Fine-Tuning Cross-Encoders for Detecting Hallucinations in Financial Customer Chatbots

Tahlee Stone
University of California, Berkeley, School of Information
tahlee.stone@berkeley.edu

Abstract

Mitigating hallucination risk in large language models (LLMs) is essential for customer-facing production chatbots, particularly in sensitive sectors such as finance and healthcare. This paper presents a groundedness classification approach that fine-tunes a cross-encoder model to detect hallucinations in LLM-generated responses. The model is trained on two datasets: the benchmark HaluEval corpus and a synthetically constructed challenger set derived from U.S. retail bank customer interactions. Performance is evaluated against three baselines: a bi-encoder classifier, a prompted GPT-4o groundedness binary classifier, and a commercial guardrail (AWS Bedrock). The cross-encoder jointly encodes the query, context, and response, enabling fine-grained detection of factual inconsistencies. Results show that fine-tuned cross-encoders outperform all baselines, offering a scalable and effective solution for post-generation hallucination detection in financial dialogue systems.

1 Introduction

Large Language Models (LLMs) have rapidly advanced the field of natural language generation (NLG), producing text that is fluent, coherent, and contextually appropriate. However, a critical and persistent challenge remains: hallucination. Hallucination refers to the generation of content that is not grounded in the source material—facts or claims that are either unsupported, inconsistent with the given context, or entirely fabricated. In high-stakes domains such as finance, law, and healthcare, these inaccuracies can lead to harmful misinformation, erode user trust, and impair real-world decision-making (**??**).

To address this issue, groundedness guardrails—systems that verify whether generated responses adhere to a trusted knowledge base—have become essential for LLM deployment. These systems frame hallucination detection as a post-generation classification task over (query, context, response) triplets. A response is deemed *grounded* if all its factual assertions are supported by the provided context; otherwise, it is classified as *ungrounded* or hallucinated. Recent research has demonstrated that classifiers using **cross-encoder architectures** are particularly effective at this task, as they can jointly encode and attend to the full interaction between the query, context, and model response (**??**).

This paper presents a groundedness detection framework in which we fine-tune a **cross-encoder classifier** on a domain-specific dataset drawn from a financial knowledge source—the Reserve Bank of Australia (RBA) website. We generate and label a corpus of synthetic examples using large language models, creating a dataset of (query, context, response) instances annotated as grounded or ungrounded. We then train a cross-encoder model and evaluate its ability to detect hallucinations.

We compare the performance of this classifier against three baselines:

1. A **bi-encoder** model that encodes the query, context, and response independently;
2. A **prompted LLM-based verifier** using self-check prompts (e.g., SelfCheckGPT);
3. An **off-the-shelf groundedness guardrail**, such as AWS Bedrock's hallucination filter.

While cross-encoders incur a higher computational cost than bi-encoders, they offer a practical and cost-effective alternative to full LLM-based verification. Their ability to model token-level interactions allows for fine-grained assessment of semantic consistency, outperforming both bi-encoders and zero-shot LLMs in prior work on entailment and factuality detection (**??**).

Our contributions are threefold:

- We construct a new, domain-grounded dataset for hallucination detection in financial lan-

guage, derived from a trustworthy public source;

- We implement and benchmark a cross-encoder classifier against bi-encoder and LLM-based baselines;
- We demonstrate that cross-encoders strike an effective balance between accuracy and efficiency, making them a viable component in real-world LLM guardrails.

2 Related Work

2.1 Hallucination Detection

[Prior work: FactCC, SelfCheckGPT, GPT-4 verifiers.]

2.2 Cross-Encoders vs Bi-Encoders

[Comparison of architectures and use cases.]

3 Data

3.1 Benchmark Dataset: HaluEval

4 Data

We utilize the HaluEval dataset (**?**), a large-scale benchmark for evaluating hallucinations in large language models (LLMs). HaluEval consists of 35,000 examples covering three tasks—question answering, knowledge-grounded dialogue, and summarization—and includes both human-annotated and automatically generated (hallucinated) responses.

Each instance in HaluEval includes a user query, a supporting context passage, and a response generated by an LLM (e.g., ChatGPT). The dataset provides binary hallucination labels (grounded vs. ungrounded), annotated by expert raters using a high-agreement labeling protocol. This rich alignment between query, context, and response enables robust training of supervised models for groundedness detection.

For our experiments, we sample 500 QA-style examples and expand each into two labeled datapoints: one grounded response and one hallucinated response. This results in a dataset of 1,000 (query, context, response) triplets, each labeled as either grounded (1) or hallucinated (0). The HaluEval benchmark was chosen primarily because it contains high-quality, aligned triplets required for training cross-encoder models. Furthermore, its diversity of hallucination types (e.g., factual contradiction, unverifiability, inference errors) aligns well with our objective of learning nuanced hallucination detection.

