RAGulator: Lightweight Out-of-Context Detectors for Grounded Text Generation

Ian Poey **Jiajun Liu** **Qishuai Zhong**
OCBC AI Lab
{ianpoey, qishuaizhong}@ocbc.com

Abstract

Real-time identification of out-of-context outputs from large language models (LLMs) is crucial for enterprises to safely adopt retrieval augmented generation (RAG) systems. In this work, we develop lightweight models capable of detecting when LLM-generated text deviates from retrieved source documents semantically. We compare their performance against open-source alternatives on data from credit policy and sustainability reports used in the banking industry. The fine-tuned DeBERTa model stands out for its superior performance, speed, and simplicity, as it requires no additional preprocessing or feature engineering. While recent research often prioritises state-of-the-art accuracy through fine-tuned generative LLMs and complex training pipelines, we demonstrate how detection models are deployed efficiently with high speed and minimal resource usage.

1 Introduction

In enterprise settings, Generative AI has received widespread adoption as a tool to uplift employees' productivity (Brachman et al., 2024). Enterprises require a high degree of factuality of generated answers in popular language tasks, such as question-answering (QA) and summarisation.

Retrieval Augmented Generation (RAG) (Lewis et al., 2020) allows users to interact with large language models (LLMs) for industry-specific knowledge spanning multiple use cases and document types. It is adaptable to tasks requiring evolving knowledge, such as synthesising information from the latest industry news. However, a key limitation of RAG is the generation of outputs that are observed to be out-of-context. This is related to hallucination of LLMs, manifesting itself in inconsistent or fabricated claims (Huang et al., 2025) that can be subtle and phrased confidently even if factually incorrect (Li et al., 2024). For highly sensitive working environments such as financial institutions, the inability to ensure faithful LLM output can be one of the biggest limitations to widespread adoption of LLM-powered applications (Maple et al., 2024).

To address this barrier to enterprise adoption, in this study we narrow the scope of "hallucination" and focus only on hallucinations that render the LLM response semantically inconsistent with the provided context. This is commonly known as *faithfulness hallucination* (Huang et al., 2025; Es et al., 2024; Saad-Falcon et al., 2024), *contextual hallucination* (Chuang et al., 2024), or a lack of *grounded factuality* (Bespoke Labs) or *support* (Belyi et al., 2025); to aid business understanding, we adopt the term *out-of-context* (OOC). We consider any LLM-generated response to a RAG prompt as semantically OOC if any part of the response is ungrounded based on the retrieved context alone, even if it is otherwise factual according to world knowledge. In contrast, an *in-context* response is one where every claim embedded in the response can be inferred solely from the retrieved context.

Approaches to mitigate OOC generation can be broadly classified into 2 types – black-box and grey-box methods. **Black-box methods** employ strong generative LLMs to assess if a candidate answer is OOC, and these LLMs are usually augmented with various prompting or fine-tuning techniques. Examples of black-box methods include RARR (Gao et al., 2023), WikiChat (Semnani et al., 2023), FreshPrompt (Vu et al., 2024), SelfCheckGPT (Manakul et al., 2023), RAGAS (Es et al., 2024), ChainPoll (Friel and Sanyal, 2023), and Lynx (Ravi et al., 2024). However, black-box approaches that rely on strong closed-source LLM judges are less suitable for enterprises that are constrained by budget and/or data privacy requirements. **Grey-box methods** are alternatives which aim to detect OOC generation through a proxy metric or model. Grey-box approaches either assess

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 1057–1071

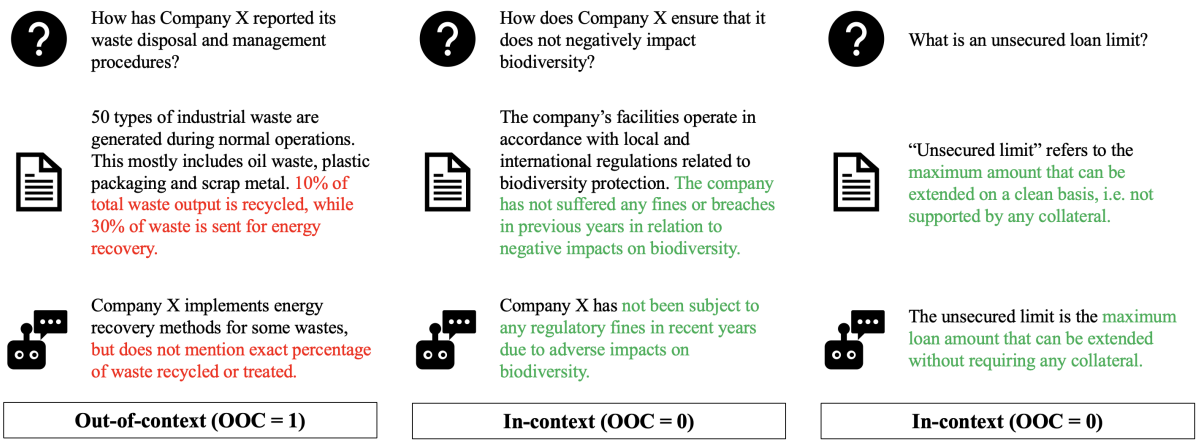

Figure 1: Example LLM responses from RAG systems in the banking industry.

final/hidden LLM states, such as FLARE (Jiang et al., 2023) and Lookback Lens (Chuang et al., 2024), or use a score computed by an independent discriminative model of lower complexity, such as SummaC (Laban et al., 2022), AlignScore (Zha et al., 2023), HHEM (Mendelevitch et al., 2024), and Luna (Belyi et al., 2025).

