Structured Document Generation for Industrial Equipment

Karol Lynch, Fabio Lorenzi, John Sheehan, Duygu Kabakci-Zorlu, Bradley Eck

IBM Research Europe, Dublin, Ireland
karol_lynch@ie.ibm.com, fabio.lorenzi1@ibm.com, john.d.sheehan@ie.ibm.com, duygu.kabakci.zorlu@ibm.com, bradley.eck@ie.ibm.com

Abstract

We describe an application that uses large language models to generate structured documents related to industrial equipment, specifically focusing on Failure Modes and Effects Analysis (FMEAs). Our novel application uses techniques in structured document generation, in-context learning, and ensembling to create high-quality structured content that subject matter experts supervise through a user-centric interface that presents FMEA entities as UI elements. Novel evaluation metrics for structured document generation are also proposed. Our empirical results, based on 71 asset evaluations, demonstrate the individual and combined contributions of these techniques, with an overall effectiveness that varies between a recall of 0.669 and a precision of 0.91. Qualitative feedback from target users validates the practicality of the described approach to seamlessly integrate expert supervision with generative AI in a labour-saving workflow.

Introduction

In this paper we describe an emerging application of AI, in particular Large Language Models (LLMs), for the generation of a particular class of structured documents. Failure Modes and Effects Analysis (FMEAs) document the composition of a piece of equipment or system and its likely points of failure (Rausand and Høyland 2004). FMEAs also typically include information on the consequences or effects of a failure and possible interventions to prevent it.

The traditional method of creating an FMEA relies on a group of experts. Often a facilitator takes the convened experts through an interview process aimed at populating the various parts of the final FMEA document (Cooper 2015). These parts typically include the equipment boundary and list of maintainable components, failure modes of each component comprised of a degradation mechanism and influence, mitigation strategies for each failure mode and extended descriptions of the mitigation activities used to describe the preventative work to be carried out.

The information contained in an FMEA comes from a variety of sources. Equipment manufacturers provide manuals describing the operation and maintenance of their products. Professional societies and companies maintain specifications used in equipment procurement. Best practice guides prepared by maintenance engineers have recommendations for design and operation. Because FMEAs are an expert assimilation of data from a variety of sources they are both expensive to create (Cooper 2015) and valuable to have. Due to the cost, many companies do not have FMEAs for all of their equipment. Our motivation here is to leverage today's AI technology to speed up FMEA creation.

FMEA documents pose several challenges for the application of AI methods. FMEAs are inherently structured documents and the ability to store and manage FMEAs in a structured database and support for a UI with structured fields is essential for effective FMEA management. Any solution for generating FMEAs should capture their complex structure. FMEA documents are highly specialised and domain specific. FMEAs with more than 1500 failure modes are not uncommon and their creation incurs multi-million dollar labour costs (Cooper 2015). Thus FMEAs are often proprietary and unavailable to LLMs for training. Further, FMEAs exhibit a tree-like structure with one-to-many relationship between FMEA elements such as components, degradation mechanisms and degradation influences. This tree-like structure means errors propagate through the document (e.g., an incorrect component leads to several incorrect degradation mechanisms). These characteristics of FMEAs make an interesting emerging application of generative AI.

We continue this paper by summarizing related work, followed by a description of our system for generating FMEAs that includes contributions in structured document generation, in-context learning and LLM ensembles. An additional contribution involves a metric for evaluating both the structure and semantics of new content. Further the system's performance is evaluated through both empirical and end-user studies. Finally, we discuss the path to end-user deployment of this capability in the enterprise setting.

Related Work

Related work is organised by the following areas of particular relevance: 1) In-Context Learning; 2) Ensembling; 3) LLM Evaluation and 4) Structured Document Generation.

In-Context learning In-Context Learning (a.k.a. Few-shot learning) (Brown, Mann et al. 2020) refers to the technique of providing in-prompt examples that are directly related to the task at hand and it forms a crucial component of

our application. Liu et al. (2022) proposed identifying pertinent examples using the k-nearest neighbours (k-NN) algorithm within the embedding space of a sentence encoder. Nori et al. (2023) employ a similar technique which they refer to as dynamic few-shot prompting (DFSP). DFSP is an efficient and effective method of harnessing task specific training data, and is especially valuable in the context of multiple LLMs as it can be used without updating billions of parameters for each model. In our application we add human supervision as a final example selection step after applying DFSP. We also incorporate previous work on example shuffling for example order sensitivity (Liu et al. 2022; Pezeshkpour and Hruschka 2023; Zheng et al. 2023).