This curated subset allows for efficient training while preserving task diversity, making it suitable for evaluating both fine-tuned cross-encoders and zero-shot guardrail baselines.

Attribute	Value
Source Dataset	HaluEval (QA subset)
Task Type	Question Answering
Initial Samples	500
Total Instances After Expansion	1,000
Grounded Responses	500
Hallucinated Responses	500
Input Format	(Query, Context, Response) tri
Label Type	Binary (0 = Hallucinated, 1 =
Annotation Method	Human + Synthetic Generatio

Table 1: Summary of the dataset used for fine-tuning and evaluation.

4.1 Synthetic Challenger Dataset

[Describe generation method, domain specificity, labeling process.]

5 Methodology

5.1 Model Architecture

[Cross-encoder structure. Input formatting. Binary classification.]

5.2 Baselines

- Bi-encoder classifier
- Prompted GPT-4o binary groundedness classifier
- AWS Bedrock hallucination guardrail

5.3 Footnotes

Footnotes are inserted with the `\footnote` command.[1]

5.4 Tables and figures

See Table **??** for an example of a table and its caption. **Do not override the default caption sizes.** See Figure 1 for an example of a figure and its caption.

[1]This is a footnote.

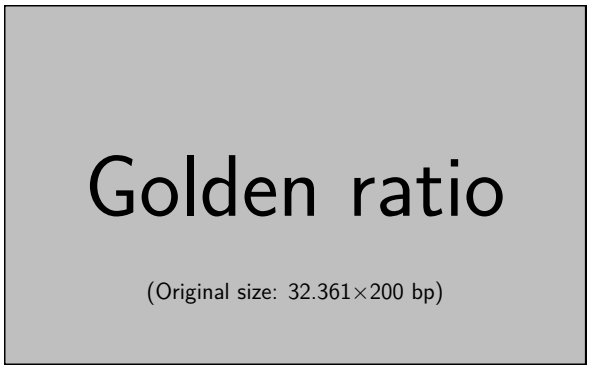

Figure 1: A figure with a caption that runs for more than one line. Example image is usually available through the `mwe` package without even mentioning it in the preamble.

5.5 Hyperlinks

Users of older versions of LaTeX may encounter the following error during compilation:

```
\pdfendlink ended up in different
nesting level than \pdfstartlink.
```

This happens when pdfLaTeX is used and a citation splits across a page boundary. The best way to fix this is to upgrade LaTeX to 2018-12-01 or later.

5.6 Citations

`\usepackage{graphicx}`.

Table **??** shows the syntax supported by the style files. We encourage you to use the natbib styles. You can use the command `\citet` (cite in text) to get "author (year)" citations, like this citation to a paper by Gusfield (1997). You can use the command `\citep` (cite in parentheses) to get "(author, year)" citations (Gusfield, 1997). You can use the command `\citealp` (alternative cite without parentheses) to get "author, year" citations, which is useful for using citations within parentheses (e.g. Gusfield, 1997).

5.7 Training Setup

[Loss function, hyperparameters, evaluation metrics.]

6 Results

6.1 Main Evaluation

6.2 Error Analysis

[Discuss types of hallucinations missed, false positives, patterns across datasets.]

7 Discussion

[Tradeoffs of using cross-encoders. Latency vs performance. Domain transferability. Real-world deployment constraints.]

8 Limitations

[Model size, cost, domain specificity, labeling assumptions.]

9 Conclusion

[Summary of findings. Implications for scalable, reliable financial dialogue systems. Future work.]

9.1 References

The LaTeX and BibTeX style files provided roughly follow the American Psychological Association format. If your own bib file is named `custom.bib`, then placing the following before any appendices in your LaTeX file will generate the references section for you:

```
\bibliographystyle{acl_natbib}
\bibliography{custom}
```

You can obtain the complete ACL Anthology as a BibTeX file from https://aclweb.org/anthology/anthology.bib.gz. To include both the Anthology and your own .bib file, use the following instead of the above.

```
\bibliographystyle{acl_natbib}
\bibliography{anthology,custom}
```

Please see Section 9 for information on preparing BibTeX files.

9.2 Appendices

Use `\appendix` before any appendix section to switch the section numbering over to letters. See Appendix A for an example.

10 BibTeX Files

Unicode cannot be used in BibTeX entries, and some ways of typing special characters can disrupt BibTeX's alphabetization. The recommended way of typing special characters is shown in Table **??**.

Please ensure that BibTeX records contain DOIs or URLs when possible, and for all the ACL materials that you reference. Use the `doi` field for DOIs and the `url` field for URLs. If a BibTeX entry has a URL or DOI field, the paper title in the references section will appear as a hyperlink to the paper, using the hyperref LaTeX package.