In this work, we propose RAGulator, a series of lightweight OOC detectors for RAG applications. We create a training dataset which simulates both OOC and in-context RAG prompts, from a range of public datasets originally constructed for various traditional NLP tasks. Furthermore, we compare 2 types of grey-box discriminative models – fine-tuned BERT-based classifiers and ensemble meta-classifiers trained on numerical features derived from text. Generative labelling with an LLM annotator is employed where necessary to adapt the training dataset for BERT-based classifier fine-tuning. We show that while a large LLM can show good agreement with human annotation in labelling data for BERT classifier fine-tuning, our predictive models outperform the same LLM in a zero-shot setting on the enterprise OOC detection task, while also surpassing other open-source detectors in most cases. The results highlight the need for specialised models for OOC detection in enterprise RAG.

2 Method

2.1 Problem formulation

We train a lightweight grey-box discriminator to detect semantically OOC LLM-generated sentences from a RAG system. Figure 1 illustrates examples of OOC and in-context sentences extracted from RAG systems in the banking industry. In the OOC example (left), the LLM fails to find the answer within retrieved documents, and the generated statement is inconsistent with the context. Whereas, in-context statements are clearly grounded in the documents regardless of whether their relevance to the question is low (middle) or high (right).

We formulate the problem of semantic OOC closely following the setup by Tang et al. (2024). Any RAG prompt $Prompt(\mathcal{D}, x)$ to an LLM contains an associated set of retrieved documents $\mathcal{D} = \{D_1, ..., D_{|\mathcal{D}|}\}$ and a question x, to which the LLM responds with text that can be broken into an unordered set of sentences $\mathbf{c} = \{c_1, ..., c_{|\mathbf{c}|}\}$. We treat each sentence c_i as an independent candidate claim to be verified. To detect OOC candidates, we define a discriminator

$$M(\mathcal{D}, c_i) \in \{0, 1\}$$

that classifies each candidate as `out-of-context`, 1, or `in-context`, 0, according to the set of retrieved documents $\mathcal{D}$.

Certain model architectures (e.g. BERT) face limitations on the amount of text that can be accepted by the model in one pass. This would require the set of documents $\mathcal{D}$ to be arbitrarily re-split into J text chunks $\{D_1, ..., D_J\}$ of varying sizes not more than the model-defined limit. In this case, the overall discriminator can be defined as

$$\min_j M(\mathcal{D}_j, c_i) \in \{0, 1\}$$

implying that if and only if the claim c_i is OOC with respect to every text chunk D_j, then it can be considered OOC overall.

2.2 Dataset curation

We construct a dataset by adapting publicly available datasets, sampling and preprocessing them to simulate LLM-generated sentences and RAG-retrieved contexts of various lengths. The goal is to

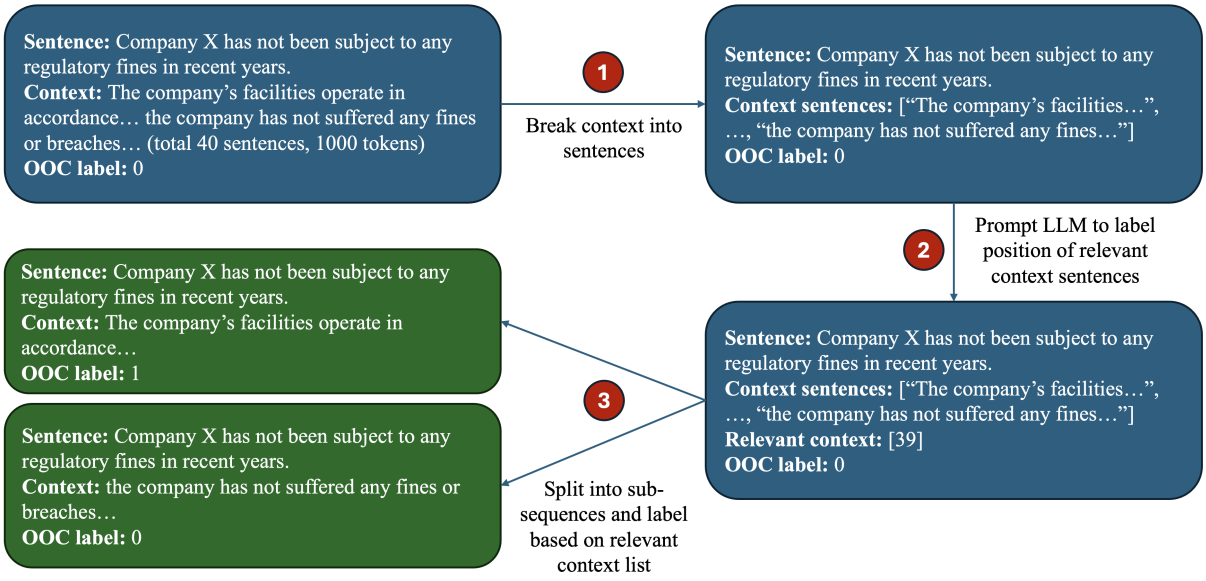

Figure 2: Illustrated flow of the generative labelling process.

create a dataset with sentence-context pairs. This curation only involves sentence tokenisation and random sampling. Datasets belonging to summarisation (extractive and abstractive) and semantic textual similarity tasks are selected for adaptation.

