M.Sc. Thesis
Master of Science in Engineering

DTU Compute
Department of Applied Mathematics and Computer Science

Logical Fallacy Detection using LLMs

Master Thesis

Eleftherios Katiforis (s222725)

Kongens Lyngby 2025

Department of Applied Mathematics and Computer Science
Technical University of Denmark
Richard Petersens Plads
Building 324, Room 160
2800 Kongens Lyngby, Denmark
www.compute.dtu.dk

Supervisors:

Abstract

Preface

The present MSc thesis is a partial fulfillment of the requirements for the Master of Science degree in Human-Centered Artificial Intelligence at Technical University of Denmark.

The project accounts for 35 ECTS and is conducted between 4.05.2024 and 04.03.2025. The exact topic along with the objectives of the project are defined in collaboration between the author and the supervisors.

Acknowledgements

Contents

CHAPTER 1

Introduction

1.1 Thesis structure

CHAPTER 2

Background & Related Work

2.1 Fallacy Detection

Da San Martino et al. (2019) [3] were the first to aim to detect propaganda in news articles at a granular level by identifying specific techniques used within the text, rather than labeling entire articles as propagandistic. The authors introduced a new dataset of news articles, annotated with 18 distinct propaganda techniques, such as loaded language, name calling, appeal to fear, and whataboutism. This approach enables a more detailed and explainable analysis of how propaganda operates within specific text fragments.

Jin et al. (2022) [5] focused on detecting logical fallacies in text, a task essential for identifying flawed reasoning that can propagate misinformation The authors introduced the LOGIC dataset, which contains 2,449 samples annotated with 13 types of logical fallacies, along with a specialized climate change dataset (LOGICCLIMATE) for fallacies found in climate discussions. Furthermore, they presented a structure-aware model that outperformed standard language models like BERT and RoBERTa in detecting logical fallacies, significantly improving the F1 score. The model focuses on the logical structure of arguments rather than just the content. This work focuses more on the classification of annotated samples that consist of 2-3 sentences rather than the detection of logical fallacies in large corpora of text.

Alhindi et al. (2022) [1] explored the detection of fallacies across different types of datasets using a unified multitask framework. The main innovation of the paper was the use of instruction-based prompting with a T5 model, which enabled the model to handle 28 unique fallacies across various genres and domains by transforming them into natural language instructions. This approach significantly outperformed traditional methods trained on specific datasets or tasks, offering a broad, adaptable, and effective system for fallacy recognition. The authors address several challenges in fallacy detection, such as dataset heterogeneity and the high number of classification labels, by unifying multiple datasets under a single framework. They demonstrate that their method not only improves the macro F1 scores by considerable margins but also provides insights into the effects of model size and prompt choices on classification accuracy.

Vorakitphan et al. (2023) [7] introduced PROTECT (PROpaganda Text dEteCTion), a system designed to automatically detect and classify propaganda in texts. PROTECT uses semantic and argumentation features to analyze texts for propaganda content, identifying the techniques employed. PROTECT first identifies propaganda snippets within a given text and then classifies them based on the type of propaganda technique used. The system utilizes a transformer architecture, likely leveraging models like BERT, to perform its classification tasks. It is evaluated using standard datasets known for propaganda analysis introduced by Da San Martino et al. (2019)[3], showing promising results. PROTECT is equipped with a user-friendly web interface and API, making it accessible for public use and allowing for the analysis of user-submitted texts. This work contributed to the field by providing a tool that not only automates the detection of propaganda but also helps educate the public about the nuances of manipulative content in media.

Goffredo et al. (2023) [4] delved into the detection and categorization of logical fallacies in political debates. The authors extended a pre-existing dataset of U.S. presidential debates to include the 2020 Trump-Biden debates, enriching it with detailed annotations of argumentative structures and fallacies. The methodology focused on a transformer-based neural network model integrating text analysis with argumentative and engineered features, significantly improving the detection and classification of fallacies. The study presents a nuanced approach to handling fallacious arguments by incorporating the relations and components of the arguments, which is a novel contribution to the field of computational linguistics and argument mining.

