Argument Mining with Fine-Tuned Large Language Models

Jérémie Cabessa[1,2], Hugo Hernault[3], Umer Mushtaq[4]

[1]DAVID Lab, University of Versailles (UVSQ) – Paris-Saclay, 78035 Versailles, France
[2]Institute of Computer Science of the Czech Academy of Sciences, 18207 Prague 8, Czech Republic
[3]Playtika AI, CH-1003 Lausanne, Switzerland
[4]L3i, University of La Rochelle, 17042 La Rochelle, France
jeremie.cabessa@uvsq.fr hugoh@playtika.com umer.mushtaq@univ-lr.fr

Abstract

An end-to-end argument mining (AM) pipeline takes a text as input and provides its argumentative structure as output by identifying and classifying the argument units and argument relations in the text. In this work, we approach AM using fine-tuned large language models (LLMs). We model the three main sub-tasks of the AM pipeline, as well as their joint formulation, as text generation tasks. We fine-tune eight popular quantized and non-quantized LLMs – LLaMA-3, LLaMA-3.1, Gemma-2, Mistral, Phi-3, Qwen-2 – which are among the most capable open-weight models, on the benchmark PE, AbstRCT, and CDCP datasets that represent diverse data sources. Our approach achieves state-of-the-art results across all AM sub-tasks and datasets, showing significant improvements over previous benchmarks.

1 Introduction

Argument Mining (AM) involves the automatic analysis and parsing of the argumentative structure in natural language texts across diverse domains (Palau and Moens, 2009; Cabrio and Villata, 2018). A complete AM pipeline takes a text as input, identifies and classifies the argument units and argument relations in the text, and outputs the resulting argumentative structure. Sub-tasks of AM include: (1) identifying argument components (ACs) in the text, (2) classifying argument components according to their roles (ACC), (3) identifying argument relations (ARs) between argument components (ARI), and (4) classifying the stance of those relations (ARC) (Stab and Gurevych, 2017).

Initial approaches to AM employ traditional supervised machine learning algorithms, such as Maximum Entropy classifiers (Mochales and Moens, 2011), Logistic Regressions (Levy et al., 2014) and Support Vector Machines (Stab and Gurevych, 2017; Habernal and Gurevych, 2017). Subsequent studies introduce more advanced neural network-based models, including RNNs (Eger et al., 2017; Niculae et al., 2017) and LSTMs/BiLSTMs (Haddadan et al., 2019; Potash et al., 2017; Mayer et al., 2020; Kuribayashi et al., 2019). These investigations convey two core messages: (i) the centrality of incorporating additional task-specific contextual, structural, and syntactic features in the models, and (ii) the importance of capturing the global sequentiality of the argumentative and discursive flow in the text.

Based on these considerations, Bao et al. (2021) propose a joint model for the ACC and ARI tasks using a sequential transition-based BERT–BiLSTM architecture. Additionally, Mushtaq and Cabessa (2022, 2023) introduce customized BERT-based models that integrate contextual, structural, and syntactic features provided in textual form rather than numerically.

Large Language Models (LLMs) have become the dominant paradigm in NLP, demonstrating outstanding performance across a range of tasks and exhibiting notable emergent capabilities (Zhao et al., 2023; Wei et al., 2022). In the realm of LLMs, AM tasks are generally reframed as text generation tasks. Pojoni et al. (2023) use GPT–4 for argument mining in transcribed podcasts. Al Zubaer et al. (2023) approach ACC in the legal domain as text generation using GPT–3.5 and GPT–4. Liu et al. (2023) incorporate COT technique into a customized BART-base text generation model. For every AM sub-task, in addition to the class label, their model also generates a path from the root component to the query component as demonstration of the model's reasoning.

Furthermore, Nori et al. (2023) show that an in-context learning (ICL) approach with GPT (OpenAI, 2023), LLaMA (Touvron et al., 2023), and Qwen (Bai et al., 2023) outperforms fine-tuning for several NLP tasks. However, for Argument Mining, Cabessa et al. (2024) notice that further fine-tuning of LLMs is necessary to optimally capture the ar-

gumentative flow and the sequentiality of argument components and relations.