Ensembles Ensembling methods that leverage the diverse strengths of various LLMs and/or prompts often demonstrate improved task performance (Jiang, Ren, and Lin 2023). Wang et al. (2023) employ multiple chain-of-thought prompts (Wei et al. 2024) to induce multiple reasoning paths within a single LLM, with the final answer being determined through a majority vote among the outputs generated by the different reasoning paths. In contrast to our approaches, this method is primarily focused on atomic LLM outputs and reasoning tasks. Further it does not support the assembling of pieces of different outputs into a single coherent response. LLM-BLENDER (Jiang, Ren, and Lin 2023), is an ensemble framework that takes the outputs of multiple LLMs, selects the best responses for a given question and then merges them into a single improved response. In contrast to this and other approaches based on supervised learning (Lu et al. 2024) our ensembling method does not require the training of (multiple) additional models.

LLM Evaluation A variety of metrics are employed for evaluating the performance of LLMs, including ROUGE (Lin 2004), perplexity (Jelinek et al. 1977) and BLEU (Bilingual Evaluation Understudy) (Papineni et al. 2002). However these metrics do not consider document structure. Moreover, ROUGE and BLEU, for instance, only consider surface text representations (i.e., synonyms are not considered to be matches) and operate at the token rather than entity level. Other metrics such as BERTScore (Zhang et al. 2020) do consider semantic representations but operate either at the sentence or the token level. In contrast, we propose an approach for evaluating structured LLM outputs that transparently maps to key application performance indicators, considers the semantic meaning of the text rather than simply its surface representation and operates at the entity rather than the token level.

Structured Generation Although LLMs demonstrate huge capability in natural language generation, producing complex structured documents in an unsupervised setting remains a challenging task. There has been significant effort invested in producing structured outputs for supervised LLM based applications such as coding assistants (Jiang et al. 2024b). In the unsupervised setting Béchard and Ayala (2024) leverage Retrieval-Augmented Generation to improve the quality of generated structured documents representing work-flows. Finally, the Langchain framework[1] offers a partial approach to structured generation, which involves defining an input schema and optionally including few shot examples of application schemas for formatting prompts and validating responses.

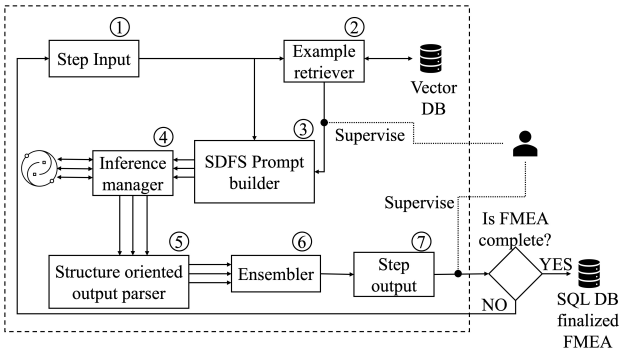

Figure 1: Flow Diagram of our Interactive Application

Solution Approach

Our solution for generating FMEAs uses in-context learning (ICL), structured generation and ensembling to enable interactive creation of new documents. Examples in our system come from an existing database of hundreds of FMEAs created by experts. Users drive the creation process through a graphical interface that integrates their supervision with LLM generation. The overall application flow (c.f., Figure 1) proceeds as follows: 1) The user initiates FMEA creation by inputting an asset name; 2) User Input is encoded and related examples are retrieved from a Vector DB, then supervised by the user; 3) Multiple prompts are constructed using the selected examples, step instruction and prompt templates; 4) The inference manager dispatches prompts to an LLM service; 5) Individual LLM responses are parsed and validated; 6) Ensemble methods merge multiple responses into a single output which is supervised by the user; 7) Final FMEAs are assembled from all steps and stored in an SQL DB.

The generation of structured content facilitates a UI design that clearly visualises FMEA entities such as components, allowing for ease of supervision. As depicted in Figure 2, the UI design encompasses several key features: i) Supervision of example selection; ii) Representation of FMEA entitites as UI elements; iii) Decomposition into FMEA aligned steps and iv) Supervision of the outputs of each step. The effectiveness of our tool depends on the quality of the generated FMEAs, and we now describe multiple techniques that we devise to achieve this.