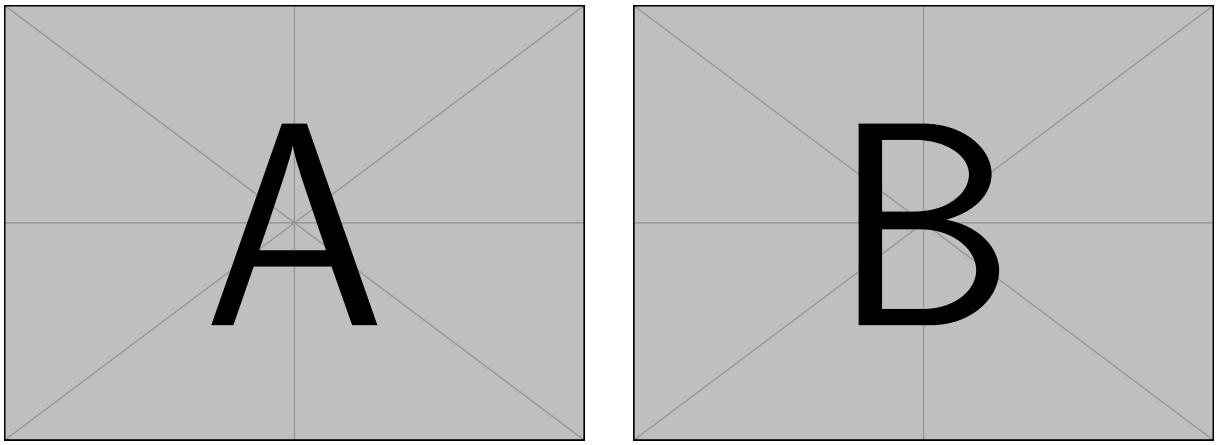

Figure 2: A minimal working example to demonstrate how to place two images side-by-side.

Limitations

ACL 2023 requires all submissions to have a section titled "Limitations", for discussing the limitations of the paper as a complement to the discussion of strengths in the main text. This section should occur after the conclusion, but before the references. It will not count towards the page limit. The discussion of limitations is mandatory. Papers without a limitation section will be desk-rejected without review.

While we are open to different types of limitations, just mentioning that a set of results have been shown for English only probably does not reflect what we expect. Mentioning that the method works mostly for languages with limited morphology, like English, is a much better alternative. In addition, limitations such as low scalability to long text, the requirement of large GPU resources, or other things that inspire crucial further investigation are welcome.

Ethics Statement

Scientific work published at ACL 2023 must comply with the ACL Ethics Policy.[2] We encourage all authors to include an explicit ethics statement on the broader impact of the work, or other ethical considerations after the conclusion but before the references. The ethics statement will not count toward the page limit (8 pages for long, 4 pages for short papers).

[2] https://www.aclweb.org/portal/content/acl-code-ethics

Acknowledgements

This document has been adapted by Jordan Boyd-Graber, Naoaki Okazaki, Anna Rogers from the style files used for earlier ACL, EMNLP and NAACL proceedings, including those for EACL 2023 by Isabelle Augenstein and Andreas Vlachos, EMNLP 2022 by Yue Zhang, Ryan Cotterell and Lea Frermann, ACL 2020 by Steven Bethard, Ryan Cotterell and Rui Yan, ACL 2019 by Douwe Kiela and Ivan Vulić, NAACL 2019 by Stephanie Lukin and Alla Roskovskaya, ACL 2018 by Shay Cohen, Kevin Gimpel, and Wei Lu, NAACL 2018 by Margaret Mitchell and Stephanie Lukin, BibTEX suggestions for (NA)ACL 2017/2018 from Jason Eisner, ACL 2017 by Dan Gildea and Min-Yen Kan, NAACL 2017 by Margaret Mitchell, ACL 2012 by Maggie Li and Michael White, ACL 2010 by Jing-Shin Chang and Philipp Koehn, ACL 2008 by Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, ACL 2005 by Hwee Tou Ng and Kemal Oflazer, ACL 2002 by Eugene Charniak and Dekang Lin, and earlier ACL and EACL formats written by several people, including John Chen, Henry S. Thompson and Donald Walker. Additional elements were taken from the formatting instructions of the *International Joint Conference on Artificial Intelligence* and the *Conference on Computer Vision and Pattern Recognition*.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Galen Andrew and Jianfeng Gao. 2007. Scalable train-

ing of L_1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885, Austin, Texas. Association for Computational Linguistics.

James W. Cooley and John W. Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90):297–301.

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Noise reduction and targeted exploration in imitation learning for Abstract Meaning Representation parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–11, Berlin, Germany. Association for Computational Linguistics.

Dan Gusfield. 1997. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, Cambridge, UK.

Mary Harper. 2014. Learning from 26 languages: Program management and science in the babel program. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, page 1, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

A Example Appendix

This is a section in the appendix.