In this work, we approach AM using fine-tuned LLMs. We model the three main sub-tasks of the AM pipeline (ACC, ARI, ARC) as well as their joint formulations (ARIC and ACC–ARI–ARC), as text generation tasks. We fine-tune eight popular LLMs – LLaMA-3, LLaMA-3.1, Gemma-2, Mistral, Phi-3, Qwen-2 – which are the most capable open-weight models available, on the benchmark PE, AbstRCT, and CDCP datasets that reflect a diversity of data sources. We achieve state-of-the-art results across all AM sub-tasks and datasets, showing significant improvements over previous benchmarks. We attribute these results to the LLMs' remarkable ability to capture the sequentiality of arguments in the text and leverage relevant contextual and structural information from the entire document. Our code is freely available on GitHub.

2 Methodology

2.1 Datasets

We conducted our experiments on three benchmark datasets for Argument Mining: PE (Stab and Gurevych, 2017), AbstRCT (Mayer, 2020) and CDCP (Park and Cardie, 2018). The PE dataset consists of 402 structured Persuasive Essays on various topics. The AbstRCT dataset consists of 650 abstracts of Randomized Controlled Trials selected from PUBMed. The CDCP dataset consists of 731 user comments on Consumer Debt Collection Practices (CDCP). The statistics of these datasets are given in Tables 6, 7 and 8 (Appendix A).

2.2 Tasks

We focus on three main classification tasks that form the core of the AM pipeline: Argument Component Classification (ACC), Argument Relation Identification (ARI) and Argument Relation Classification (ARC). To address these tasks using LLMs, we reformulate them as *text generation tasks*.

ACC: This task involves classifying each argument component (AC) in a text into distinct component types (e.g., 'major claim', 'claim', 'premise'). To accomplish this, we instruct the LLM to generate a list of AC types in the following format:

```
[argument_type (str),..., argument_type (str)]
```

Classification metrics are computed by comparing the lists of generated AC types and ground truth AC types.

ARI: This task involves identifying argument relations (ARs) between pairs of ACs in a text. In practice, this amounts to classifying each pair of distinct ACs as either 'related' or 'non-related'. For this purpose, we instruct the LLM to generate a list of AC pairs in the following format:

```
[[source_AC (int), target_AC (int)],...,
 [source_AC (int), target_AC (int)]]
```

The generated pairs of ACs are then classified as 'related', while non-generated pairs are classified as 'non-related'. Classification metrics are computed by comparing the generated and ground truth pairs.

ARC: This task involves classifying each related argument relation (i, j) into a specific type (e.g., 'support' or 'attack'). To that end, we provide the list of related ACs in the prompt and instruct the LLM to generate a list of corresponding relation types in the following format:

```
[rel_type (str),..., rel_type (str)]
```

ARIC (ARI+ARC): This task involves jointly identifying and classifying argument relations in a text. In this case, we instruct the LLM to generate a list of triplets in the following format:

```
[[source_AC (int), target_AC (int), rel_type (str)]
                        ,...,
 [source_AC (int), target_AC (int), rel_type (str)]]
```

The generated triplets are then classified as 'support' or 'attack' (or 'evidence' or 'reason'), while non-generated triplets are classified as 'none'.

ACC–ARI–ARC: Finally, we also model the three tasks jointly. To achieve this, we instruct the LLM to generate two lists: one for the ACs' types (as in the ACC task), and one for related pairs of ACs along with their AR types (as for the ARIC task). Individual classification metrics for the ACC, ARI, and ARC tasks can then be computed as described above.

2.3 Fine-tuning modalities

Fine-tuning refers to the process of further training a pre-trained LLM on a specific downstream task. LLMs with billions of parameters can be fine-tuned efficiently using the QLoRA strategy, which employs a frozen n-bit quantized version of the pre-trained weights and trains rank decomposition matrices (low rank adapters) of the model's layers (Dettmers et al., 2023).

We fine-tune the following LLMs on PE, AbstRCT, and CDCP datasets for ACC, ARI, ARC

and ARIC tasks: LLaMA-3, LLaMA-3.1, Gemma-2, Mistral, Phi-3 and Qwen-2, with different size and quantization configurations. Following the literature, PE and CDCP datasets are evaluated on the ACC, ARI, and ARC tasks, while AbstRCT dataset is evaluated on the ACC and ARIC tasks(Liu et al., 2023). These four tasks are reformulated as *text generation tasks* as described in Section 2.2.