2.2 Large Language Models

1. Write about LLMs, attention?
2. Reasoning in LLMs
3. LLMs and fallacy classification

2.3 Chain of Thought

Wei et al. (2022) [2] introduced the concept of "Chain-of-Thought Prompting" in their paper Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. This technique involves providing models with a series of intermediate reasoning steps as exemplars during In-Context Learning. Their findings showed that models with over 100 billion parameters experienced significant improvements in solving complex multi-step problems, such as arithmetic and commonsense reasoning. However, smaller models (under 100 billion parameters) not only failed to benefit from this approach but sometimes performed worse, generating fluent but illogical reasoning chains. Additionally, they observed variability in performance—sometimes as much as 20%—due to differences in hand-crafted exemplars created by various annotators.

Building on Wei et al.'s work, Kojima et al. (2022) [6] addressed the primary limitation of Chain-of-Thought Prompting: the dependence on hand-crafted exemplars and their associated performance variability. Their approach introduced a simple phrase, "Let's think step-by-step," to the prompt, encouraging the model to generate logical intermediate reasoning steps. By appending these steps to the original prompt, they achieved notable performance improvements across tasks previously evaluated using Chain-of-Thought Prompting, such as arithmetic and commonsense reasoning. However, while their method reduced reliance on hand-crafted examples, it did not yield as significant a performance boost as the original approach and remained ineffective for smaller models with fewer than 100 billion parameters.

