indeed solely correct and complete), 12.0% are satisfactory (that is, the question is coherent and the labeled correct answer is correct but incomplete, with other possible answers being correct as well), and the remaining 8.0% are poor (that is, the labeled correct answer is objectively incorrect). This indicates that the dataset is of relatively high quality; it signifies that most of the answers labeled as correct are indeed correct and are generally free from hallucination. Furthermore, the data points investigated capture a representative sample of topics and complexity.

However, this imperfect dataset quality introduces some error into the reported accuracy (generated via automated testing methods). Because the automatically-generated and human-generated labels sometimes differ, there may be some cases where the system's behavior is determined to be incorrect even though it matches human behavior, or where the system's behavior is determined to be correct even though it does not match human behavior. These competing factors may limit the effect of these dataset errors on the results, but some uncertainty remains; the real accuracy may fluctuate about the reported accuracy. While the real system accuracies may not exactly match the reported figures, these values provide a reasonably accurate estimate and allow the identification and analysis of trends. The lack of an existing dataset and the difficulties in procuring a sufficiently large human-generated dataset make this issue impossible to resolve in this paper; future research may involve creating a higher-quality dataset to better quantify system performance with the help of experienced industry professionals. It must be noted that even human-generated datasets are

subject to data quality issues, causing synthetic dataset generation techniques, such as the one used in this paper, to be used across many subfields of machine learning (Nikolenko 2019; Savage 2023; Lu et al. 2024). This dataset represents the best possible means to evaluate the proposed system at present; the above experiments prove that even human experts are fallible and prone to mistakes in detail-oriented tasks (like creating such a dataset), and expanding such extensive data validation procedures to the entire dataset would be a monumental task (consuming on the order of hundreds of expensive person-hours) that is out of the scope of the paper.

Evaluation Metrics

In this paper, the accuracy of LLM responses is computed in three ways: logarithm-likelihood, semantic similarity, and lexical similarity. In each case, the LLM of interest is supplied with a system-formatted prompt (Fig. 8) combining a sample question (the second `{}`) with its corresponding context (the first `{}`). These metrics each compare the resulting model behavior with each of the corresponding possible answers to the multiple choice questions present in the generated testing dataset to determine system accuracy.

In the logarithm-likelihood approach, the model's internal state is evaluated; the probability that the LLM will generate each possible response is calculated based on its hidden parameters (Gao et al. 2021). This is computed by aggregating the probability that the model will generate each successive output token given that all previous prompt and output tokens have already been generated. The score S_{ll} of each prompt-response concatenation consisting of a series of n tokens t_i where the first j tokens comprise the prompt is given by Eq. (3). If the correct answer is the most likely text to be generated by the model, the LLM is said to have answered correctly; if any of the other incorrect answers are more likely to be generated, the LLM is determined to be incorrect in its response.

$$S_{ll} = \sum_{i=j+1}^{n} \log P(t_i \mid t_{<i}) \tag{3}$$

For the latter two approaches, the model generates a response based on the provided prompt

that is systematically compared with each possible answer. Similarity scores between the model-generated response and each possible example response are computed in both semantic (Risch et al. 2021) and lexical (Rajpurkar et al. 2018) fashions. Semantic similarity scores are computed by the same doc2vec text embeddings model utilized by the search algorithm; the similarity score S_{ss} for each labeled response X with an embeddings vector $\boldsymbol{X}$ and generated response R with an embeddings vector $\boldsymbol{R}$ is given by Eq. (4). Lexical similarity scores are represented by F1 scores tallied based on analysis of the individual words present in each response; the similarity score S_{ls} for each labeled response X with l_X total tokens where each unique token x_i appears c_{x_i} times and generated response R with l_R total tokens where each unique token r_j appears c_{r_j} times is given by Eq. (5). If the response labeled as correct is the most similar to the one actually generated, it is counted as a correct answer; if any of the other potential answers are more similar to the generated response, the response is labeled as incorrect.

$$S_{ss} = \frac{\boldsymbol{X} \cdot \boldsymbol{R}}{\|\boldsymbol{X}\|\|\boldsymbol{R}\|} \tag{4}$$

$$S_{ls} = \frac{2 \sum_i min(c_{x_i}, c_{r_j}|_{j:x_i=r_j})}{l_X + l_R} \tag{5}$$

While these metrics do not completely reflect the quality and accuracy of system responses, they provide a practical means of numerically studying the performance of different system configurations. These three approaches are widely used (Beeching et al. 2023; Guo et al. 2023) and provide a holistic view of system performance by capturing the model's internal knowledge (logarithm-likelihood) as well as the accuracy of the answers it actually generates (similarities).

Implementation Architecture

The computer used for software development and for running experiments is a 64-bit, 80-core Intel Xeon Gold 6242R processor running the Linux Ubuntu 22.04 operating system at 3.1 GHz with 128 GB of RAM. All locally-run LLMs are inferenced using a 4-bit quantization on two 48 GB NVIDIA A6000 GPUs. All software was developed in Python 3.10, primarily utilizing the

PyTorch (Paszke et al. 2019), HuggingFace's Transformers (Wolf et al. 2020), and the Gensim (Rehurek and Sojka 2011) libraries. Gensim is used for the data engineering pipeline stage (to apply search algorithms), while PyTorch and Transformers are used for the chatbot application stage (to load, configure, and inference all locally-run LLMs). Code used during the experiments is available online at `https://github.com/mqp2259/LLMforBuildingCodes`.

Experimental Results

System configuration is determined by the searching algorithm utilized as well as the LLM employed. If either of these components fails, the entire system will fail. To isolate the performance of each component, each is evaluated separately.

The effectiveness of lexical and semantic search algorithms in recovering the associated context document with each natural language query in the testing dataset from the 6238-document database is computed (Table 1). Both algorithms achieve a workable accuracy, but, despite the comparatively primitive methods used, the lexical search remains superior. However, this disparity may be a function of the testing dataset itself and these results may not generalize to real-world use. Each natural language question in the testing dataset is generated directly based off of the verbatim text present in the NBCC, so the questions tend to use the exact same terms as the ones contained in the NBCC. Real-world questions would not necessarily use these exact same terms, so the search algorithm may falter when appropriate synonyms are substituted into the query. Since the lexical algorithm searches based on the exact words present, it achieves a high accuracy here, but may not for entirely original human-generated questions.