For all datasets, the text (essay, abstract or comment) is given in the prompt with argument components delimited by tags of the form:

`<AC1>. . .</AC1>, <AC2>. . .</AC2>, . . .`

For PE dataset, two fine-tuning modalities incorporating different contextual and structural information are considered[1]: (i) Paragraph/Essay level: The LLMs are trained and tested on data samples consisting of either individual paragraphs or full essays, respectively; (ii) With/Without structural tags: At the essay level, markup tags of the form:

`<PARAGRAPH_TYPE>. . .</PARAGRAPH_TYPE>`

delimiting the topic, introduction, body paragraphs and conclusion of the essays are inserted in the text. Implementations details are given in Appendix B and examples of training samples are provided in Appendix E.

2.4 Post-processing and guardrails

For each task, the predictions generated by the model are automatically post-processed and the corresponding classification metrics then computed. To this end, we implemented *hallucination guardrails* to filter out predictions with incorrect formats or lengths.

For ACC and ARC, generated lists with incorrect lengths or formats were modified by either removing elements or adding incorrect predictions, which were then counted as errors. For ARI, we evaluated all possible pairs of ACs, classifying them as `REL` or `N-REL` based on whether they were generated by the model or not. For ARIC, the two strategies described above were combined to filter out invalid predictions. Further details are provided in Appendix C.

3 Results

The results for PE, AbstRCT and CDCP datasets are provided in Tables 1, 2 and 3, respectively. As usual, the results are reported using macro F1 scores, and the previous state-of-the-art (SOTA) results are given in the first row of each table.

Overall, our approach surpasses the previous SOTA results across all datasets and tasks. We attribute this improvement to the LLMs' ability to effectively capture the sequentiality of ACs and to leverage contextual and structural information from the entire text. These capabilities are crucial for achieving competitive results across all AM tasks (Stab and Gurevych, 2017; Kuribayashi et al., 2019; Mushtaq and Cabessa, 2022, 2023).

ACC: For the three datasets, our method outperforms the previous SOTA, with significant improvements observed for CDCP dataset. We attribute this performance to the LLMs' ability to capture the sequential pattern of ACs in the argumentative flow.

For PE dataset, an ablation study demonstrating the influence of the paragraph versus essay level approaches, and of excluding versus including structural tags is provided in Appendix D. For ACC, the essay-level approach seems the most optimal, likely because the argumentative flow extends throughout the entire essay.

Model	ACC	ARI	ARC
Liu et al. (2023)	89.2	82.7	81.0
LLaMA-3-8B (4bit)	88.2	**83.5**	86.8
LLaMA-3-8B	**89.5**	83.0	89.6
LLaMA-3-70B (4bit)	87.5	82.0	94.2
LLaMA-3.1-8B (4bit)	88.5	80.7	**95.9**
Gemma-2-9B (4bit)	86.7	79.7	89.4
Mistral-7B (4bit)	88.1	81.7	81.5
Phi-3-mini (4bit)	82.1	79.4	81.9
Qwen-2-7B (4bit)	85.3	80.8	92.0

Table 1: Results for PE dataset. The first line reports the previous SOTA results. For ACC, the reported results are based on the essay level, with structural tags inserted. For ARI and ARC, the results are based on the paragraph level, without inclusion of structural tags and AC types. Significantly higher scores can be achieved by including markup tags (see Appendix D).

ARI: For both PE and CDCP datasets (the AbstRCT dataset being evaluated on the joint ARIC task), our method significantly improves upon the previous SOTA. We conjecture that LLMs are particularly effective at determining which ACs are related based on their relative positions in the text.

In the PE dataset, as argument relations occur within a paragraph, the ARI task is formulated at the paragraph level (Stab and Gurevych, 2017). Consistent with this formulation, the ARI task performs better at the local paragraph level than at

[1]These modalities are not applicable to AbstRCT and CDCP datasets, as they do not have a paragraph structure.

the more global essay level. The ablation study in Appendix D shows that the inclusion of markup tags drastically improves the results for this task. These modalities are considered less applicable for a practical AM pipeline, and therefore not reported as the main results.