CHAPTER 3

Method

CHAPTER 4

Experiments

4.1 Dataset

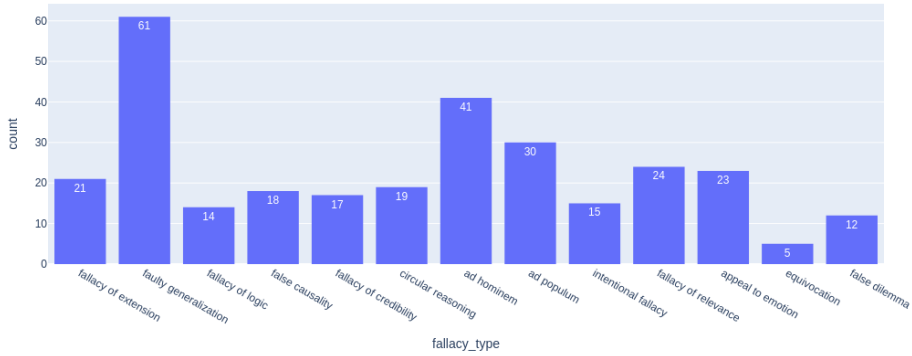

Figure 4.1: LOGIC Dataset

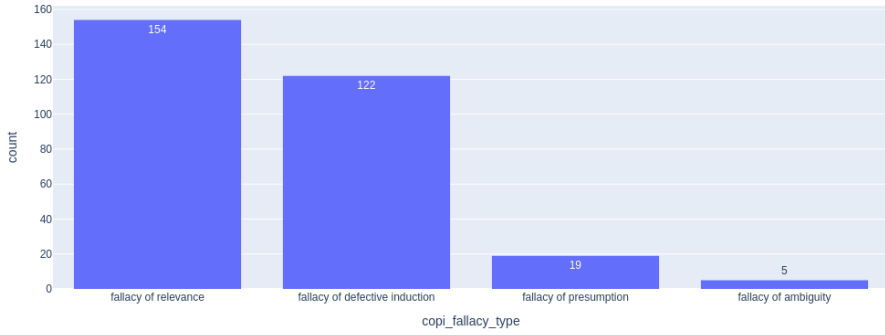

Figure 4.2: LOGIC Dataset to coarse-grained classes by Copi

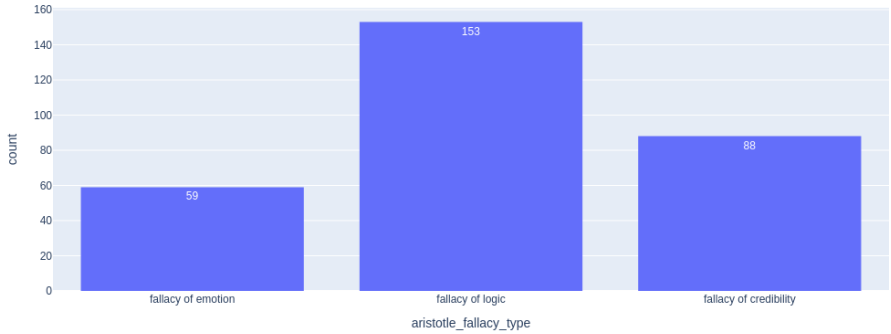

Figure 4.3: LOGIC Dataset to coarse-grained classes by Aristotle

4.2 Zero-shot

4.2.1 Fine Grained Logic Classes

Table 4.1: Fine-grained Classes Results

model	def	prompt	accuracy	f1 score	failed	unknown labels
falcon-mamba-7b-instruct	False	1	19.667	0.062	6.667	9.333
falcon-mamba-7b-instruct	True	1	20.333	0.071	5.667	12.333
Meta-Llama-3.1-8B-Instruct	False	1	41.333	0.219	0.000	1.000
Meta-Llama-3.1-8B-Instruct	True	1	39.000	0.242	0.000	0.667
Mistral-7B-Instruct-v0.3	False	1	39.000	0.133	0.000	4.667
Mistral-7B-Instruct-v0.3	True	1	34.667	0.186	0.000	6.333
falcon-mamba-7b-instruct	False	2	20.000	0.060	5.333	24.000
falcon-mamba-7b-instruct	True	2	19.667	0.063	14.667	21.667
Meta-Llama-3.1-8B-Instruct	False	2	**41.000**	0.191	0.333	1.667
Meta-Llama-3.1-8B-Instruct	True	2	40.667	**0.219**	0.333	0.667
Mistral-7B-Instruct-v0.3	False	2	38.667	0.161	0.000	3.000
Mistral-7B-Instruct-v0.3	True	2	31.333	0.148	0.000	9.000