The success of various LLMs in producing the correct answer to each natural language question present in the testing dataset given the appropriate context document (where the correct document is automatically supplied without any searching stage) is determined (Table 2). Specifically, Falcon (Penedo et al. 2023) in seven- and 40-billion-parameter versions, LLaMA-2 (Touvron et al. 2023a) in seven-, 13-, and 70-billion-parameter versions, and GPT-3.5 Turbo (ChatGPT) (Brown et al. 2020) and GPT-4 Turbo (OpenAI 2023) models are evaluated. The logarithm-likelihood metric is unavailable for the GPT models because they are closed-source; their internal state cannot be

probed, which is required to compute this figure. Unsurprisingly, model performance generally scales directly with model size (Fig. 9); this finding is consistent with previous work in the space of LLM-based NLP (Zhao et al. 2023; Shanahan 2023). However, the relationship is modest and small models still demonstrate great performance. The LLaMA-2 models consistently outperform the Falcon models, which may indicate that LLaMA-2 is better overall, consistent with previous analyses, or just better at this specific downstream task. While the testing dataset was generated by a LLaMA-2 model, it was a separate model using a very different prompt, so this should not have a major effect (even with this setup, the LLaMA-2 models did not achieve 100% accuracy). The GPT models generally perform better than their free and open-source counterparts, as expected, but the margin is very small. Surprisingly, GPT-4 seems to exhibit worse performance than GPT-3.5, but this is likely because the typically longer responses produced by this LLM are penalized by the similarity algorithms used. The other two metrics are significantly higher than semantic-similarity-based scores across all models, which is consistent with the better performance of the lexical search algorithm. This indicates a general weakness in the semantic models used rather than a weakness in the system itself; lexical-based approaches remain the standard method of LLM evaluation (Liang et al. 2023). Given the similarity between the other two superior metrics, they are likely to represent the real accuracy of the system.

Overall, performance is strong but can be further improved – the search algorithm and LLM each can behave correctly for a large majority of instances. This indicates that, when coupled together, the complete system may answer the majority of questions correctly. Crucially, the embedded citation mechanism allows engineers to easily determine when the system is incorrect simply by quickly reading the passage referenced. Given the system's good accuracy and traceability, it may be sufficient for use in some real-world applications. However, system responses should not be trusted without verification.

Qualitatively, most configurations behave similarly and all systems are able to answer questions reasonably well (Fig. 10). Complete system interactions involve all stages of the proposed framework, including processing the raw code or standard and building knowledge databases in the

data engineering pipeline, as well as searching for context documents and prompt engineering in the chatbot application (Fig. 11).

These successful system interactions contrast starkly with the corresponding interactions with a solitary LLM when no context information is provided or when an inferior prompt is provided (Fig. 12). These experiments serve to highlight the fact that each of the different system components are essential to produce proper overall system behavior. Each trial of sample interactions shows the system output under a different modification; in each column, the exact same input query and LLM as in Fig. 10 are utilized. The first trial illustrates that, when no context information is provided – that is, when the exact same system prompt is employed but with an empty context document – factually incorrect responses are generated. The second trial proves that, even when the LLM is instructed to consider the NBCC specifically – that is, when the system prompt is altered to instead preface the user input with `Human: According to the National Building Code of Canada 2020,` but a context document is still not provided – this remains the case. The third trial exhibits that, when no system directive is provided to prime the LLM – that is, when the exact same system prompt is used, including the context document, but with `You are a helpful assistant who truthfully answers a human's questions based on provided context.` removed – the system does not behave as a conversational question-answering assistant and merely regurgitates the provided context. These experiments emphasize that the RAG and prompt engineering techniques utilized in this paper are indeed necessary for strong performance on this task; without these components, system responses are uninformed and ill-composed.

These system interactions can also be compared to corresponding interactions with a trained human engineer (Fig. 13). These experiments serve to contextualize system behavior and verify several important advantages of the proposed method. Each trial of sample interactions shows the responses produced by a different engineer; in each column, the exact same input query as in Figs. 10 and 12 is utilized. These sample responses were sourced from three authors, who are trained engineers, instructed to draft written answers to questions about the NBCC using only traditional methods; internet access was restricted and the use of AI tools was prohibited. Both

methods have imperfect correctness – the system-generated response to the first sample question is incomplete and Engineer #3's response to the second sample question is wrong. This proves that even trained engineers make mistakes and underscores that responses generated by both methods must be verified. Qualitatively, these human- and system-generated responses are quite similar. System-generated responses are easier to understand because they are well-formatted, being entirely free of spelling and grammatical errors, and state the answer to the user's question directly and in clear terms. In contrast, human-generated answers are prone to typographical and other errors because writing a well-crafted original response takes a significant amount of time and effort. However, system-generated responses also tend to be less thorough because the proposed method is only able to consult a small excerpt of the NBCC that may not contain all the relevant and required information. This means that the system is particularly valuable for specific provisions, while more abstract and general questions may require further system improvement. Conversely, human engineers are able to understand the broader context of the relevant provisions within the code or standard because they can freely traverse the document. This is important because oftentimes related provisions are not located near each other within a code or standard. A major benefit of the proposed system is that it returns accurate citations for users to quickly find the relevant section of the code or standard; traditional computerized search functions that work by exact character matching are overly sensitive to small input perturbations, but the advanced search algorithms used by the system are not. Crucially, this experiment verifies that the responses can be obtained from the proposed system significantly faster than by the traditional method; system execution time varies greatly depending on hardware configuration and question complexity, but may be as much as 10 times faster than the average engineer's response time of 3 min 31 s. The absence of a substantial and high-quality human-made dataset currently limits the scope for a more comprehensive comparative analysis. However, this analysis is planned for future work.

This paper merely introduces a complete framework enabling engineers to answer questions about codes and standards in a novel manner. The framework itself is highly modularized, meaning that increasingly powerful components can be effortlessly integrated as they emerge. Therefore,

the performance of systems implementing this framework can be further improved. For example, superior search algorithms can be developed and new LLMs, such as LLaMA-3 (Touvron et al. 2023b), can be incorporated in the future.

CONCLUSIONS AND FUTURE WORK

Final Remarks

The promising results of this paper indicate that using modern LLMs to answer questions about civil and structural engineering codes and standards is feasible. A lexical search algorithm coupled with a LLaMA-2 model produced the best results, and system performance scaled with LLM size. The framework designed is scalable, democratized, portable, and robust; even small models performed well and, crucially, the pipeline was entirely open-source. Considerable strides to overcoming significant obstacles of the past, including the capacity for natural language interaction and LLM hallucination, have been made in this work. The proposed framework and system have the potential to fundamentally change how engineers interact with codes and standards, improving the efficiency of the design process. Furthermore, this framework holds promise for powerful applications to diverse domains where the truthfulness of precise and complex technical information is important. With future optimization and the adoption of rapidly developing NLP technologies, the effectiveness of systems implementing the proposed framework will only continue to improve.