Model	ACC	ARIC
Liu et al. (2023)	92.7 / 94.1 / 92.8	74.3 / 73.9 / 75.0
LLaMA-3-8B (4bit)	91.8 / 95.8 / 94.1	71.9 / 72.9 / 76.5
LLaMA-3-8B	92.6 / **95.9** / **94.2**	71.7 / **75.7** / 76.5
LLaMA-3-70B (4bit)	92.5 / 94.9 / 93.6	74.0 / 73.0 / **77.1**
LLaMA-3.1-8B (4bit)	92.4 / 95.7 / 93.2	72.3 / 73.8 / 76.8
Gemma-2-9B (4bit)	93.0 / 95.7 / 93.6	74.5 / 70.0 / 75.7
Mistral-7B (4bit)	**93.7** / 95.1 / 93.4	**74.9** / 71.9 / 75.1
Phi-3-mini (4bit)	91.4 / 94.6 / 93.3	66.3 / 66.8 / 71.1
Qwen-2-7B (4bit)	61.8 / 95.1 / 94.0	69.1 / 70.7 / 74.0

Table 2: Results for AbstRCT dataset on the 3 test sets gla / mix / neo. The first line reports the previous SOTA results.

ARC: For ARC also, for both PE and CDCP datasets, our method drastically improves upon the previous SOTA. We also attribute this performance to the strong capabilities of LLMs in predicting the AR types based on the positions of ACs in the argumentative flow.

For PE dataset, better results are obtained at the paragraph level, for the same reasons mentioned earlier. However, there is no clear evidence that the addition of structural tags improves the results (see Appendix D for further details).

Model	ACC	ARI	ARC
Bao et al. (2021)	82.5	67.8	–
LLaMA-3-8B (4bit)	85.2	70.6	73.5
LLaMA-3-8B	85.8	69.3	64.6
LLaMA-3-70B (4bit)	87.0	72.2	76.7
LLaMA-3.1-8B (4bit)	**87.3**	**72.3**	**80.4**
Gemma-2-9B (4bit)	78.4	71.2	72.1
Mistral-7B (4bit)	77.7	70.2	54.9
Phi-3-mini (4bit)	83.1	61.0	76.5
Qwen-2-7B (4bit)	79.1	65.8	58.5

Table 3: Results for CDCP dataset. The first line reports the previous SOTA results.

ARIC: For AbstRCT dataset (the only one evaluated on ARIC), our approach also improves upon the previous SOTA. However, we cannot assess whether grouping the ARI and ARC tasks into ARIC affects their individual performance. A thorough comparison of the individual and joint task approaches is deferred to future work.

Dataset	ACC	ARI	ARC
PE	87.5 / 87.9	80.7 / 80.6	82.9 / 79.9
AbstRCT	93.0 / 95.4 / 94.0	84.7 / 84.0 / 84.5	62.6 / 66.1 / 65.2
CDCP	83.2	69.6	41.0

Table 4: Results for the joint ACC–ARI–ARC task obtained by fine-tuning the model LLaMA-3-8B (4bit). For PE, the results are without/with including structural tags. For AbstRCT, the results are for the gla/mix/neo test sets.

ACC–ARI–ARC: The joint ACC–ARI–ARC task implements a multi-task setting that eliminates the need for multiple fine-tuned models. The results for this task are reported in Table 4. For computational efficiency, only the LLaMA-3-8B (4bit) model was utilized. Consistent with the findings of Kuribayashi et al. (2019), Bao et al. (2021), and Liu et al. (2023), the joint task modeling negatively impacts the performance of the individual tasks.

Ablation study: For the PE dataset, we conducted an ablation study for LLaMA-3-8B and LLaMA-3-8B (4bit) to evaluate the impact of using an essay-level versus a paragraph-level approach (essay/paragraph) and the effect of including or excluding (1/0) structural tags and component type tags (the latter applying only to ARI and ARC). The results are reported in Table 5.

Both paragraph and essay modalities capture the sequentiality of ACs in the argumentative flow, but at different scales. The inclusion of markup tags provides contextual and structural features, which are known to be crucial for enhancing performance on AM tasks (Stab and Gurevych, 2017; Kuribayashi et al., 2019; Mushtaq and Cabessa, 2022).