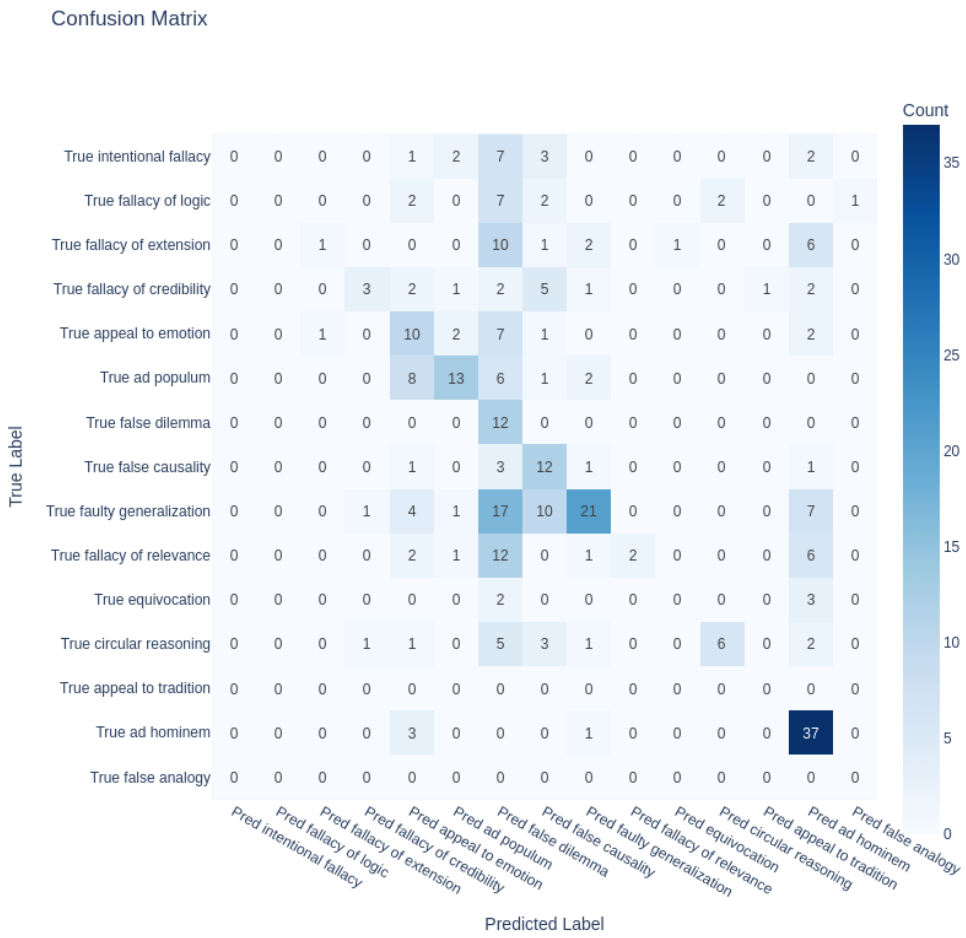

Figure 4.4: Best performing model's results Confusion Matrix

4.2.2 Coarse Grained Copi Classes

Table 4.2: Coarse-grained Copi Classes Results

model	def	prompt	accuracy	f1 score	failed	unknown
falcon-mamba-7b-instruct	False	1	41.333	0.082	8.000	3.000
falcon-mamba-7b-instruct	True	1	45.667	0.102	11.667	2.000
Meta-Llama-3.1-8B-Instruct	False	1	49.333	0.277	0.000	0.333
Meta-Llama-3.1-8B-Instruct	True	1	**57.000**	0.259	0.000	0.667
Mistral-7B-Instruct-v0.3	False	1	43.333	0.086	0.000	5.333
Mistral-7B-Instruct-v0.3	True	1	50.333	0.084	0.000	2.333
falcon-mamba-7b-instruct	False	2	29.667	0.036	0.333	21.333
falcon-mamba-7b-instruct	True	2	14.000	0.027	0.333	64.333
Meta-Llama-3.1-8B-Instruct	False	2	55.000	**0.300**	0.333	0.000
Meta-Llama-3.1-8B-Instruct	True	2	51.667	0.231	0.333	0.333
Mistral-7B-Instruct-v0.3	False	2	32.000	0.056	0.000	12.000
Mistral-7B-Instruct-v0.3	True	2	48.000	0.071	0.000	12.333