Summarisation task datasets. Originally each row in the dataset is an abstract-article pair. Preprocessing is done by randomly pairing abstracts with unrelated articles to create OOC pairs, then applying sentence tokenisation to the abstracts to create one example for each abstract sentence. Both extractive and abstractive summarisation datasets are adapted, consisting of BBC (Greene and Cunningham, 2006), CNN/Daily Mail (version 3.0.0) (Nallapati et al., 2016), and PubMed (Cohan et al., 2018) datasets.

Semantic textual similarity (STS) datasets. Originally each row in the dataset is a pair of sentences, with a label indicating if the pair of sentences are similar. Preprocessing is done by inserting random sentences from the datasets to one of the sentences in the pair to simulate a long "context". The original labels are mapped to our definitions to indicate if the pair is an OOC pair. The adapted datasets consist of the Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005) and Stanford Natural Language Inference (SNLI, version 1.0) (Bowman et al., 2015) datasets.

For MRPC, the semantically-equivalent indicator is mapped to `in-context` if present and `out-of-context` if absent. For SNLI, labels provided by human annotators are originally in the set {`entailment`, `contradiction`, `neutral`}. We ignore sentence pairs labelled `neutral` and those without unanimous agreement, while mapping the labels `entailment` to `in-context` and `contradiction` to `out-of-context`.

In our eventual curated dataset, we ensure each simulated sentence contains between 5-100 tokens, and that each simulated context contains between 100-5000 tokens. We sample only from author-provided training and test splits to form training and holdout evaluation data respectively.

2.3 Generative labelling

Standard BERT models have a 512-token limit. In this study, generative labelling is employed for data used in fine-tuning to ensure compatibility with a wide range of BERT model variants.

For sentence-context examples derived from summarisation datasets and labelled as in-context (≈34% of dataset), the LLM-"generated" sentence is relevant to at least one part of the context, but the exact position is unknown. As shown in Figure 2, we utilise Llama-3.1-70B-Instruct (Grattafiori et al., 2024) to label each sentence by prompting it to return the positions of context sentences that are relevant. With this output, we split the sentence-context example into windows of tokenised sub-sequences (i.e. sentence + context) that are each no longer than 512 tokens, and label the sub-sequences based on whether the sub-sequence contains a relevant context sentence.

2.4 Feature engineering

We incorporate classic machine translation metrics to compare response (candidate) and context (reference) alongside semantic similarity features derived from distance metrics computed using encoder-only models. Preliminary analysis reveals that the feature distributions for in-context LLM-generated sentences differ significantly from those of OOC sentences. This section provides a brief overview of the implementation of these features, with additional details in Section A.

Precision score. We calculate this feature as the fraction of words in the response that also appear in the context (Melamed et al., 2003). We apply the precision score after text preprocessing and word deduplication.

Unigram and bigram perplexities. We construct a probability distribution (i.e. dictionary of token frequencies) from the context's unigram and bigram tokens after text preprocessing, then calculate sum negative log likelihood and normalise by the respective number of candidate n-grams.

Maximum embedding similarity score. We compute the maximum pairwise similarity between embeddings of the response and each context sentence using the pretrained bge-small-en-v1.5 model (Xiao et al., 2024).

Maximum reranker relevance score. We compute the maximum relevance score between the response and each context sentence using the pretrained bge-reranker-base model.

2.5 Models

We train two meta-classifiers of LightGBM and Random Forest architectures respectively. These are selected due to their widespread adoption and reliability in enterprise settings. In parallel, we leverage the generative labelling technique to train two BERT-based classifiers: deberta-v3-large[1] initialised with default pretrained weights, and xlm-roberta-large[2] initialised from the bge-reranker-large checkpoint, effectively fine-tuning a reranker for OOC detection. Training details for all models are found in Section B.

[1] huggingface.co/microsoft/deberta-v3-large
[2] huggingface.co/BAAI/bge-reranker-large

		Training		Evaluation	
Type	Dataset	Rows	% OOC	Rows	% OOC
Summ.	BBC	10000	50.0	-	-
	CNN/Daily Mail	20000	50.0	-	-
	PubMed	20000	50.0	4000	50.0
STS	MRPC	3904	32.1	1697	33.3
	SNLI	20000	50.0	4000	50.0
Private	ESG	-	-	386	44.6
	CP	-	-	184	40.2
Total		**73904**	**49.1**	**10267**	**46.9**

Table 1: Dataset statistics for training and evaluation. The number of examples and out-of-context proportion is stated for each data subset.

3 Evaluation

3.1 Public data

We evaluate RAGulator models on an **in-distribution** holdout split of our simulated RAG dataset, comprising 9,697 rows sampled from the PubMed, MRPC, and SNLI subsets.

3.2 Industry data

We also prepare an **out-of-distribution** collection of 570 RAG responses that are gathered and hand-annotated for evaluation. This private dataset includes 386 responses related to sustainability-themed documents analysed by corporate banking relationship managers ("ESG"), and 184 responses associated with credit policy documents used by bank officers in loan approval processes ("CP"). The examples illustrated in Figure 1 are adapted from anonymised and truncated excerpts of this private dataset to preserve confidentiality.

Table 1 reports the composition per-source and the proportion of OOC labels in training and evaluation datasets.

3.3 Model inference

When running inference with BERT-based classifiers, we predict on each of the tokenised sub-sequences of the original sentence-context pair. We then aggregate model predictions to obtain a single prediction at the sentence level. We take the minimum probability across all sub-sequences as the overall OOC probability; as long as one of the generated sub-sequences is in-context, we consider the entire generated sentence as in-context.