However, it should be acknowledged that thorough analysis of the risks associated with the use of this framework and system has not yet been undertaken. The system developed displays strong behavior both quantitatively and qualitatively and promises to be a useful tool to assist designers and engineers in completing their work more efficiently and effectively, but it should only be used with extreme caution as an assistance to trained professional engineers in its current state.

Future Work

Work on this project should continue in many areas. Primarily, the application of state-of-the-art and emerging AI technologies and practices may improve the present results. For example, employing parameter-efficient fine-tuning methods with specialized datasets to expose the LLMs

used to more technical civil and structural engineering jargon and to improve the in-context question-answering ability of LLMs employed may significantly enhance system behavior. Additionally, implementing reinforcement learning techniques based on automated reward functions or human annotations may better the quality and safety of system responses. Moreover, applying more complex prompt structures and patterns, such as few-shot prompting, may improve behavior further. Furthermore, other existing and forthcoming LLMs may achieve better performance than the ones employed in this preliminary study. Dataset procurement, both for system training via fine-tuning or reinforcement learning or for improved system performance analysis, has the potential to enhance systems implementing the proposed framework and to provide a better means of benchmarking performance. Data quality is of the utmost importance in machine learning projects; collecting and analyzing higher quality data, preferably extracted from real-world contexts, may substantially improve model training and evaluation. Crowdsourced annotations of realistic data would overcome current data quality limitations, and this is currently being explored. Assembling a large, high-quality dataset may facilitate further research into this problem by creating a consistent method of evaluation standardized across many distinct works. Such a dataset may facilitate other features, such as robustness to unanswerable questions.

Ensuring compliance with codes and standards complex task that manifests in many different forms. Besides providing a means for engineers to interact verbally with codes and standards in a question-and-answer fashion, the presented framework may be combined with other new and emerging technologies in the future. For example, it may be integrated with existing automated tools for checking compliance of particular engineering designs and drawings. Furthermore, it may be combined with emerging systems currently in development enabling generative-AI-based structural design. The proposed framework has other vast and powerful potential applications within civil and structural engineering but outside of the design domain; for example, similar systems may be applied to improve safety on construction sites.

DATA AVAILABILITY STATEMENT

Some or all data, models, or code generated or used during the study are available in a repos-

itory online in accordance with funder data retention policies: `https://github.com/mqp2259/LLMforBuildingCodes`.

ACKNOWLEDGMENTS

This research was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) (ALLRP 581074-22) and the University of Alberta (Research Exploration Fund).

REFERENCES

(1988). "Term weighting approaches in automatic text retrieval." *Information Processing and Management*, 24(5), 323–328.

Aluga, M. (2023). "Application of ChatGPT in civil engineering." *East African Journal of Engineering*, 6(1), 104–112.

Anil, R. et al. (2023). "PaLM 2 technical report.

Beach, T. H., Rezgui, Y., Li, H., and Kasim, T. (2015). "A rule-based semantic approach for automated regulatory compliance in the construction sector." *Expert Systems with Applications*, 42(12), 5219–5231.

Beeching, E. et al. (2023). "Open LLM leaderboard.

Bommasani, R. et al. (2022). "On the opportunities and risks of foundation models.

Brown, T. B. et al. (2020). "Language models are few-shot learners.

Canadian Commission on Building and Fire Codes (2022). "National Building Code of Canada: 2020, <https://doi.org/10.4224/w324-hv93> (Mar). v. 1, 792 p.; v. 2, 708 p.

Chen, B., Zhang, Z., Langrené, N., and Zhu, S. (2023a). "Unleashing the potential of prompt engineering in large language models: a comprehensive review.

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). "Reading wikipedia to answer open-domain questions.

Chen, W., Verga, P., de Jong, M., Wieting, J., and Cohen, W. (2023b). "Augmenting pre-trained language models with QA-memory for open-domain question answering.

Ching, F. and Winkel, S. (2021). *Building Codes Illustrated: A Guide to Understanding the 2021 International Building Code*. Building Codes Illustrated. Wiley, <https://books.google.ca/books?id=Ke5JEAAAQBAJ>.

Chowdhery, A. et al. (2022). "PaLM: Scaling language modeling with pathways.

Clark, P. et al. (2018). "Think you have solved question answering? Try ARC, the AI2 reasoning challenge.

Dai, A. M., Olah, C., and Le, Q. V. (2015). "Document embedding with paragraph vectors.

Devlin, J., Chang, M. W., Lee, K., and Toutanova, K. (2019). "BERT: Pre-training of deep bidirectional transformers for language understanding.

Douka, S., Abdine, H., Vazirgiannis, M., Hamdani, R. E., and Amariles, D. R. (2022). "JuriBERT: A masked-language model adaptation for french legal text.

Fabbri, A., Ng, P., Wang, Z., Nallapati, R., and Xiang, B. (2020). "Template-based question generation from retrieved sentences for improved unsupervised question answering." *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, eds., Online, Association for Computational Linguistics, 4508–4513, <https://aclanthology.org/2020.acl-main.413> (July).

Feldman, P., Foulds, J. R., and Pan, S. (2023). "Trapping LLM hallucinations using tagged context prompts.

Fuchs, S. (2021). "Natural language processing for building code interpretation: Systematic literature review report (May).

Gao, L. et al. (2021). "A framework for few-shot language model evaluation, <https://doi.org/10.5281/zenodo.5371628> (September).

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. (2024). "Retrieval-augmented generation for large language models: A survey.

Guo, Q., Cao, S., and Yi, Z. (2022). "A medical question answering system using large language models and knowledge graphs." *International Journal of Intelligent Systems*, 37(11), 8548–8564.

Guo, Z. et al. (2023). "Evaluating large language models: A comprehensive survey.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. W. (2020). "REALM: Retrieval-augmented language model pre-training.

Head, C. B., Jasper, P., McConnachie, M., Raftree, L., and Higdon, G. (2023). "Large language model applications for evaluation: Opportunities and ethical implications." *New Directions for Evaluation*, 2023(178-179), 33–46.

Hendrycks, D. et al. (2021). "Measuring massive multitask language understanding.

Hoffmann, J. et al. (2022). "Training compute-optimal large language models.

Islam, S. et al. (2023). "A comprehensive survey on applications of transformers for deep learning tasks.

Jiang, J., Zhou, K., Dong, Z., Ye, K., Zhao, W. X., and Wen, J.-R. (2023). "StructGPT: A general framework for large language model to reason over structured data.