Structured Generation

We combine several techniques to generate structured FMEA documents using LLMs that were not aligned to this task. Firstly, we devise a dynamic prompting strategy that issues a content dependent series of prompts, each targeting a specific FMEA step. This method allows us to structure our

[1]https://github.com/langchain-ai/langchain

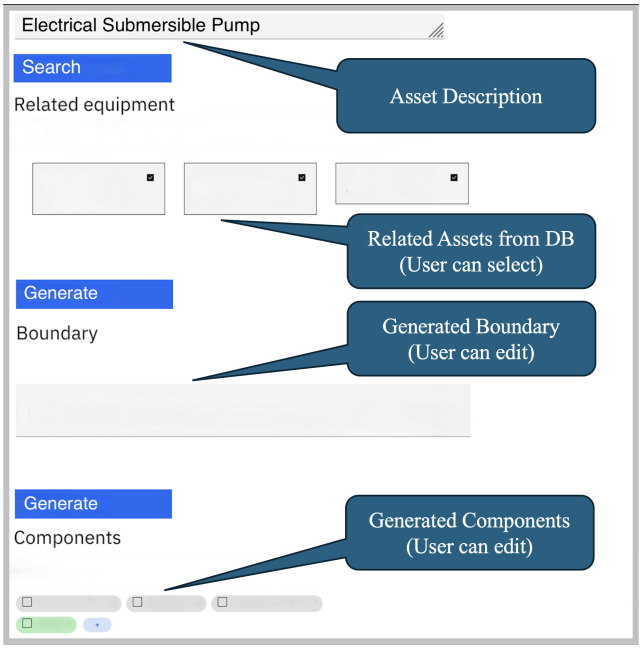

Figure 2: FMEA creation UI showing user supervision of related examples and generated components.

queries to align with the overall hierarchical structure of an FMEA, ensuring that all FMEA elements are addressed. In order to generate the internal structure of each FMEA step, two additional prompting techniques are employed: i) examples retrieved for ICL are lightly preprocessed to adhere to the required structure; ii) output parsers (robust rule based parsers) are used that transform the raw text output from the LLM into structured data formats, such as lists. We also perform data validation using the Python package Pydantic[2] to ensure that the required structural format is adhered to. Note, the structured nature of the generated content enables us to mitigate repetitions, a prevalent issue in language models (Holtzman et al. 2020).

Metrics for Structured Responses Popular LLM evaluation metrics such as ROUGE, perplexity, and BLEU do not consider document structure. Further, metrics such as BLEU and ROUGE, are based on surface text representations (e.g., synonyms are not considered) and use tokens rather than entities for comparisons (c.f. Related Work for further discussion of metrics). In order to accurately assess structure as well as semantic content we introduce a collection of interpretable performance evaluation metrics. These metrics are designed to capture the inherent structure of documents such as FMEAs. Further, by directly considering user concerns such as "What fraction of components did the system generate?", we ensure alignment with application performance.

We propose a group of related metrics Structured Semantic Entity Evaluation (SSEE) for evaluating any structured content that can be represented as a collections of sets, including documents with a tree like structure like FMEAs.

[2]https://docs.pydantic.dev

Entity indicates that the evaluation is focused on entities (e.g., components) rather than tokens. Structured indicates that we consider structure not just content, and semantic because we consider the semantic representation of entities rather than their surface representations. We introduce SSEE Recall in Algorithm 1, SSEE Precision which is the analogue of SSEE Recall for precision (specification is omitted to save space) and SSEE F1 as the harmonic mean of SSEE Precision and SSEE Recall. SSEE Recall and SSEE Precision both consider the similarity of entities in the candidate and gold sets in a shared embedding space. Sorting in Algorithm 1 is used to identify the closest matches between the gold and candidate entities.

Algorithm 1: Structured Semantic Entity Evaluation - Recall

Given: Set of gold entities $\mathcal{E}_{\text{gold}}$, set of candidate entities $\mathcal{E}_{\text{cand}}$, sentence encoder $\mathcal{S}_{\mathcal{E}}$, similarity metric $\mathcal{S}_{\mathcal{M}}$, and similarity threshold $\mathcal{T}$.