For evaluation, we use Llama-3.1-70B-Instruct hosted on four H100 GPUs for black-box methods, and transformer-based models for grey-box methods hosted on one V100 GPU.

Framework/Model	Test - Overall			Test - ESG			Test - CP			Speed
	AUROC	AUPRC	F1	AUROC	AUPRC	F1	AUROC	AUPRC	F1	(sentences/s)
Zero-shot prompt (baseline)	89.3	91.0	88.7	81.7	85.5	78.9	75.7	82.8	68.4	3.03
RAGAS Faithfulness	93.7	92.9	90.6	82.0	85.5	81.8	86.9	78.8	**82.7**	0.08
SelfCheckGPT-Prompt	-	-	-	64.3	69.6	63.2	78.1	78.5	74.1	0.41
AlignScore-large	92.3	91.7	84.0	82.5	77.4	75.6	80.9	68.9	72.8	9.99
MiniCheck-deberta-v3-large	91.4	87.4	84.1	84.8	81.0	78.0	84.2	83.5	74.5	9.45
Bespoke-MiniCheck-7B	95.8	95.4	89.0	86.9	83.8	79.3	89.8	**87.3**	81.4	9.82
RAGulator-Random-Forest	94.5	94.6	86.7	**89.7**	87.4	**81.9**	89.6	85.3	79.7	2.01
RAGulator-LightGBM	95.5	95.4	87.4	89.0	**87.8**	81.8	89.7	81.7	77.9	2.03
RAGulator-deberta-v3-large	**98.2**	**98.3**	**93.5**	87.0	84.6	77.6	**90.1**	86.5	80.3	19.08
RAGulator-xlm-roberta-large	97.1	97.2	91.1	86.2	83.7	77.6	88.4	86.3	78.6	26.09

Table 2: Results on overall evaluation dataset and private in-house ESG and CP subsets. The best and second-best scores are shown in **bold** and underlined respectively. Cells marked with a dash indicate that the metric is not applicable to the method with explanation in Section E.

Framework/Model	Test - PubMed			Test - MRPC			Test - SNLI		
	AUROC	AUPRC	F1	AUROC	AUPRC	F1	AUROC	AUPRC	F1
Zero-shot prompt (baseline)	94.4	95.0	94.7	65.9	62.1	54.3	93.3	94.8	93.4
RAGAS Faithfulness	94.6	97.1	93.9	67.0	82.4	75.1	96.9	97.8	95.7
SelfCheckGPT-Prompt	-	-	-	-	-	-	-	-	-
AlignScore-large	86.2	80.0	82.0	70.2	50.5	57.1	**99.8**	**99.8**	**98.5**
MiniCheck-deberta-v3-large	96.3	95.3	90.9	72.8	55.2	59.0	97.9	98.2	93.5
Bespoke-MiniCheck-7B	99.1	98.7	96.4	74.3	62.8	60.0	99.2	99.3	96.0
RAGulator-Random-Forest	99.4	99.2	**97.6**	76.0	65.3	59.7	94.5	94.8	87.3
RAGulator-LightGBM	99.4	99.0	97.4	76.5	65.2	60.2	94.8	95.4	87.3
RAGulator-deberta-v3-large	**99.5**	**99.5**	97.4	**93.7**	**88.2**	**81.0**	99.5	99.6	98.0
RAGulator-xlm-roberta-large	99.0	98.7	95.4	88.9	80.5	74.9	99.4	99.5	96.4

Table 3: Results on each public data subset (PubMed, MRPC, and SNLI). The best and second-best scores are shown in **bold** and underlined respectively. Cells marked with a dash indicate that the metric is not applicable to the method with explanation in Section E.

Prompting method	Accuracy (%)	κ (%)
0-shot	98.5	77.8
0-shot, CoT	98.3	78.6
5-shot	98.6	80.5
5-shot, CoT	98.7	83.6

Table 4: Measure of inter-rater agreement between human annotation and each Llama-3.1 prompting method.

4 Results

4.1 Generative vs. human labelling

We verify the effectiveness of generative labelling by experimenting with several prompting methods on a sample of 58 sentence-context pair examples (20 BBC, 20 CNN/Daily Mail, 18 PubMed), comprising 2,696 context sentences. Table 4 shows the inter-rater agreement between human annotation and each Llama-3.1 annotator prompted with different templates, such as few-shot prompting, chain-of-thought prompting (CoT) (Kojima et al., 2022), or both (Wei et al., 2022). Details on prompting are found in Section C.

We find that the use of five-shot and CoT increases the reliability of generative labelling compared to human annotation, evidenced by the increase in Cohen's kappa (κ). Nevertheless, direct zero-shot prompting is already in substantial agreement with human observation. We adopt zero-shot for generative labelling as this method is the fastest, utilises the fewest tokens, and least likely to deviate in terms of desired LLM output structure.

4.2 Model performance on evaluation datasets

Tables 2 and 3 present the evaluation results on our private ESG and CP subsets, and on the public subsets (PubMed, MRPC, and SNLI), respectively. We report area-under-curve (AUC) scores as well as F1 scores computed after threshold tuning.