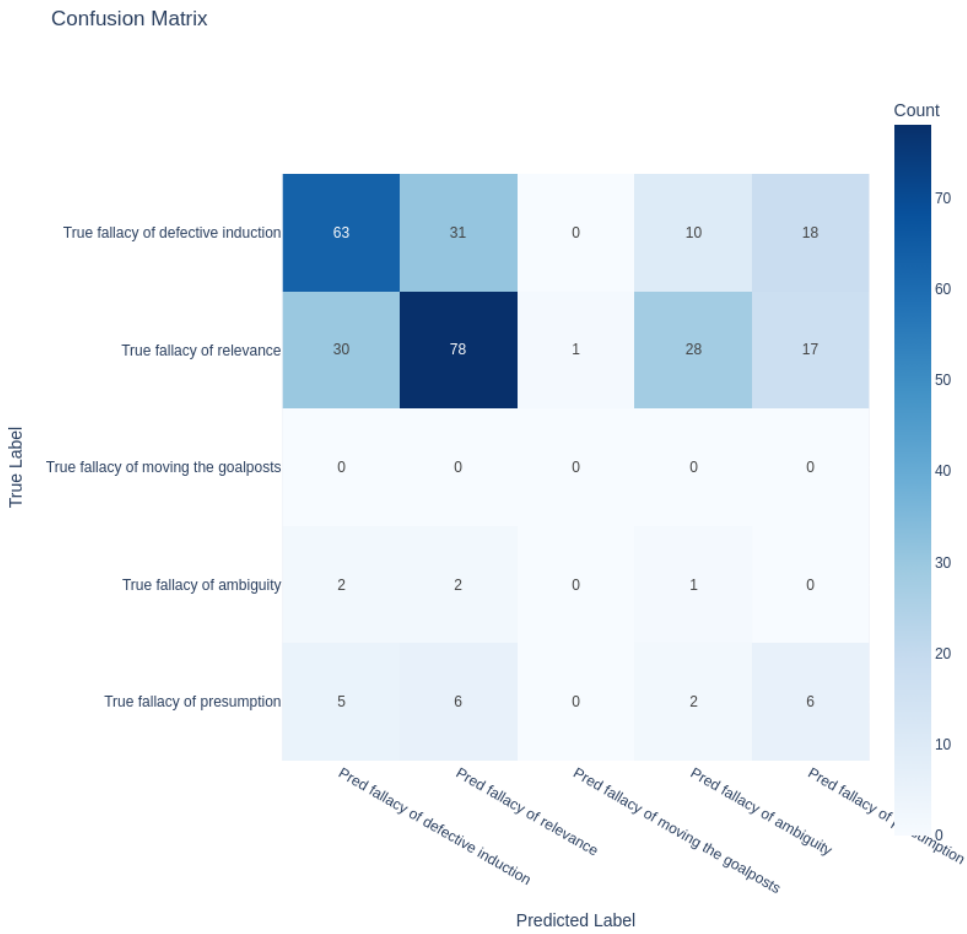

Figure 4.5: Best performing model's results Confusion Matrix

Table 4.3: Results when mapping fine-grained results to Copi's coarse-grained classes

model	def	prompt	accuracy	f1 score	failed	unknown
falcon-mamba-7b-instruct	False	1	50.000	0.263	16.333	0.000
falcon-mamba-7b-instruct	True	1	47.667	0.251	18.000	0.000
Meta-Llama-3.1-8B-Instruct	False	1	70.000	0.377	1.333	0.000
Meta-Llama-3.1-8B-Instruct	True	1	**67.333**	0.366	0.667	0.000
Mistral-7B-Instruct-v0.3	False	1	62.333	0.301	6.000	0.000
Mistral-7B-Instruct-v0.3	True	1	63.000	0.320	7.000	0.000
falcon-mamba-7b-instruct	False	2	47.000	0.277	29.333	0.000
falcon-mamba-7b-instruct	True	2	40.667	0.241	36.667	0.000
Meta-Llama-3.1-8B-Instruct	False	2	67.000	**0.385**	3.333	0.000
Meta-Llama-3.1-8B-Instruct	True	2	66.667	0.377	1.333	0.000
Mistral-7B-Instruct-v0.3	False	2	66.667	0.373	4.667	0.000
Mistral-7B-Instruct-v0.3	True	2	61.000	0.331	10.000	0.000

4.2.3 Coarse Grained Aritotle Classes

Table 4.4: Coarse-grained Aritotle Classes Results

model	def	prompt	accuracy	f1 score	failed	unknown
falcon-mamba-7b-instruct	False	1	35.667	0.115	3.667	2.333
falcon-mamba-7b-instruct	True	1	32.000	0.058	4.667	5.000
Meta-Llama-3.1-8B-Instruct	False	1	50.333	0.457	0.000	0.000
Meta-Llama-3.1-8B-Instruct	True	1	51.333	0.156	0.000	2.333
Mistral-7B-Instruct-v0.3	False	1	52.333	0.108	0.000	7.333
Mistral-7B-Instruct-v0.3	True	1	50.333	0.107	0.000	8.667
falcon-mamba-7b-instruct	False	2	37.333	0.125	0.000	9.333
falcon-mamba-7b-instruct	True	2	28.667	0.065	1.667	30.667
Meta-Llama-3.1-8B-Instruct	False	2	42.667	0.289	0.667	0.000
Meta-Llama-3.1-8B-Instruct	True	2	53.667	**0.292**	0.667	0.333
Mistral-7B-Instruct-v0.3	False	2	56.000	0.088	0.000	9.667
Mistral-7B-Instruct-v0.3	True	2	**59.000**	0.280	0.000	1.333