Jiang, S., Wang, N., and Wu, J. (2019). "Combining bim and ontology to facilitate intelligent green building evaluation." *Journal of Computing in Civil Engineering*, 33(1), 04018069.

Jiang, Z., Araki, J., Ding, H., and Neubig, G. (2021). "How can we know when language models know? on the calibration of language models for question answering." *Transactions of the Association for Computational Linguistics*, 9, 962–977.

Kamalloo, E., Dziri, N., Clarke, C. L. A., and Rafiei, D. (2023). "Evaluating open-domain question answering in the era of large language models.

Karpukhin, V. et al. (2020). "Dense passage retrieval for open-domain question answering.

Krishna, K., Roy, A., and Iyyer, M. (2021). "Hurdles to progress in long-form question answering.

Le, Q. V. and Mikolov, T. (2014). "Distributed representations of sentences and documents.

Lee, K., Chang, M. W., and Toutanova, K. (2019). "Latent retrieval for weakly supervised open domain question answering.

Liang, P. et al. (2023). "Holistic evaluation of language models.

Lin, S., Hilton, J., and Evans, O. (2022). "TruthfulQA: Measuring how models mimic human falsehoods.

Lu, Y., Shen, M., Wang, H., Wang, X., van Rechem, C., Fu, T., and Wei, W. (2024). "Machine learning for synthetic data generation: A review.

McKenna, N., Li, T., Cheng, L., Hosseini, M. J., Johnson, M., and Steedman, M. (2023). "Sources of hallucination by large language models on inference tasks.

Menick, J. et al. (2022). "Teaching language models to support answers with verified quotes.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). "Efficient estimation of word representations in vector space.

Min, B. et al. (2023). "Recent advances in natural language processing via large pre-trained language

models: A survey." *ACM Comput. Surv.*, 56(2).

Mishra, A. and Jain, S. K. (2016). "A survey on question answering systems with classification." *Journal of King Saud University - Computer and Information Sciences*, 28(3), 345–361.

Naveed, H. et al. (2023). "A comprehensive overview of large language models.

Nawari, N. O. (2019). "Generalized adaptive framework for computerizing the building design review process." *Journal of Architectural Engineering*, 25(2), 04019004.

Nikolenko, S. I. (2019). "Synthetic data for deep learning.

OpenAI (2023). "GPT-4 technical report.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). "Pytorch: An imperative style, high-performance deep learning library." *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 8024–8035.

Penedo, G. et al. (2023). "The RefinedWeb dataset for Falcon LLM: Outperforming curated corpora with web data, and web data only.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018a). "Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2018b). "Language models are unsupervised multitask learners.

Rajpurkar, P., Jia, R., and Liang, P. (2018). "Know what you don't know: Unanswerable questions for SQuAD.

Ram, O. et al. (2023). "In-context retrieval-augmented language models.

Rehurek, R. and Sojka, P. (2011). "Gensim–python framework for vector space modelling." *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2).

Risch, J., Möller, T., Gutsch, J., and Pietsch, M. (2021). "Semantic answer similarity for evaluating question answering models.

Robertson, S. E. and Jones, K. S. (1976). "Relevance weighting of search terms." *Journal of the*

American Society for Information Science, 27(3), 129–146.

Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., and Chadha, A. (2024). "A systematic survey of prompt engineering in large language models: Techniques and applications.

Savage, N. (2023). "Synthetic data could be better than real data." *Nature*.

Shanahan, M. (2023). "Talking about large language models.

Singhal, K. et al. (2023). "Towards expert-level medical question answering with large language models.

Sneiders, E. (2002). "Automated question answering using question templates that cover the conceptual model of the database." *Natural Language Processing and Information Systems*, B. Andersson, M. Bergholtz, and P. Johannesson, eds., Berlin, Heidelberg, Springer Berlin Heidelberg, 235–239.

Taori, R. et al. (2023). "Stanford Alpaca: An instruction-following LLaMA model." *GitHub repository*.

Thoppilan, R. et al. (2022). "LaMDA: Language models for dialog applications.

Touvron, H. et al. (2023a). "LLaMA 2: Open foundation and fine-tuned chat models.

Touvron, H. et al. (2023b). "LLaMA: Open and efficient foundation language models.

Trotman, A., Puurula, A., and Burgess, B. (2014). "Improvements to BM25 and language models examined." *Proceedings of the 19th Australasian Document Computing Symposium*, ADCS '14, New York, NY, USA, Association for Computing Machinery, 58–65, <https://doi.org/10.1145/2682862.2682863>.

Vaswani, A. et al. (2017). "Attention is all you need." *CoRR*, 30.

Wason, R. (2018). "Deep learning: Evolution and expansion." *Cognitive Systems Research*, 52, 701–708.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2023). "Chain-of-thought prompting elicits reasoning in large language models.

Wolf, T. et al. (2020). "HuggingFace's transformers: State-of-the-art natural language processing.

Wu, C., Zhang, X., Zhang, Y., Wang, Y., and Xie, W. (2023). "PMC-LLaMA: Further finetuning

LLaMA on medical papers.

Xu, X., Wang, H., Ma, L., and Luo, H. (2019). "Automatic construction of semantic knowledge graph from building codes." *Advanced Engineering Informatics*, 42, 100964.

Yang, F. et al. (2023a). "Empower large language model to perform better on industrial domain-specific question answering.

Yang, H., Liu, X.-Y., and Wang, C. D. (2023b). "FinGPT: Open-source financial large language models.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. (2019). "HellaSwag: Can a machine really finish your sentence?

Zhang, J. and El-Gohary, N. M. (2016). "Semantic nlp-based information extraction from construction regulatory documents for automated compliance checking." *Journal of Computing in Civil Engineering*, 30(2), 04015014.

Zhao, W. X. et al. (2023). "A survey of large language models.

Zheng, Z., Chen, K.-Y., Cao, X.-Y., Lu, X.-Z., and Lin, J.-R. (2023). "LLM-FuncMapper: Function identification for interpreting complex clauses in building codes via LLM.

Zhong, B., Pan, X., Love, P. E., Sun, J., and Tao, C. (2020). "Hazard analysis: A deep learning and text mining framework for accident prevention." *Advanced Engineering Informatics*, 46, 101152.

Zhuang, Y., Yu, Y., Wang, K., Sun, H., and Zhang, C. (2023). "Toolqa: A dataset for llm question answering with external tools.

List of Tables

TABLE 1. Results of lexical and semantic search algorithms on the testing dataset. Figures represent the fraction of times that the correct context document is present among the top x number of most relevant documents returned by the search algorithm.