Performance against baseline. We compare our models with direct zero-shot prompting of Llama-3.1 as a baseline. Details on prompting and extracted OOC probabilities that allow for the reporting of AUC scores are given in Section D. Across the evaluation dataset, our models consistently out-

perform the baseline in AUC metrics. The observed performance gap is largest in the CP dataset, with RAGulator-deberta-v3-large showing a 19% increase in AUROC and a 17% increase in F1 score, despite being significantly smaller than Llama-3.1. BERT-based RAGulator models are at least 630% faster than baseline, while the meta-classifiers are up to 34% slower. However, the 70B LLM requires significantly more VRAM compared to both meta-classifiers, even with underlying models needed for feature generation.

Performance against other detectors. We compare our models with other black-box methods and open-source grey-box detectors (details in Section E). RAGulator models outperform evaluated detectors in most metrics, with the exception of Bespoke-MiniCheck-7B that has a slight edge in the CP dataset. Despite scoring 0.9% lower in AUPRC and 1.4% lower in F1 score, RAGulator-deberta-v3-large exhibits overall competitive performance as an encoder-only model with 5.6% of the parameters and 194% of the speed. While RAGAS Faithfulness obtains the highest F1 score for the CP dataset, its speed is more than 30 times slower than baseline and is likely impractical for inference in real-time or at scale. On the ESG subset, the meta-classifiers surpass other detectors across all metrics regardless of size, and are also faster than black-box detectors.

5 Discussion

5.1 Model deployment in enterprise

Enterprise LLM usage patterns motivate the need for OOC detectors that optimise the tradeoff between accuracy and efficiency for low-latency deployment at scale. Our LLM cluster infrastructure receives over 2 million requests per month, totalling 2.2 billion input tokens and generating over 190 million output tokens. The most frequently used application involving RAG sustains a peak load of up to 100 requests per minute. To meet demands, we deploy RAGulator-deberta-v3-large[3,4] as a real-time guardrail for RAG requests in the enterprise, using a GPU-enabled serving framework with load balancing and autoscaling capabilities, enabling high availability and throughput. To further maximise model utilisation and reduce latency, we implement mini-batching and dynamic mapping

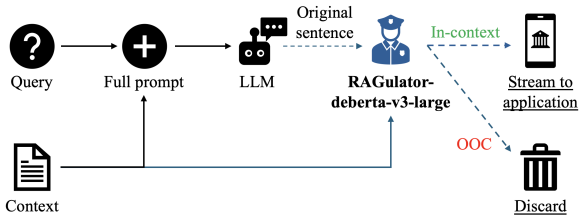

Figure 3: Illustrated setup for OOC detection with a RAGulator BERT-based classifier in real-time RAG workflows, where LLM-generated sentences are streamed to the guardrail at sentence-level.

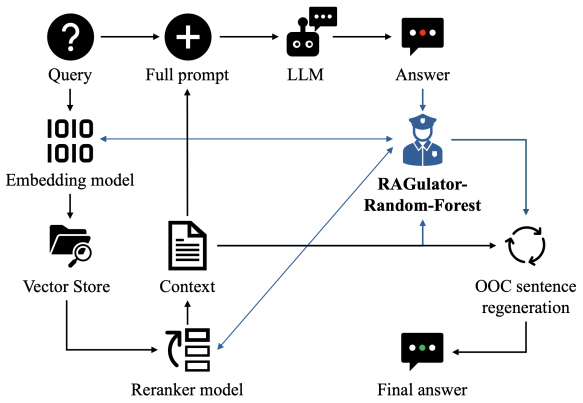

Figure 4: Illustrated setup for OOC detection with a RAGulator meta-classifier in batch RAG workflows, where the LLM-generated answer is sent to the guardrail to mark OOC sentences for regeneration.

of sentence-context pairs aggregated across concurrent requests containing multiple inputs. Figure 3 illustrates this setup, which is most effective for use cases that require real-time RAG responses, such as in AI copilot applications. Sentences are streamed to RAGulator with accompanying context as soon as each sentence is fully generated, and passed to the user-facing application only if it is marked as in-context. By serving OOC detection with a small specialised model, the real-time requirement is satisfied with low resource usage.

For RAGulator meta-classifiers, the most computationally intensive step in both training and inference is the calculation of embedding similarity and reranker relevance scores during feature engineering. These metrics are precomputed in the context retrieval phase of the RAG pipeline, enabling reuse of the same embedding and reranking models within the feature engineering process. By aligning feature computation with existing GPU-enabled retrieval infrastructure, no additional GPU resource needs to be consumed by OOC detection. In practice, we deploy RAGulator-Random-

[3]Code available at github.com/ipoeyke/RAGulator

[4]Model weights available at huggingface.co/ipoeyke/ragulator-deberta-v3-large

Forest in batch workflows for enterprise RAG, asynchronously using embedding and reranking services that are already active. Figure 4 illustrates this setup, which is most effective for use cases where meta-classifiers outperform BERT-based models and where real-time prediction is not required.

In both real-time and batch scenarios, sentences marked as OOC can either be excluded from the final answer, or trigger re-retrieval and sentence regeneration steps as implemented in active retrieval paradigms such as FLARE (Jiang et al., 2023).

5.2 Model evaluation in enterprise

We introduce an out-of-distribution evaluation dataset not only to prevent data leakage, but also to assess if a given OOC model is suitable for specific use cases in the organisation. We ensure that these goals are met by collecting private data from enterprise domains that differ from the training distribution, thereby avoiding overlap between training and test sets. While public benchmarks are useful for rapidly surveying the performance of open-source OOC detection models, their results can diverge from real-world behaviour in enterprise settings. As such, model selection should be guided by the specific requirements and measured performance in the intended use case.