For ACC, there is no clear pattern indicating which contextual scale performs best. However, adding markup tags appears to slightly improve performance at the paragraph level, but not at the essay level. In contrast, the ARI and ARC tasks are inherently modeled at the paragraph level, resulting in significantly higher performance at this scale. Moreover, for ARI, the inclusion of markup tags substantially improves results. Notably, the addition of AC type tags (results from the ACC task) markedly enhances the performance, setting a new SOTA. Conversely, for ARC, the inclusion of tags does not appear to yield improvements. A deeper analysis of these findings is provided in Appendix D.

Model	context	tags	ACC	ARI	ARC
LLaMA 3 8B (4bit)	paragraph	0	87.7	81.2	80.0
LLaMA-3-8b	paragraph	0	86.3	81.0	**89.6**
LLaMA 3 8B (4bit)	paragraph	1	88.2	**83.5** / 92.8 / 93.5	86.8
LLaMA-3-8b	paragraph	1	87.3	83.0 / 92.5 / **93.7**	89.1
LLaMA 3 8B (4bit)	essay	0	86.8	65.6	64.0
LLaMA-3-8b	essay	0	**89.5**	49.7	64.0
LLaMA 3 8B (4bit)	essay	1	87.0	77.0	71.5
LLaMA-3-8b	essay	1	86.3	77.1	64.3
LLaMA 3 70B (4bit)	essay/para/para	1/0/0	87.5	82.0	94.2
LLaMA 3.1 8B (4bit)	essay/para/para	1/0/0	88.5	80.7	95.9

Table 5: Results for PE dataset. For ARI task with mode 'tags=1', the results $x/y/z$ correspond to the three ablation settings where: only the paragraph tags are provided, only the AC type tags are provided, or both the paragraph and the AC type tags are provided, respectively. State-of-the-art results are highlighted in boldface.

4 Conclusion

In this study, we have demonstrated that fine-tuned LLMs achieve state-of-the-art results across the main AM tasks, including ACC, ARI, ARC as well as the joint ARIC and ACC–ARI–ARC tasks. Notably, our fine-tuning approach reaches SOTA results across all datasets and single tasks. It is task-agnostic, removing the need for specialized models, custom architectures, feature engineering, or tailored loss functions. Furthermore, quantized models with less than 10 billion parameters can be run effectively on consumer-grade hardware, making it accessible and practical.

Research in argument mining (AM) has highlighted the importance of incorporating contextual, structural, and syntactic features to effectively address key AM sub-tasks (Stab and Gurevych, 2017). Earlier studies primarily relied on hand-crafted features integrated into classical machine learning (ML) models (Stab and Gurevych, 2017). In contrast, modern approaches, including our LLM fine-tuning approach, allow these features to be learned in an abstract manner from the contextual information provided as input (Kuribayashi et al., 2019; Mushtaq and Cabessa, 2022, 2023). While this shift generally enhances performance, it comes at the expense of reduced explainability.

This work opens up several avenues for future research. First, we aim to address the ARI task, which offers the greatest potential for improvement. To this end, we will search for crafted structural and contextual features that could enhance the results. More generally, to better understand and analyze the reasoning process of LLMs, we plan to integrate a Chain-of-Thought (CoT) component to our prompting and post-processing modules. Transfer learning between AM tasks could also be studied. On a technical-architectural level, we will also investigate the LLM's behavior and feature importance by examining and visualizing the attention heads.

Beyond fine-tuning, we intend to compare our approach to zero-shot and in-context learning (ICL) methods to assess their relative effectiveness. Finally, we plan to study the ability of LLMs to generate complete argumentative structures in one go. This work, along with the aforementioned research directions, will form the core of a detailed follow-up paper.

Limitations

While we experimented with eight popular models, our study could be extended to include additional LLMs of various sizes, context windows and quantization configurations. Currently, SOTA results are achieved using the default hyper-parameters from LLaMA-Factory. A more thorough hyper-parameter search could further enhance the results. In addition, the statistical robustness of our results could be examined in greater detail.