Encode the entities of both $\mathcal{E}_{\text{gold}}$ and $\mathcal{E}_{\text{cand}}$ using $\mathcal{S}_{\mathcal{E}}$.
for each entity e_g in $\mathcal{E}_{\text{gold}}$ **do** ▷ Get and sort cands
    $\mathcal{L}_{e_g} = []$ ▷ Candidate matches of e_g
    for each entity e_c in $\mathcal{E}_{\text{cand}}$ **do**
        if $\mathcal{S}_{\mathcal{M}}(e_g, e_c) \geq \mathcal{T}$ **then**
            $\mathcal{L}_{e_g}.append(e_c)$
        end if
    end for
    $\mathcal{L}_{e_g}^{sorted} = \mathcal{L}_{e_g}$ sorted by similarity score with e_g
end for
Let $\mathcal{M}_{\text{cand}} = \emptyset$ and $\mathcal{L}_{\text{gold}} = list(\mathcal{E}_{\text{gold}})$
Let $\mathcal{L}_{\text{gold}}^{sorted} = \mathcal{L}_{\text{gold}}$ sorted by similarity score of $\mathcal{L}_{e_g}^{sorted}[0]$ with e_g
for each entity e_g in $\mathcal{L}_{\text{gold}}^{sorted}$ **do** ▷ Get best match of e_g
    for each entity e_c in $\mathcal{L}_{e_g}^{sorted}$ **do**
        if $e_c \notin \mathcal{M}_{\text{cand}}$ **then**
            $\mathcal{M}_{\text{cand}} = \mathcal{M}_{\text{cand}} \cup \{e_c\}$
            break ▷ Move to next gold entity
        end if
    end for
end for
Recall $= \frac{|\mathcal{M}_{\text{cand}}|}{|\mathcal{E}_{\text{gold}}|}$

In-Context Learning

We introduce a variant of ICL that incorporates user supervision into the example selection methods of Liu et al. (2022); Nori et al. (2023) which we term Supervised Dynamic Few Shot Prompting (SDFSP). A *shot* refers to an input-output pair. For example when generating components, the input of a shot is an asset's name and the output is its component list. SDFSP involves selecting relevant examples from a curated library of over 700 high-quality FMEAs, which includes tens of thousands of failure modes. For a given input, we algorithmically retrieve and rank the most pertinent examples, which are then refined by the user selecting the most relevant ones. The proposed approach combines dense retrieval methods with expert knowledge to select the

most relevant examples for generation. The specific algorithmic approach used for retrieving relevant shots is as follows. Training shot inputs and user queries are encoded in a shared embedding space using a sentence encoder. For training shot inputs this happens offline, whilst for user queries this happens in real-time. A query's k nearest neighbours in the training set are retrieved and ranked using the cosine similarity measure. The retrieved shots are preprocessed and added to the prompt post expert supervision. During evaluation, the impact of shot orderings and counts on the text generation quality is also explored.

Model Ensembling

We propose a novel technique for LLM ensembling that supports the merging of complex outputs into a coherent response. Our confidence in an answer is increased when multiple models and/or prompts converge upon it, as this indicates a consensus among them. This intuition forms the foundation of our approach. We label our technique as Counter-Based Entity Ensembling (CBEE), as it is counter based and uses semantic representations of entities, and specify it in detail in Algorithm 2. CBEE combines outputs from different models and/or prompts by leveraging the structure of their generated content. This approach allows CBEE to surpass the performance of individual models while maintaining coherence in the generated text. As CBBE does not require supervised learning as used in previous approaches (Jiang, Ren, and Lin 2023; Lu et al. 2024), it can be used to cheaply fuse the internal knowledge from many LLMs, whilst also learning from our library of existing FMEAs via ICL. CBBE can be applied to any structured content that can be expressed as a hierarchical arrangement of lists, which encompasses a wide range of formats, including but not limited to documents with a tree-like structure such as FMEAs. CBBE also offers the flexibility to trade-off between recall and precision by adjusting the count threshold (see Algorithm 2). Our implementation of CBBE uses a customised clustering strategy that initialises a new cluster by randomly selecting a free element as a seed, and greedily adding other available elements to the cluster provided they satisfy the similarity requirement relative to all existing elements within the cluster. This process continues until all free elements have been incorporated into a cluster.

Evaluation

We now report on qualitative and quantitative evaluations of our FMEA creation application. We describe detailed empirical experiments for generating the set of maintainable components of an equipment type. The set of maintainable components or failure locations for an industrial asset is a key part of an FMEA indicating the parts of an asset likely to fail. Furthermore, we present valuable feedback on our application gathered from Subject Matter Experts (SMEs). All experiments use greedy decoding and the *all-mpnet-base-v2*[3] sentence encoder.