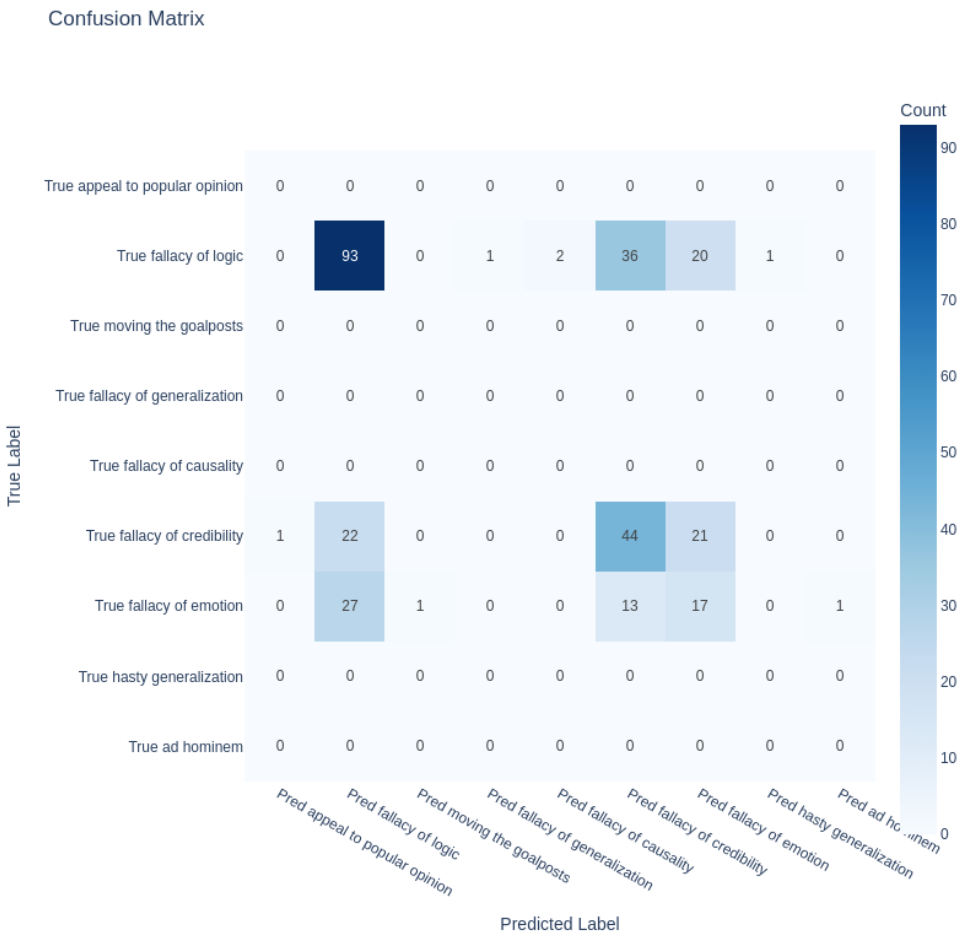

Figure 4.6: Best performing model's results Confusion Matrix

Table 4.5: Results when mapping fine-grained results to Aritotle's coarse-grained classes

model	def	prompt	accuracy	f1 score	failed	unknown
falcon-mamba-7b-instruct	False	1	40.333	0.316	16.333	0.000
falcon-mamba-7b-instruct	True	1	36.667	0.289	18.000	0.000
Meta-Llama-3.1-8B-Instruct	False	1	63.000	0.398	1.333	0.000
Meta-Llama-3.1-8B-Instruct	True	1	62.667	0.408	0.667	0.000
Mistral-7B-Instruct-v0.3	False	1	60.000	0.414	6.000	0.000
Mistral-7B-Instruct-v0.3	True	1	57.667	0.401	7.000	0.000
falcon-mamba-7b-instruct	False	2	38.667	0.329	29.333	0.000
falcon-mamba-7b-instruct	True	2	38.333	0.339	36.667	0.000
Meta-Llama-3.1-8B-Instruct	False	2	**65.333**	**0.436**	3.333	0.000
Meta-Llama-3.1-8B-Instruct	True	2	65.000	0.422	1.333	0.000
Mistral-7B-Instruct-v0.3	False	2	61.000	0.359	4.667	0.000
Mistral-7B-Instruct-v0.3	True	2	54.333	0.327	10.000	0.000