Search Algorithm	Search Accuracy			
	Top 1	Top 3	Top 5	Top 10
Lexical (BM25)	59.0%	76.2%	80.9%	85.5%
Semantic (doc2vec)	32.7%	47.6%	52.8%	60.0%

TABLE 2. Results of various LLMs on the testing dataset.

Large Language Model	Response Accuracy		
	Logarithm Likelihood	Semantic Similarity	Lexical Similarity
Falcon-7B	72.9%	44.5%	73.5%
Falcon-40B	75.9%	48.4%	76.3%
LLaMA-2-7B	77.2%	44.0%	77.4%
LLaMA-2-13B	79.9%	47.0%	78.1%
LLaMA-2-70B	82.0%	47.1%	77.5%
GPT-3.5 (ChatGPT)	–	53.6%	80.2%
GPT-4	–	49.7%	77.3%

List of Figures

13 Sample human engineer interactions. Subfigures (a), (b), and (c) were written by Engineer #1. Subfigures (d), (e), and (f) were written by Engineer #2. Subfigures (g), (h), and (i) were written by Engineer #3. All participants were instructed to draft written responses to questions about the NBCC using only traditional methods. 52

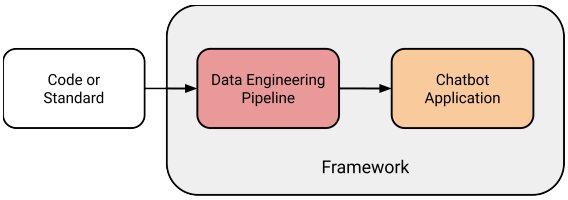


Fig. 1. General architecture of the two-step framework converting the raw code or standard of interest into searchable databases and an interactive chatbot.

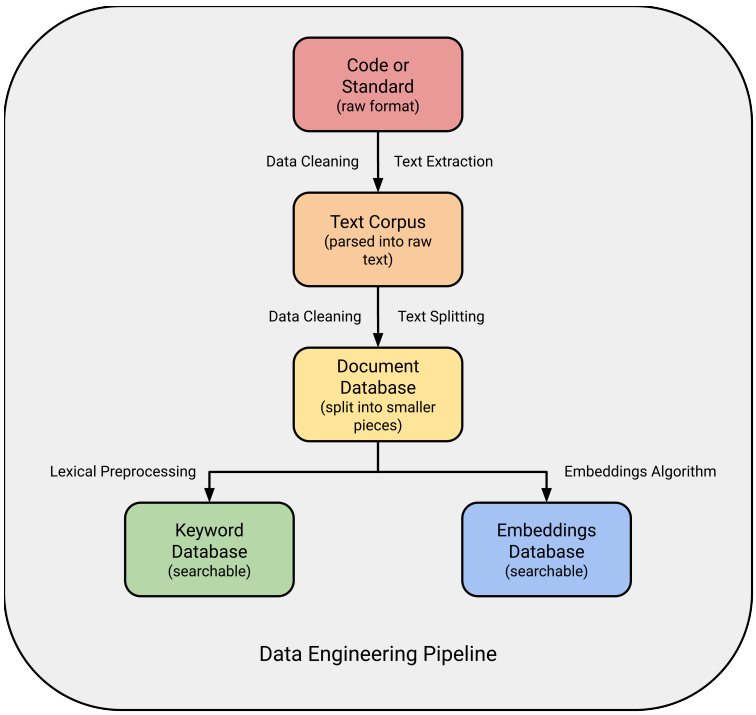


Fig. 2. General architecture of the data engineering pipeline transforming the raw information in the code or standard into searchable databases used by the chatbot application.

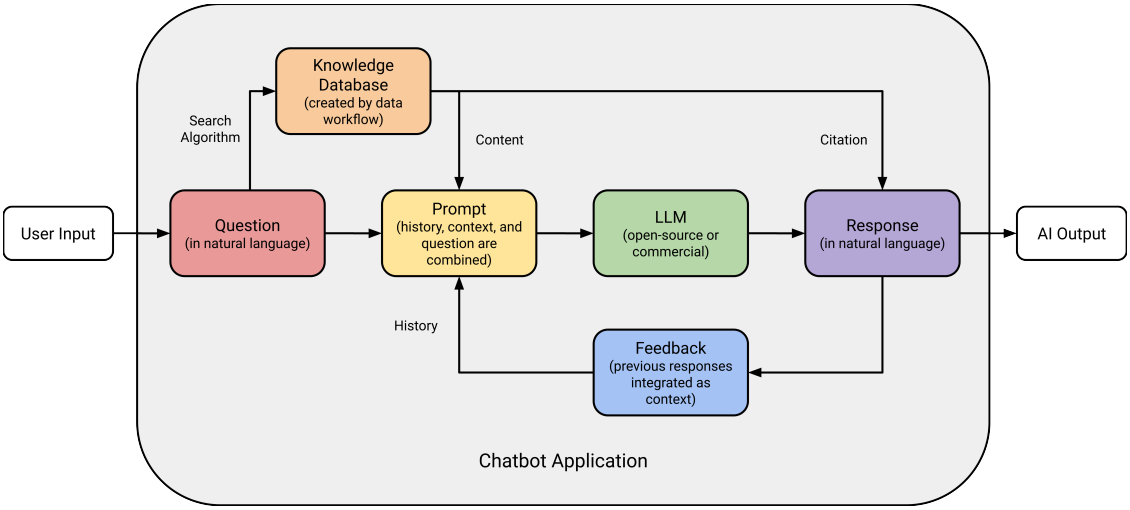


Fig. 3. General architecture of the chatbot application allowing users to interact with the code or standard of interest in natural language.