The input to the component generation step is a short description of the equipment type (e.g. 'Electrical Submersible Pump'). Experiments used our propriety database of FMEAs developed by SMEs and a variety of LLMs: Llama 3 (70B) (Touvron, Martin et al. 2023), Flan-UL2 (20B) (Tay et al. 2023), and Mixtral 8x7B (8x7B) (Jiang et al. 2024a). The database comprises 714 examples. For evaluation purposes we divided the database into train (n=571; 80%), validation (n=71; 10%) and test (n=72; 10%) splits. Examples for ICL are drawn from the training split. Hyperparameters such as the prompts, example selection criteria and output lengths and CBBE hyperparameters were selected based on the validation split.

[3]http://huggingface.co/sentence-transformers

Individual Models

For scale and practicality experiments we use DFSP (with 5 and 15 shots), rather than SDFSP. If all shots don't fit in a model's input limit we drop the least relevant shots. This is primarily an issue when using 15-shots with FLAN-UL2 as it has a receptive field of 2048 tokens (Tay et al. 2023). We consider three different permutations for ICL shots: *Worst Last* - Ordered by most relevant to least relevant shot; *Worst First* - Ordered by least relevant to most relevant shot; and *Worst Middle* which places shots closer to the middle the less relevant they are. For example, for 5-shot, where i indicates the i'th most relevant example, the Worst Last ordering is $[0, 1, 2, 3, 4]$, Worst First ordering is $[4, 3, 2, 1, 0]$, and the Worst Middle ordering is $[0, 2, 4, 3, 1]$. The results of the single model experiments are shown in Table 1.

Individual models performed this task well. Llama 3 is the top performing model for the metrics SSEE Recall (0.605), SSEE F1 (0.587) and ROUGE-1 (0.551), with Flan-UL2 producing the top score for SSEE Precision (0.63). Zero-Shot performance was significantly lower than DFSP performance, emphasizing the need for effective ICL techniques

Algorithm 2: Counter-Based Entity Ensembling

Given: Entity lists $\mathcal{L}_i, \forall i \in \{1, 2, \ldots, n\}$, a sentence encoder $\mathcal{S}_\mathcal{E}$, a similarity metric $\mathcal{S}_\mathcal{M}$, a similarity threshold $\mathcal{T}_{sim}$, a count threshold $\mathcal{T}_{cnt}$, and a clustering algorithm $\mathcal{C}(\mathcal{L}, \mathcal{S}_\mathcal{E}, \mathcal{S}_\mathcal{M}, \mathcal{T}_{sim})$ which clusters the items in $\mathcal{L}$ such that two items a and b are in the same cluster only if $\mathcal{S}_\mathcal{M}\left(\mathcal{S}_\mathcal{E}\left(a\right), \mathcal{S}_\mathcal{E}\left(b\right)\right) \geq \mathcal{T}_{sim}$.

for $i \in \{1, 2, \ldots, n\}$ **do** ▷ Individual lists
  $\mathcal{L}_i^{cluster} = \mathcal{C}(\mathcal{L}_i, \mathcal{S}_\mathcal{E}, \mathcal{S}_\mathcal{M}, \mathcal{T}_{sim})$
  Let $\mathcal{L}_i^u = []$ ▷ Keep an element from each cluster
  for each cluster cls in $\mathcal{L}_i^{cluster}$ **do**
    $\mathcal{L}_i^u.append(e), e$ is an element of cls.
  end for
end for
Let $\mathcal{L} = \mathcal{L}_1^u + \mathcal{L}_2^u + \ldots + \mathcal{L}_n^u$
$\mathcal{L}^{cluster} = \mathcal{C}(\mathcal{L}, \mathcal{S}_\mathcal{E}, \mathcal{S}_\mathcal{M}, \mathcal{T}_{sim})$
Let $\mathcal{L}^{merged} = []$
for each cluster cls in $\mathcal{L}^{cluster}$ **do**
  if $|cls| \geq \mathcal{T}_{cnt}$ **then**
    $\mathcal{L}^{merged}.append(e), e$ is an element of cls.
  end if
end for
return $\mathcal{L}^{merged}$