4.3 Basic CoT

4.3.1 Fine Grained Logic Classes

Table 4.6: Fine-grained Classes Results

model	def	prompt	accuracy	f1 score	failed	unknown
falcon-mamba-7b-instruct	False	1	3.333	0.031	66.000	1.667
falcon-mamba-7b-instruct	True	1	11.667	0.085	47.000	2.000
Meta-Llama-3.1-8B-Instruct	False	1	37.000	0.184	0.333	0.667
Meta-Llama-3.1-8B-Instruct	True	1	35.000	0.193	0.667	1.333
Mistral-7B-Instruct-v0.3	False	1	38.333	0.146	0.000	3.333
Mistral-7B-Instruct-v0.3	True	1	33.667	0.156	0.667	7.333
falcon-mamba-7b-instruct	False	2	16.667	0.134	28.333	0.000
falcon-mamba-7b-instruct	True	2	20.667	0.138	10.667	0.000
Meta-Llama-3.1-8B-Instruct	False	2	35.667	0.208	1.000	0.000
Meta-Llama-3.1-8B-Instruct	True	2	**39.333**	**0.246**	1.333	0.000
Mistral-7B-Instruct-v0.3	False	2	37.667	0.206	4.000	0.000
Mistral-7B-Instruct-v0.3	True	2	33.333	0.201	4.667	0.000

4.3.2 Coarse Grained Copi Classes

Table 4.7: Coarse-grained Copi Classes Results

model	def	prompt	accuracy	f1 score	failed	unknown
falcon-mamba-7b-instruct	False	1	19.000	0.089	60.333	1.667
falcon-mamba-7b-instruct	True	1	27.000	0.093	43.333	4.000
Meta-Llama-3.1-8B-Instruct	False	1	50.667	0.223	0.667	0.333
Meta-Llama-3.1-8B-Instruct	True	1	53.000	0.229	0.667	0.333
Mistral-7B-Instruct-v0.3	False	1	34.333	0.052	0.000	8.000
Mistral-7B-Instruct-v0.3	True	1	49.667	0.096	0.000	2.333
falcon-mamba-7b-instruct	False	2	32.333	0.130	14.333	0.000
falcon-mamba-7b-instruct	True	2	41.333	0.203	3.000	0.000
Meta-Llama-3.1-8B-Instruct	False	2	**49.000**	**0.276**	0.333	0.000
Meta-Llama-3.1-8B-Instruct	True	2	46.333	0.269	1.333	0.000
Mistral-7B-Instruct-v0.3	False	2	33.667	0.207	7.667	0.000
Mistral-7B-Instruct-v0.3	True	2	47.333	0.248	6.333	0.000

4.3.3 Coarse Grained Aritotle Classes

Table 4.8: Coarse-grained Aritotle Classes Results

model	def	prompt	accuracy	f1 score	failed	unknown
falcon-mamba-7b-instruct	False	1	11.667	0.101	72.000	0.333
falcon-mamba-7b-instruct	True	1	14.667	0.065	59.333	2.333
Meta-Llama-3.1-8B-Instruct	False	1	52.667	0.324	1.333	0.000
Meta-Llama-3.1-8B-Instruct	True	1	55.333	0.373	1.000	0.000
Mistral-7B-Instruct-v0.3	False	1	54.000	0.121	0.000	6.000
Mistral-7B-Instruct-v0.3	True	1	54.667	0.131	0.000	3.667
falcon-mamba-7b-instruct	False	2	30.000	0.227	17.333	0.000
falcon-mamba-7b-instruct	True	2	35.000	0.251	6.667	0.000
Meta-Llama-3.1-8B-Instruct	False	2	38.667	0.240	0.667	0.000
Meta-Llama-3.1-8B-Instruct	True	2	48.333	0.334	2.667	0.000
Mistral-7B-Instruct-v0.3	False	2	**54.333**	**0.398**	6.333	0.000
Mistral-7B-Instruct-v0.3	True	2	48.667	0.352	6.667	0.000