6 Conclusion

In this study, we demonstrate a low-resource pipeline for data gathering and training of small discriminative models to detect LLM-generated texts that are semantically out-of-context. We show that these models not only outperform zero-shot detection and black-box judges powered by Llama-3.1-70B-Instruct, a much larger generative LLM, but also surpass several popular grey-box detectors on enterprise data. RAGulator-deberta-v3-large achieves an optimal balance between performance and efficiency, is over 6 times faster than zero-shot prompting, and does not require additional text preprocessing or feature engineering. Although emerging work has focused on LLM-as-a-judge methods or on fine-tuning of generative LLMs to achieve state-of-the-art performance, speed and resource limits are concerns when considering to deploy such pipelines (Parthasarathy et al., 2024). For enterprises bound by strict data security rules and resource constraints, our work presents a favourable alternative for the training and serving of on-premise OOC detection.

Limitations

We frame OOC detection as a binary classification problem, without distinguishing between common fine-grained subtypes such as entity, relation, invented, or subjective OOC occurrences (Mishra et al., 2024). A popular direction in recent work is to use strong LLMs (e.g. GPT-4) to generate high-quality candidates semantically similar to the retrieved context that contain OOC subtypes. This includes error insertion, answer perturbation, and new claim generation, reducing the need for human annotation (Ravi et al., 2024; Tang et al., 2024; Mishra et al., 2024; Li et al., 2023; Tang et al., 2025). While effective, these pipelines and subsequent fine-tuning are often infeasible in closed or resource-constrained environments. Our method instead applies random substitutions and insertions to construct sentence-context pairs with binary labels, offering scalability and adaptability across domains, though potentially introducing a larger semantic gap than LLM-curated datasets. Future work may quantify this "simulation gap" and assess subtype-specific performance to better understand the trade-off between realism and scalability.

Our approach assumes sentence-level independence, limiting the ability to model inter-sentence dependencies. However, this design reflects enterprise requirements, where a single OOC sentence can invalidate an entire response. We further apply context splitting for BERT-based models, which may cause losses in semantic information if multiple chunks are needed to fully support a single sentence. Despite this limitation, RAGulator models perform comparably or better than black-box methods with much longer context windows, including Bespoke-MiniCheck-7B (32,768 tokens) and zero-shot prompting of Llama-3.1 (128,000 tokens). Given that contexts in our dataset do not exceed 5,000 tokens, these methods should avoid semantic losses, suggesting that sentence-level detection and context splitting remain appropriate for enterprise OOC detection. Future work may develop resource-efficient models that capture inter-sentence and inter-chunk relationships for real-time deployment.

Ethics Statement

To maintain the privacy of related organisations, we anonymised all entity names and substituted all numerical values that were mentioned in industry examples highlighted in this work.

Acknowledgements

We express our gratitude to Donald MacDonald (Head, OCBC Group Data Office) and Kenneth Zhu (Head, OCBC AI Lab) for their invaluable support and access to on-premise computing resources. We also thank Adrien Chenailler and Chengquan Ju for reviewing an earlier version of this work and providing helpful suggestions. We also express our appreciation to all reviewers for their insightful feedback.

References

Masha Belyi, Robert Friel, Shuai Shao, and Atindriyo Sanyal. 2025. Luna: A lightweight evaluation model to catch language model hallucinations with high accuracy and low cost. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 398–409, Abu Dhabi, UAE. Association for Computational Linguistics.

Bespoke Labs. A leaderboard for grounded factuality. Accessed on Sep 27, 2024.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Michelle Brachman, Amina El-Ashry, Casey Dugan, and Werner Geyer. 2024. How knowledge workers use and want to use llms in an enterprise context. In *CHI'24: CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery.

Yung-Sung Chuang, Linlu Qiu, Cheng-Yu Hsieh, Ranjay Krishna, Yoon Kim, and James R. Glass. 2024. Lookback lens: Detecting and mitigating contextual hallucinations in large language models using only attention maps. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1419–1436, Miami, Florida, USA. Association for Computational Linguistics.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Robert Friel and Atindriyo Sanyal. 2023. Chainpoll: A high efficacy method for llm hallucination detection. *arXiv preprint arXiv:2310.18344*.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. 2023. RARR: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16477–16508, Toronto, Canada. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Philippe Laban, Tobias Schnabel, Paul N. Bennett, and Marti A. Hearst. 2022. SummaC: Re-visiting NLI-based models for inconsistency detection in summarization. *Transactions of the Association for Computational Linguistics*, 10:163–177.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020.

Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. HaluEval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464, Singapore. Association for Computational Linguistics.

Ningke Li, Yuekang Li, Yi Liu, Ling Shi, Kailong Wang, and Haoyu Wang. 2024. Drowzee: Metamorphic testing for fact-conflicting hallucination detection in large language models. *Proceedings of the ACM on Programming Languages*, 8(OOPSLA2):1843–1872.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.

Carsten Maple, Alpay Sabuncuoglu, Lukasz Szpruch, Andrew Elliot, Gesine Reinert, and Tony Zemaitis. 2024. The impact of large language models in finance: Towards trustworthy adoption. Technical report, The Alan Turing Institute.

I. Dan Melamed, Ryan Green, and Joseph P. Turian. 2003. Precision and recall of machine translation. In *Companion Volume of the Proceedings of HLT-NAACL 2003 - Short Papers*, pages 61–63.

Ofer Mendelevitch, Forrest Bao, Miaoran Li, and Rogger Luo. 2024. Hhem 2.1: A better hallucination detection model and a new leaderboard. Accessed on Sep 23, 2024.