Model	Method	Shot Order	Shots	SSEE Recall	SSEE Precision	SSEE F1	ROUGE-1
Flan-UL2	Zero-shot	NA	0	0.047	0.218	0.077	0.102
Flan-UL2	DFSP	Worst Last	5	0.411	0.614	0.492	0.457
Flan-UL2	DFSP	Worst Middle	5	0.423	0.599	0.495	0.457
Flan-UL2	DFSP	Worst First	5	0.416	**0.630***	0.501	0.466
Flan-UL2	DFSP	Worst Last	15	0.470	0.604	0.529	0.502
Flan-UL2	DFSP	Worst Middle	15	**0.482**	0.604	**0.536**	**0.515**
Flan-UL2	DFSP	Worst First	15	0.471	0.604	0.529	0.514
Llama 3	Zero-shot	NA	0	0.176	0.165	0.171	0.180
Llama 3	DFSP	Worst Last	5	0.499	0.526	0.512	0.481
Llama 3	DFSP	Worst Middle	5	0.516	0.536	0.526	0.488
Llama 3	DFSP	Worst First	5	0.521	0.551	0.535	0.494
Llama 3	DFSP	Worst Last	15	**0.605***	0.569	**0.587***	0.548
Llama 3	DFSP	Worst Middle	15	0.593	**0.578**	0.586	**0.551***
Llama 3	DFSP	Worst First	15	0.583	0.561	0.571	0.523
Mixtral 8x7B	Zero-shot	NA	0	0.135	0.114	0.123	0.175
Mixtral 8x7B	DFSP	Worst Last	5	0.438	0.544	0.485	0.409
Mixtral 8x7B	DFSP	Worst Middle	5	0.374	**0.548**	0.445	0.349
Mixtral 8x7B	DFSP	Worst First	5	0.449	0.506	0.476	0.397
Mixtral 8x7B	DFSP	Worst Last	15	0.452	0.472	0.462	0.402
Mixtral 8x7B	DFSP	Worst Middle	15	0.448	0.460	0.454	0.375
Mixtral 8x7B	DFSP	Worst First	15	**0.513**	0.488	**0.500**	**0.443**

Table 1: Results of Individual Model Evaluation for FMEA Component Generation.

with this application. Again Llama 3 was the best performing individual model in zero-shot, with a ROUGE-1 score of 0.18, whereas Flan-UL2 performed significantly worse than the other models in this setting. In general, including 15 shots produced better results than 5 shots, except for SSEE Precision, perhaps due to the inclusion of some irrelevant content in the extra shots that the models sometimes fail to filter. In terms of shot-ordering, no clear patterns were observed that favoured a specific arrangement. However, various shot permutations provide valuable inputs for ensembling approaches. The evaluation process also validated the generated content's structure, as the gold answers are structured and the SSEE metrics are responsive to structure as well as text. This underscores the efficacy of our approach for producing both accurate and structured responses. In summary, individual models demonstrated strong performance, and incorporating more shots generally improved results, except for SSEE Precision.

Ensembles

We also evaluate the CBEE ensembling approach on the test split using the top three and the top five performing individual model, shot count and shot ordering triples on the validation split for the metric SSEE F1. The top five triples on the validation split were Llama 3 Worst Middle, Flan-UL2 Worst First, Llama 3 Worst Last, Flan-UL2 Worst Middle and Flan-UL2 Worst First, all with 15-shots. We report results for CBEE with a match threshold equal to 0.95, and vary the count threshold across a range of values to explore the balance between precision and recall. The results of the ensemble experiments are shown in Table 2. The application of ensemble methods leads to improvements across all reported metrics. Specifically, the CBEE approach with a count threshold of 1 exhibited an increase in recall of more than 10%, reaching 0.669 compared to 0.605 for any single response. CBEE with a count threshold of 5 achieved a remarkable precision of 0.91, which is particularly impressive given the difficulty of the task. The CBEE ensembling approach with count threshold set to 2 yielded the best results in metrics that balance both precision and recall (ROUGE-1 and SSEE F1). When comparing ensembles of the Best 5 to the Best 3, in general the former showed better recall, while the latter had better precision. This highlights the importance of considering both recall and precision when evaluating the performance of ensembling approaches. In summary, the CBEE ensemble method has been successful in enhancing the performance of this task, as evidenced by the improvements in recall and precision achieved.

User Feedback

We gathered user feedback on our application via survey and individual interviews. We demonstrated our application to an audience of SMEs at the Society for Maintenance & Reliability Professionals[4] annual conference. Subsequent polling revealed an overwhelming positive response to the adoption of the tool if available, as well as to the balance we struck between automation and supervision. Further, we engaged an SME having decades of experience as a reliability engineer to use our interactive system to generate 100 new FMEAs for various equipment types. The expert took on average four hours to interactively generate, review and correct a new FMEA. Even though the time for generating a new document of good quality is still standing at several person-hours, this represents a large improvement over the several person days traditionally spent on this task (Cooper 2015). Other feedback highlighted the importance of low-latency for interactivity and a preference for concise responses.