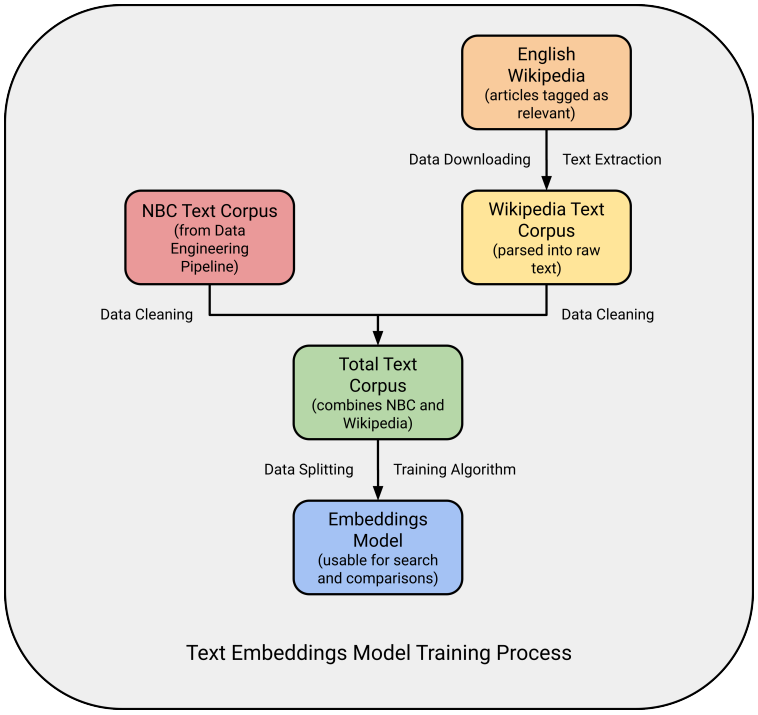


Fig. 4. General architecture of the training process for the doc2vec-based text embeddings model.

```
Generate exactly one open-ended question with one correct and
three plausible incorrect answers based only on the content in the
following document.

"{}"

All answers must be written in full sentences and they must be
of similar length.  Respond in the following format:

Question:  <question>

A: <correct answer to question>
B: <incorrect answer 1 to question>
C: <incorrect answer 2 to question>
D: <incorrect answer 3 to question>

Answer:  <letter corresponding to correct answer>
Explanation:  <justification of answer>

Question:
```

Fig. 5. The prompt used to generate the testing dataset.

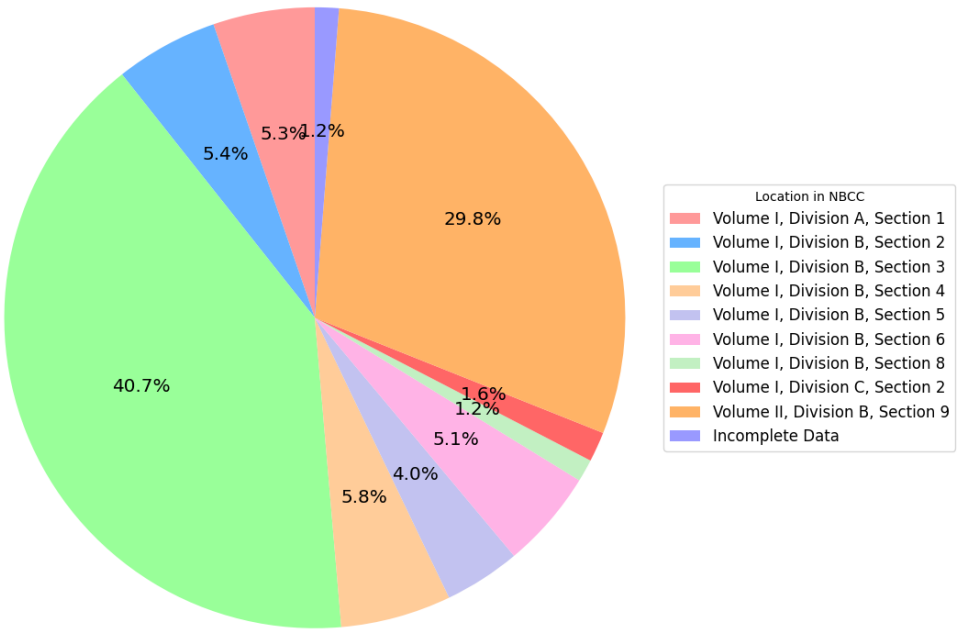


Fig. 6. Data point share as a function of location in the NBCC (Canadian Commission on Building and Fire Codes 2022).

Question: According to the excerpt from the building code, what is required for materials, components, assemblies, joints in materials, and junctions between components and junctions between assemblies exposed to precipitation?

Correct Answer: Sealed to prevent ingress of precipitation or drained to direct precipitation to the exterior.
Incorrect Answer 1: Sealed to prevent ingress of precipitation only.
Incorrect Answer 2: Drained to direct precipitation to the exterior only.
Incorrect Answer 3: Neither sealed nor drained.

Fig. 7. Sample data point, taken from Section 5.6.2.1. of the NBCC (Canadian Commission on Building and Fire Codes 2022), used to test the system.

```
You are a helpful assistant who truthfully answers a human's
questions based on provided context.

Human:  Consider the following:  "{}" {}
AI:
```

Fig. 8. The prompt utilized by the chatbot application to produce system responses.

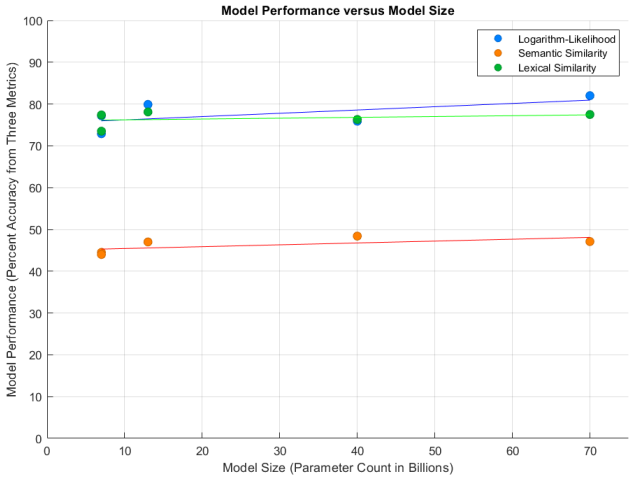
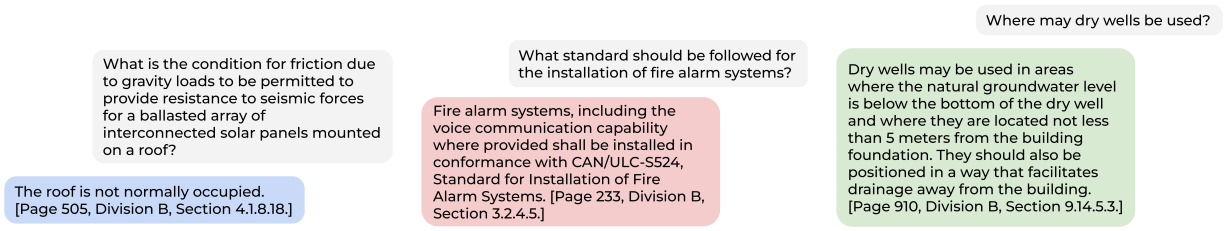


Fig. 9. Open-source LLM performance as a function of parameter count.



(a) BM25 and Falcon-7B **(b)** doc2vec and LLaMA-2-70B **(c)** BM25 and GPT-3.5 Turbo

Fig. 10. Sample correct system interactions.