4.4 From Coarse-Grained to Fine-Grained Multi-Round CoT

Table 4.9: COPI Coarse-Grained to Fine-Grained Multi-round CoT

model	def	prompt	r1_accuracy	r1_f1	acc	f1	failed
falcon-mamba-7b-instruct	False	1	39.000	0.059	4.000	0.010	0.333
falcon-mamba-7b-instruct	True	1	46.667	0.067	5.667	0.022	0.333
Meta-Llama-3.1-8B-Instruct	False	1	53.667	0.256	29.000	0.167	0.000
Meta-Llama-3.1-8B-Instruct	True	1	53.333	0.359	28.667	0.216	0.000
Mistral-7B-Instruct-v0.3	False	1	40.000	0.056	31.333	0.176	0.000
Mistral-7B-Instruct-v0.3	True	1	49.667	0.060	29.000	0.177	0.000
falcon-mamba-7b-instruct	False	2	27.333	0.034	5.333	0.018	0.000
falcon-mamba-7b-instruct	True	2	15.000	0.024	9.667	0.049	0.000
Meta-Llama-3.1-8B-Instruct	False	2	52.000	0.358	29.333	0.186	0.000
Meta-Llama-3.1-8B-Instruct	True	2	44.000	0.262	29.333	**0.236**	0.000
Mistral-7B-Instruct-v0.3	False	2	31.667	0.038	29.667	0.168	0.000
Mistral-7B-Instruct-v0.3	True	2	45.333	0.050	36.000	0.203	0.000

Table 4.10: ARISTOTLE Coarse-Grained to Fine-Grained Multi-round CoT

model	def	prompt	r1_accuracy	r1_f1	acc	f1	failed
falcon-mamba-7b-instruct	False	1	41.000	0.103	10.000	0.040	0.333
falcon-mamba-7b-instruct	True	1	36.667	0.064	13.000	0.062	0.333
Meta-Llama-3.1-8B-Instruct	False	1	56.000	0.465	31.000	0.182	0.000
Meta-Llama-3.1-8B-Instruct	True	1	51.667	0.463	30.000	0.207	0.000
Mistral-7B-Instruct-v0.3	False	1	45.000	0.035	33.667	0.154	0.000
Mistral-7B-Instruct-v0.3	True	1	50.667	0.058	26.667	0.117	0.000
falcon-mamba-7b-instruct	False	2	43.333	0.108	8.667	0.037	0.000
falcon-mamba-7b-instruct	True	2	36.667	0.054	12.000	0.053	0.000
Meta-Llama-3.1-8B-Instruct	False	2	44.667	0.316	28.667	0.172	0.000
Meta-Llama-3.1-8B-Instruct	True	2	52.667	0.479	32.333	**0.238**	0.000
Mistral-7B-Instruct-v0.3	False	2	45.667	0.042	39.000	0.200	0.000
Mistral-7B-Instruct-v0.3	True	2	51.667	0.096	29.667	0.213	0.000

CHAPTER 5

Conclusion

CHAPTER 6

Future Work

Bibliography

[1] Tariq Alhindi et al. "Multitask Instruction-based Prompting for Fallacy Recognition." In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Edited by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022, pages 8172–8187. (Visited on September 24, 2024).

[2] *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. (Visited on September 30, 2024).

[3] Giovanni Da San Martino et al. "Fine-Grained Analysis of Propaganda in News Article." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Edited by Kentaro Inui et al. Hong Kong, China: Association for Computational Linguistics, 2019, pages 5636–5646. (Visited on September 18, 2024).

[4] Pierpaolo Goffredo et al. "Argument-based Detection and Classification of Fallacies in Political Debates." In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Edited by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, 2023, pages 11101–11112. (Visited on September 24, 2024).

[5] Zhijing Jin et al. *Logical Fallacy Detection*. en. arXiv:2202.13758 [cs]. 2022. (Visited on September 18, 2024).

[6] Takeshi Kojima et al. *Large Language Models are Zero-Shot Reasoners*. arXiv:2205.11916. 2023. (Visited on December 9, 2024).

[7] Vorakit Vorakitphan, Elena Cabrio, and Serena Villata. "PROTECT – A Pipeline for Propaganda Detection and Classification." en. In: *Proceedings of the Eighth Italian Conference on Computational Linguistics CliC-it 2021*. Edited by Elisabetta Fersini, Marco Passarotti, and Viviana Patti. Accademia University Press, 2022, pages 352–358. ISBN: 9791280136947. (Visited on September 24, 2024).

{2.7182818284}ϑφερτυθιοπσδφγηξκλ