[4]https://smrp.org/

Model(s)	Ensemble	Method	Threshold	SSEE Recall	SSEE Precision	SSEE F1	ROUGE-1
Flan-UL2	Best 3	CBEE	2	0.468	0.630	0.532	0.504
Flan-UL2	Best 5	CBEE	2	0.494	0.609	0.545	0.513
Llama 3	Best 3	CBEE	2	0.578	0.624	0.600	0.564
Llama 3	Best 5	CBEE	2	0.629	0.536	0.579	0.528
Mixtral 8x7B	Best 3	CBEE	2	0.403	0.679	0.526	0.450
Mixtral 8x7B	Best 5	CBEE	2	0.556	0.534	0.545	0.496
Many	Best 3	CBEE	1	0.663	0.484	0.560	0.504
Many	Best 5	CBEE	1	**0.669**	0.457	0.543	0.490
Many	Best 3	CBEE	2	0.558	0.658	**0.604**	0.559
Many	Best 5	CBEE	2	0.610	0.584	0.597	**0.567**
Many	Best 3	CBEE	3	0.365	0.877	0.515	0.427
Many	Best 5	CBEE	5	0.309	**0.910**	0.461	0.370

Table 2: Results of Ensemble Evaluation for FMEA Component Generation.

Deployment Considerations

Deployment of our system targets complex enterprise settings where users may not be familiar with the technology of LLMs. On the system side, even though preferred software components are rapidly evolving, our application needs to support deployment environments both in the cloud and on-premise.

We address these system requirements with a loosely coupled container-based architecture (c.f., Figure 3). Users interact with a web UI that communicates with a web service with endpoints for each step in the FMEA generation process. Both the UI and web service are packaged and deployed separately as stateless applications, enabling horizontal scaling to support the required workload. The web service implements the in-context learning, structured generation and ensemble approaches used in our application. The service receives input from clients and in turn retrieves similar examples from the vector database, prompts the LLM service(s), and returns the parsed response to the user. The decoupling of the service and interface allows for easier integration with different systems, including an enterprise asset management solution and a suite of tests.

This design delegates LLM inferencing and vector database functionality to external services for flexibility and scalability. For example, customer deployments can include additional vector databases with their own FMEA collections. Also, deployments can flexibly choose their preferred LLM provider or model. While this flexibility is required in modern enterprise deployment it proves to be a challenge given the vast array of providers in the market and a lack of standardised APIs. Our system relies on both in-house and community driven[5] abstractions to interface with the back-end services. This allows users to transparently choose between back-end technologies.

Back-end inferencing endpoints by necessity limit the number of concurrent requests a caller can make, which directly effects the response time users experience. Our web service decomposes LLM calls in the pipeline to take maximum advantage of the available concurrency.

In summary we deploy our application via stateless containers that communicate over HTTP(s) to promote scalability and interoperability with different systems including an enterprise asset management solution.

[5]https://github.com/langchain-ai/langchain

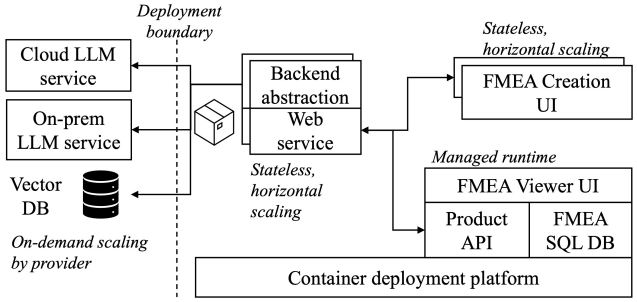

Figure 3: Architecture of the container based system deployment with its boundary and interface to external services.

Conclusions

This paper presents an application for creating structured FMEA documents using LLMs. Primary contributions include an ensemble technique for merging structured LLM outputs, and metrics for validating structured LLM responses. Quantitative and qualitative evaluations confirm the efficacy of our approach in providing productivity enhancement in a specialised quality sensitive domain. An SSEE precision of 0.91 is a specific highlight of the empirical study. We hope that the proposed solution will prove beneficial to others in generating and evaluating structured content through LLMs. Future efforts will focus on finalising deployment to end-users as well as investigating alternative approaches aimed at enhancing content quality, such as synthetic data generation and task-specific fine-tuning. In summary, this paper introduces a labour saving application of LLMs using novel approaches in structured document generation and evaluation, supported by empirical and qualitative evidence, and a clear path to product deployment.