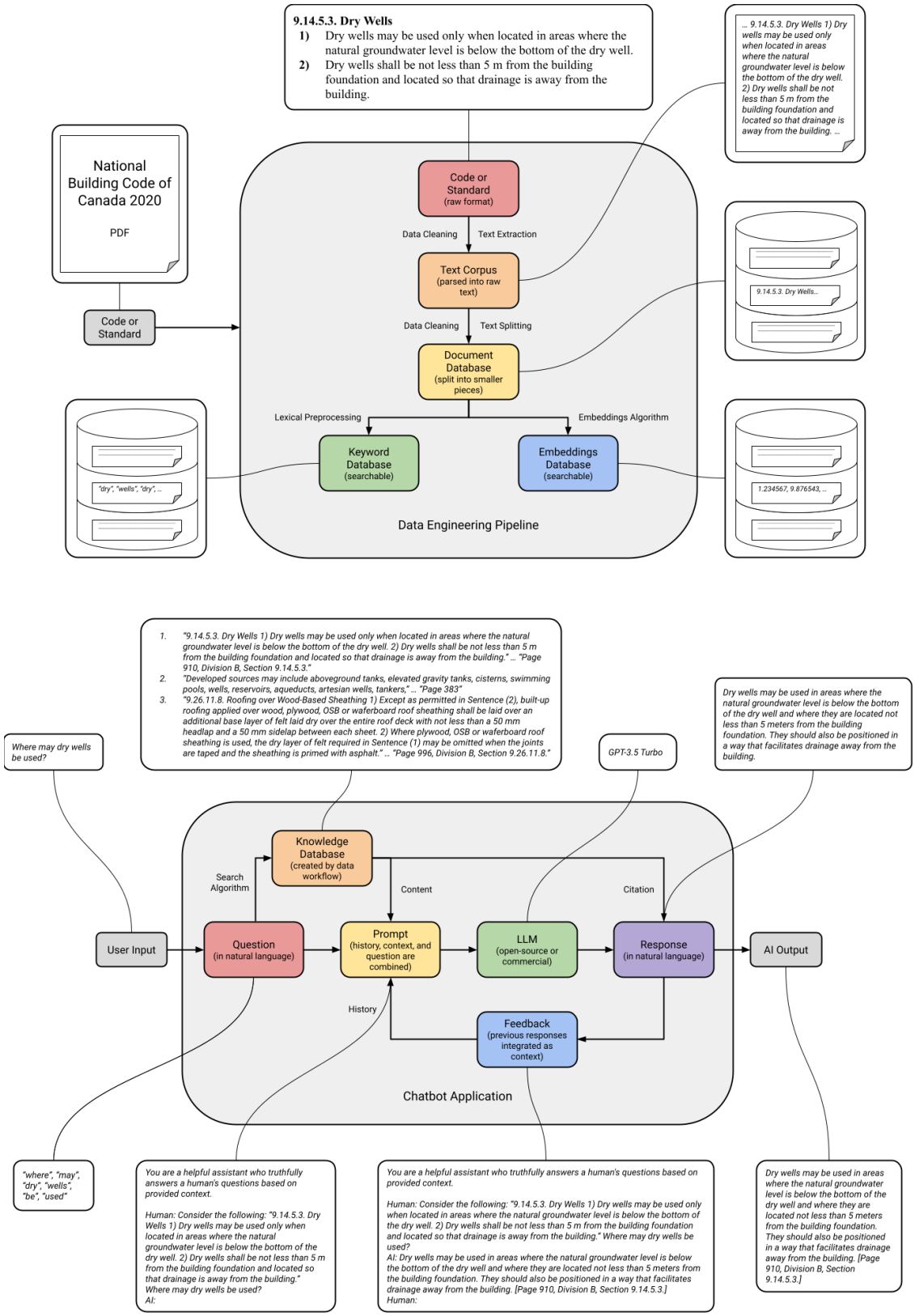
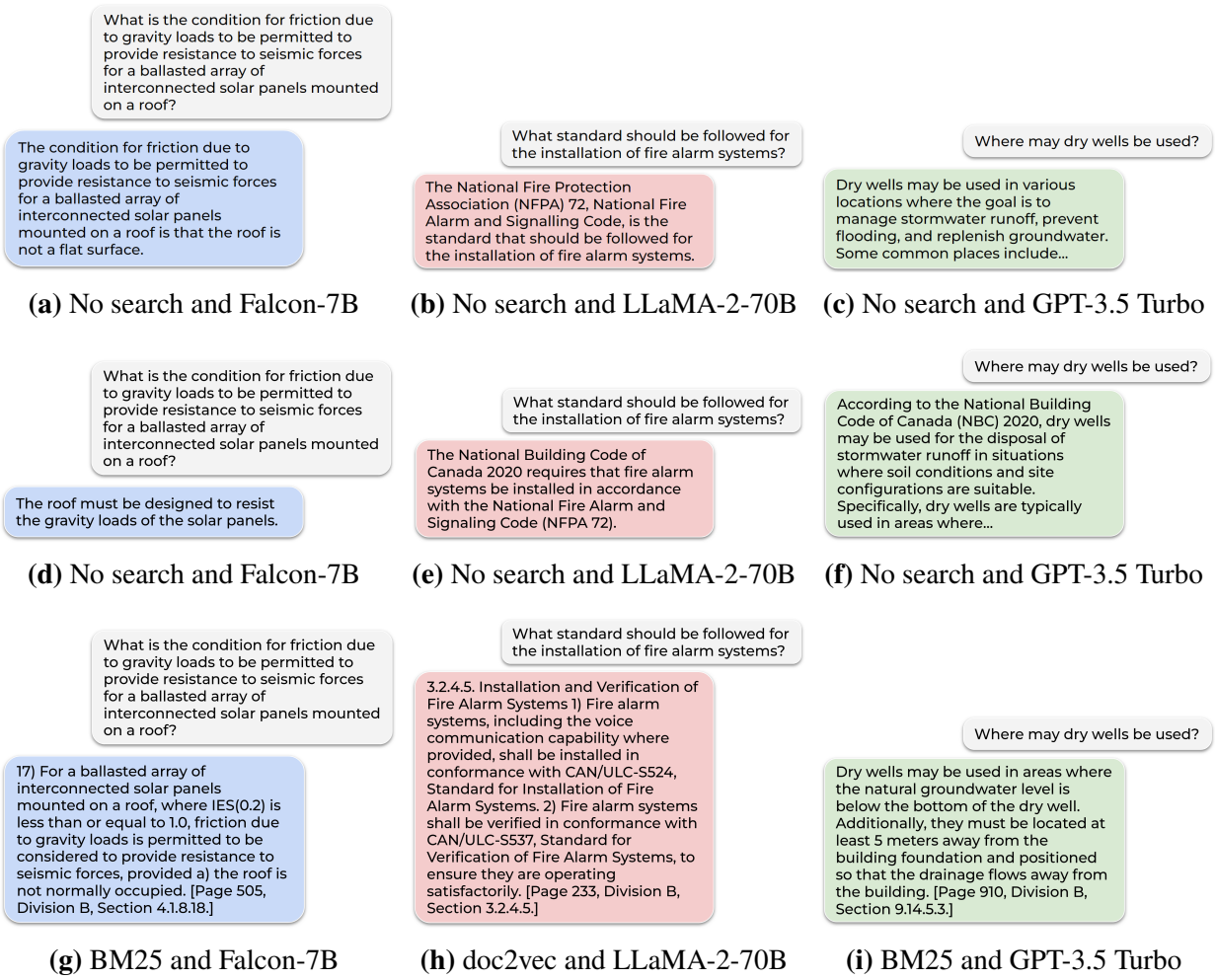