Abhika Mishra, Akari Asai, Vidhisha Balachandran, Yizhong Wang, Graham Neubig, Yulia Tsvetkov, and Hannaneh Hajishirzi. 2024. Fine-grained hallucination detection and editing for language models.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. 2024. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*.

Selvan Sunitha Ravi, Bartosz Mielczarek, Anand Kannappan, Douwe Kiela, and Rebecca Qian. 2024. Lynx: An open source hallucination evaluation model. *arXiv preprint arXiv:2407.08488*.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. ARES: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354, Mexico City, Mexico. Association for Computational Linguistics.

Sina Semnani, Violet Yao, Heidi Zhang, and Monica Lam. 2023. Wikichat: Stopping the hallucination of large language model chatbots by few-shot grounding on wikipedia. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics.

Liyan Tang, Philippe Laban, and Greg Durrett. 2024. MiniCheck: Efficient fact-checking of LLMs on grounding documents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8818–8847, Miami, Florida, USA. Association for Computational Linguistics.

Xiaqiang Tang, Jian Li, Keyu Hu, Nan Du, Xiaolong Li, Xi Zhang, Weigao Sun, and Sihong Xie. 2025. CogniBench: A legal-inspired framework and dataset for assessing cognitive faithfulness of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21567–21585, Vienna, Austria. Association for Computational Linguistics.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, and Thang Luong. 2024. FreshLLMs: Refreshing large language models with search engine augmentation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13697–13720, Bangkok, Thailand. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packaged resources to advance general chinese embedding. *arXiv preprint arXiv:2309.07597*.

Yuheng Zha, Yichi Yang, Ruichen Li, and Zhiting Hu. 2023. AlignScore: Evaluating factual consistency with a unified alignment function. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11328–11348, Toronto, Canada. Association for Computational Linguistics.

A Notes on feature engineering

Retrieved context can contain noise due to reasons such as suboptimal parsing of reference documents, presence of formatting characters, and inconsistent grammatical or lexical structure. Thus, we apply text preprocessing to both response and context before generating classical features. Preprocessing steps include lowercase conversion, removal of most punctuation symbols, stemming, and stopword removal. For semantic features, we use the raw response-context pair that BERT-based models expect as input.

Similarity and relevance scores are aggregated by maximum. As long as the candidate is relevant to one context sentence, the candidate should be considered grounded in context, therefore capturing the extent to which the candidate is supported.

B Model training hyperparameters

Table 5 shows the hyperparameter search space used to tune the LightGBM and Random Forest meta-classifiers, while Table 6 shows the relevant hyperparameters for the training of BERT classifiers. During BERT fine-tuning, we observed that the validation performance no longer increases after 3 epochs.

C Llama-3.1 for generative labelling

Figure 5 shows the prompt template for zero-shot generative labelling of training examples. To elicit zero-shot chain-of-thought, we use the technique described by Kojima et al. (2022), adding the following line to the instruction while leaving all else unchanged:

```
Let's think step-by-step. After
presenting your thought process,
list down the supporting context
sentences by writing "The answer
is:" followed by the list.
```

Figure 6 shows the prompt template for five-shot generative labelling, and Figure 7 shows the prompt template that elicits chain-of-thought for five-shot prompting, following the technique described by Wei et al. (2022).

D Direct prompting of Llama-3.1 for OOC detection

To perform OOC detection with Llama-3.1, we directly prompt it to return a binary prediction – 1 for OOC and 0 otherwise. Figure 8 shows the template used.

To obtain an OOC prediction probability $P(OOC)$, we extract the log-probabilities (logprobs) of the "0" or "1" token choices in the first output token and calculate softmax of the "1" token based on these 2 choices only:

$$P(OOC) = \frac{\exp(log(P(y=\text{"1"})))}{\exp(log(P(y=\text{"0"})))+\exp(log(P(y=\text{"1"})))}$$

where $log(P(y = \text{"1"}))$ is the logprob of predicting the "1" token and $log(P(y = \text{"0"}))$ is the logprob of predicting the "0" token.

E Comparison to other black-box frameworks and grey-box models

E.1 Black-box frameworks

We evaluate the following black-box frameworks:

RAGAS Faithfulness[5]**:** we consider only the Faithfulness metric as its definition is similar to the OOC definition, except for the generalisation to include sentences containing multiple claims. We use release version 0.2.15.

SelfCheckGPT-Prompt[6]**:** this SelfCheckGPT variant relies on an LLM judge to assess consistency between stochastically-generated answer samples. For each ESG and CP data point, we use the original question to sample 3 responses from Llama-3.1-70B-Instruct at a temperature of 1.0. The in-distribution holdout split does not apply to SelfCheckGPT, as the concept of a "question" does not exist in the dataset curation process; therefore, results on the overall evaluation dataset are not applicable. We do not include the sampling process in our measurement of inference speed. We use release version 0.1.7.

E.2 Grey-box models

We evaluate the following grey-box models:

AlignScore-large[7]**:** this model is a finetune of deberta-v3-large, and can be directly compared to RAGulator. We use the authors' original implementation of AlignScore as retrieved from GitHub (commit hash beginning with a0936d5).

MiniCheck-deberta-v3-large[8]**:** this model is a finetune of deberta-v3-large, and can be directly compared to RAGulator. We use the authors' original implementation of MiniCheck as retrieved from GitHub (commit hash beginning with c655973).