References

Brown, T.; Mann, B.; et al. 2020. Language Models are Few-Shot Learners. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Informa-*

tion Processing Systems, volume 33, 1877–1901. Curran Associates, Inc.

Béchard, P.; and Ayala, O. M. 2024. Reducing hallucination in structured outputs via Retrieval-Augmented Generation. arXiv:2404.08189.

Cooper, H. C. 2015. Capture all critical failure modes into FMEA in half the time with a simple decomposition table (Actual case study savings = $4,206,000$). *InRAMS*, $1--6$.

Holtzman, A.; Buys, J.; Du, L.; Forbes, M.; and Choi, Y. 2020. The Curious Case of Neural Text Degeneration. arXiv:1904.09751.

Jelinek, F.; Mercer, R. L.; Bahl, L. R.; and Baker, J. K. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1): S63–S63.

Jiang, A. Q.; Sablayrolles, A.; Roux, A.; Mensch, A.; Savary, B.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Hanna, E. B.; Bressand, F.; Lengyel, G.; Bour, G.; Lample, G.; Lavaud, L. R.; Saulnier, L.; Lachaux, M.-A.; Stock, P.; Subramanian, S.; Yang, S.; Antoniak, S.; Scao, T. L.; Gervet, T.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2024a. Mixtral of Experts. arXiv:2401.04088.

Jiang, D.; Ren, X.; and Lin, B. Y. 2023. LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 14165–14178. Toronto, Canada: Association for Computational Linguistics.

Jiang, J.; Wang, F.; Shen, J.; Kim, S.; and Kim, S. 2024b. A Survey on Large Language Models for Code Generation. *arXiv preprint arXiv:2406.00515*.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81. Barcelona, Spain: Association for Computational Linguistics.

Liu, J.; Shen, D.; Zhang, Y.; Dolan, B.; Carin, L.; and Chen, W. 2022. What Makes Good In-Context Examples for GPT-3? In Agirre, E.; Apidianaki, M.; and Vulić, I., eds., *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, 100–114. Dublin, Ireland and Online: Association for Computational Linguistics.

Lu, K.; Yuan, H.; Lin, R.; Lin, J.; Yuan, Z.; Zhou, C.; and Zhou, J. 2024. Routing to the Expert: Efficient Reward-guided Ensemble of Large Language Models. In Duh, K.; Gomez, H.; and Bethard, S., eds., *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 1964–1974. Mexico City, Mexico: Association for Computational Linguistics.

Nori, H.; Lee, Y. T.; Zhang, S.; Carignan, D.; Edgar, R.; Fusi, N.; King, N.; Larson, J.; Li, Y.; Liu, W.; Luo, R.; McKinney, S. M.; Ness, R. O.; Poon, H.; Qin, T.; Usuyama, N.; White, C.; and Horvitz, E. 2023. Can Generalist Foundation Models Outcompete Special-Purpose Tuning? Case Study in Medicine. arXiv:2311.16452.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, 311–318. USA: Association for Computational Linguistics.

Pezeshkpour, P.; and Hruschka, E. 2023. Large Language Models Sensitivity to The Order of Options in Multiple-Choice Questions. arXiv:2308.11483.

Rausand, M.; and Høyland, A. 2004. *System Reliability Theory: Models, Statistical Methods, and Applications (2nd ed.)*. Wiley.

Tay, Y.; Dehghani, M.; Tran, V. Q.; Garcia, X.; Wei, J.; Wang, X.; Chung, H. W.; Shakeri, S.; Bahri, D.; Schuster, T.; Zheng, H. S.; Zhou, D.; Houlsby, N.; and Metzler, D. 2023. UL2: Unifying Language Learning Paradigms. arXiv:2205.05131.

Touvron, H.; Martin, L.; et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; Narang, S.; Chowdhery, A.; and Zhou, D. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. arXiv:2203.11171.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E. H.; Le, Q. V.; and Zhou, D. 2024. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NEURIPS '22. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781713871088.

Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K. Q.; and Artzi, Y. 2020. BERTScore: Evaluating Text Generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Zheng, C.; Zhou, H.; Meng, F.; Zhou, J.; and Huang, M. 2023. Large Language Models Are Not Robust Multiple Choice Selectors. arXiv:2309.03882.