Fig. 11. Step-by-step execution details of the system on a sample input using real data from the NBCC (Canadian Commission on Building and Fire Codes 2022).

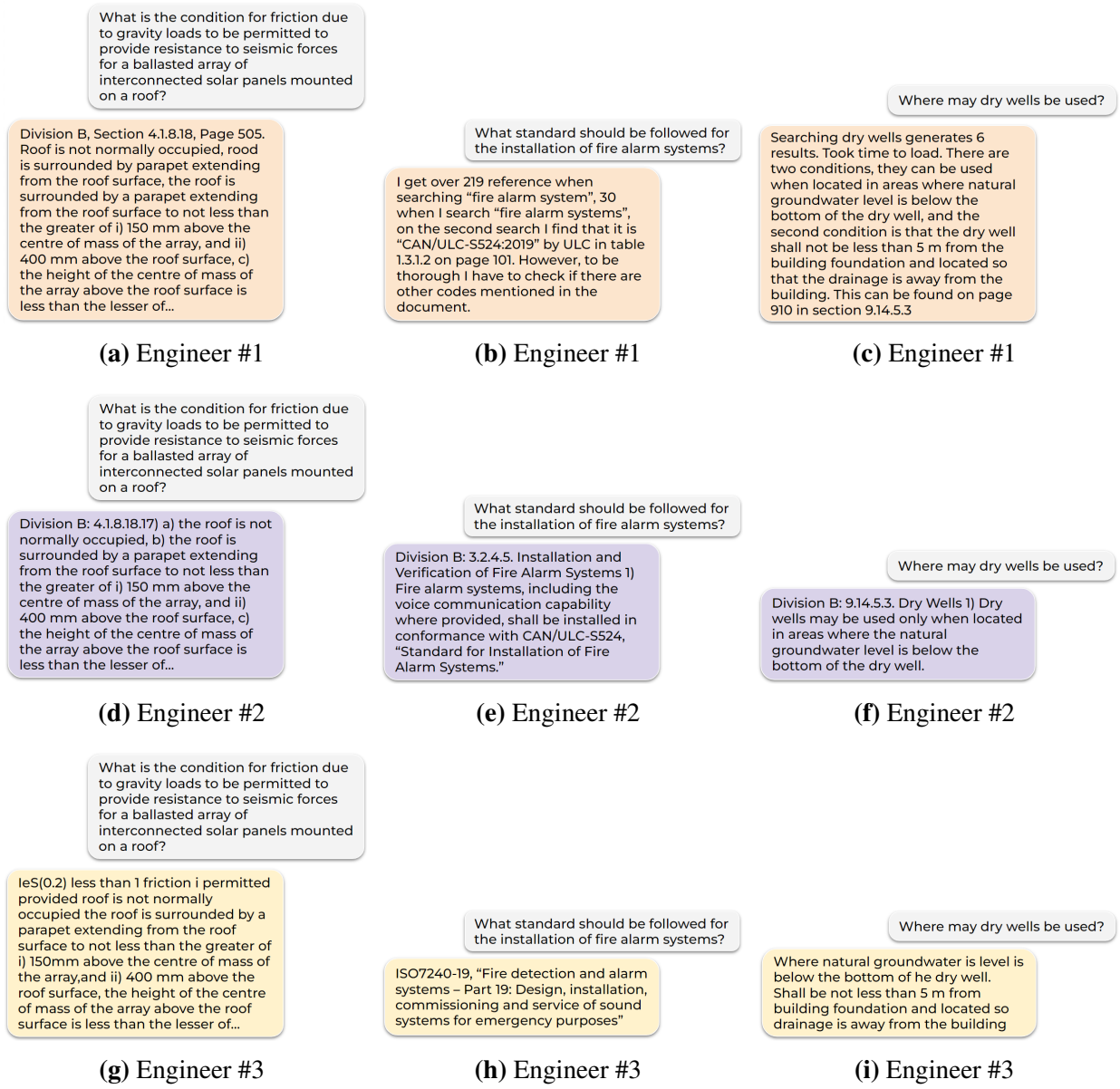


(a) No search and Falcon-7B **(b)** No search and LLaMA-2-70B **(c)** No search and GPT-3.5 Turbo

(d) No search and Falcon-7B **(e)** No search and LLaMA-2-70B **(f)** No search and GPT-3.5 Turbo

(g) BM25 and Falcon-7B **(h)** doc2vec and LLaMA-2-70B **(i)** BM25 and GPT-3.5 Turbo

Fig. 12. Sample incorrect system interactions. Subfigures (a), (b), and (c) utilize the exact system prompt but with no context document provided. Subfigures (d), (e), and (f) preface the user input with "Human: According to the National Building Code of Canada 2020," and append "AI:" with no context document provided. Subfigures (g), (h), and (i) utilize the exact system prompt, including the context document returned by the respective search algorithm, but with no "You are a helpful assistant who truthfully answers a human's questions based on provided context." directive.



(a) Engineer #1 **(b)** Engineer #1 **(c)** Engineer #1

(d) Engineer #2 **(e)** Engineer #2 **(f)** Engineer #2

(g) Engineer #3 **(h)** Engineer #3 **(i)** Engineer #3

Fig. 13. Sample human engineer interactions. Subfigures (a), (b), and (c) were written by Engineer #1. Subfigures (d), (e), and (f) were written by Engineer #2. Subfigures (g), (h), and (i) were written by Engineer #3. All participants were instructed to draft written responses to questions about the NBCC using only traditional methods.