[5] github.com/explodinggradients/ragas
[6] github.com/potsawee/selfcheckgpt
[7] github.com/yuh-zha/AlignScore
[8] github.com/Liyan06/MiniCheck

Bespoke-MiniCheck-7B[8]**:** this model is a fine-tune of the InternLM2 chat model (internlm2_5-7b-chat) consisting of 7.74 billion parameters. We use the authors' original implementation of MiniCheck, including the vLLM framework dependency, as retrieved from GitHub (commit hash beginning with c655973).

Hyperparameter	LightGBM	Random Forest
max_depth	[2,4,-1]	[1,2,3,4,5]
n_estimators	[60,100,200]	[100,325,550,775,1000]
num_leaves	[4,10,31]	-
subsample	[0.8,1.0]	-

Table 5: Grid search values for hyperparameter tuning of meta-classifiers.

Hyperparameter	Value
Batch size	16
Epochs	3
Learning rate (base model)	5×10^{-6}
Learning rate (classification head)	2×10^{-5}
Warmup steps	10% of total training steps

Table 6: Hyperparameter values used for training of BERT classifiers.

```
 ### Instruction ###
You are a professional proofreader with an eye for detail. Given a candidate sentence
and a numbered list of context sentences, your job is to identify which context
sentences support the claim in the candidate sentence. A context sentence supports
the claim in the candidate sentence if it is an exact match, or close to an exact
match. You also know that there must be at least one supporting context sentence.
Return the supporting context sentence identified by its index only. If there are
multiple supporting context sentences, delimit the indexes by comma.

### Candidate sentence ###
<This is the candidate sentence>

### Context sentences ###
0. """<This is the first context sentence>"""
1. """<This is the second context sentence>"""
...

### Response ###
```

Figure 5: Zero-shot prompt template for generative labelling.

```
 ### Instruction ###
You are a professional proofreader with an eye for detail. Given a candidate sentence
and a numbered list of context sentences, your job is to identify which context
sentences support the claim in the candidate sentence. A context sentence supports
the claim in the candidate sentence if it is an exact match, or close to an exact
match. You also know that there must be at least one supporting context sentence.
Return the supporting context sentence identified by its index only. If there are
multiple supporting context sentences, delimit the indexes by comma. A few examples
of the task will be given, followed by the actual task.

# Example 1 #
### Candidate sentence ###
Washington, D.C. is the capital of the US.

### Context sentences ###
0. """Washington, D.C., formally the District of Columbia, is the capital city and
federal district of the United States."""
1. """The city is on the Potomac River, across from Virginia."""
2.  """The city was founded in 1791, and the 6th Congress held the first session
in the unfinished Capitol Building in 1800 after the capital moved from Philadephia."""

### Response ###
[0]

<Examples 2-5>

# Actual task #
### Candidate sentence ###
<This is the candidate sentence>

### Context sentences ###
0. """<This is the first context sentence>"""
1. """<This is the second context sentence>"""
...

### Response ###
```

Figure 6: Five-shot prompt template for generative labelling.

```
 ### Instruction ###
You are a professional proofreader with an eye for detail.  Given a candidate
sentence and a numbered list of context sentences, your job is to identify which
context sentences support the claim in the candidate sentence. A context sentence
supports the claim in the candidate sentence if it is an exact match, or close to
an exact match.  You also know that there must be at least one supporting context
sentence. Let's think step by step. After presenting your thought process, list down
the supporting context sentences by writing "The answer is:" followed by the list.
Return the supporting context sentence identified by its index only.  If there are
multiple supporting context sentences, delimit the indexes by comma. A few examples
of the task will be given, followed by the actual task.

# Example 1 #
### Candidate sentence ###
Washington, D.C. is the capital of the US.

### Context sentences ###
0. """Washington, D.C., formally the District of Columbia, is the capital city and
federal district of the United States."""
1. """The city is on the Potomac River, across from Virginia."""
2.  """The city was founded in 1791, and the 6th Congress held the first session
in the unfinished Capitol Building in 1800 after the capital moved from Philadephia."""

### Response ###
To identify the supporting context sentences, I will analyse each context sentence
to see if it matches or closely matches the claim in the candidate sentence.  The
candidate sentence claims that "Washington, D.C. is the capital of the US." Context
sentence 0 states that "Washington, D.C., formally the District of Columbia, is
the capital city and federal district of the United States." This sentence is an
exact match, as it also states that Washington, D.C. is the capital of the United
States. Context sentence 1 mentions the location of the city, but does not mention
its status as the capital. Context sentence 2 mentions the history of the city and
the capital moving from Philadelphia, but does not explicitly state that Washington,
D.C. is the current capital. The answer is: [0]

<Examples 2-5>

# Actual task #
### Candidate sentence ###
<This is the candidate sentence>

### Context sentences ###
0. """<This is the first context sentence>"""
1. """<This is the second context sentence>"""
...

### Response ###
```

Figure 7: Five-shot chain-of-thought prompt template for generative labelling.

```
 ### Instruction ###
Given a candidate sentence and a reference text, your job is to identify if the
reference text supports the claim in the candidate sentence.  The reference text
supports the claim in the candidate sentence if it is an exact match. The reference
text may also support the claim if the claim is grounded in, or entails, the
reference text. Respond with the final prediction only: 0 if the claim is supported,
1 if the claim is not supported.

### Candidate sentence ###
<This is the candidate sentence>

### Reference text ###
<This is the reference text>

### Response ###
```

Figure 8: Zero-shot prompt template for OOC detection using Llama-3.1-70B-Instruct.