Acknowledgments

This work benefited from access to the computing resources of the Laboratoire Informatique, Image et Interaction (L3i), operated and hosted by the University of La Rochelle. It is financed by the French government and the Nouvelle-Acquitaine Region. This research was partially supported by Czech Science Foundation, grant AppNeCo #GA22-02067S, institutional support RVO: 67985807. Finally, we are grateful to Playtika AI for their support for this research.

References

Abdullah Al Zubaer, Michael Granitzer, and Jelena Mitrović. 2023. Performance analysis of large language models in the domain of legal argument mining. *Frontiers in Artificial Intelligence*, 6.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Jianzhu Bao, Chuang Fan, Jipeng Wu, Yixue Dang, Jiachen Du, and Ruifeng Xu. 2021. A neural transition-based model for argumentation mining. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 6354–6364. ACL.

Jérémie Cabessa, Hugo Hernault, and Umer Mushtaq. 2024. In-context learning and fine-tuning gpt for argument mining. *Preprint*, arXiv:2406.06699.

Elena Cabrio and Serena Villata. 2018. Five years of argument mining: A data-driven analysis. In *Proceedings of IJCAI 2018*, IJCAI'18, pages 5427–5433. AAAI Press.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural end-to-end learning for computational argumentation mining. In *Proceedings of ACL 2017*, pages 11–22, Vancouver, Canada. ACL.

Ivan Habernal and Iryna Gurevych. 2017. Argumentation mining in user-generated web discourse. *Computational Linguistics*, 43(1):125–179.

Shohreh Haddadan, Elena Cabrio, and Serena Villata. 2019. Yes, we can! mining arguments in 50 years of US presidential campaign debates. In *Proceedings of ACL 2019*, pages 4684–4690, Florence, Italy. ACL.

Tatsuki Kuribayashi, Hiroki Ouchi, Naoya Inoue, Paul Reisert, Toshinori Miyoshi, Jun Suzuki, and Kentaro Inui. 2019. An empirical study of span representations in argumentation structure parsing. In *Proceedings of ACL 2019*, pages 4691–4698, Florence, Italy. ACL.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context dependent claim detection. In *ICCL*.

Boyang Liu, Viktor Schlegel, Riza Batista-Navarro, and Sophia Ananiadou. 2023. Argument mining as a multi-hop generative machine reading comprehension task. In *Findings of the ACL: EMNLP 2023, Singapore, December 6-10, 2023*, pages 10846–10858. ACL.

Tobias Mayer. 2020. *Argument Mining on Clinical Trials. (Fouille d'arguments à partir des essais cliniques)*. Ph.D. thesis, Université Côte d'Azur, France.

Tobias Mayer, Elena Cabrio, and Serena Villata. 2020. Transformer-based argument mining for healthcare applications. In *Proceedings of ECAI 2020*, volume 325 of *FAIA*, pages 2108–2115. IOS Press.

Raquel Mochales and Marie-Francine Moens. 2011. Argumentation mining. *Artificial Intelligence and Law*, 19(1):1–22.

Umer Mushtaq and Jérémie Cabessa. 2022. Argument classification with BERT plus contextual, structural and syntactic features as text. In *Proceedings of ICONIP 2022*, volume 1791 of *CCIS*, pages 622–633. Springer.

Umer Mushtaq and Jérémie Cabessa. 2023. Argument mining with modular BERT and transfer learning. In *Proceedings of IJCNN 2023*, pages 1–8. IEEE.

Vlad Niculae, Joonsuk Park, and Claire Cardie. 2017. Argument mining with structured SVMs and RNNs. In *Proceedings of ACL 2017*, pages 985–995, Vancouver, Canada. ACL.

H. Nori et al. 2023. Can generalist foundation models outcompete special-purpose tuning? case study in medicine. *CoRR*, abs/2311.16452.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation mining: The detection, classification and structure of arguments in text. In *Proceedings of ICAIL 2019*, ICAIL '09, pages 98–107, New York, NY, USA. ACM.

Joonsuk Park and Claire Cardie. 2018. A corpus of erulemaking user comments for measuring evaluability of arguments. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).

Mircea-Luchian Pojoni, Lorik Dumani, and Ralf Schenkel. 2023. Argument-mining from podcasts using chatgpt. In *Proceedings of ICCBR-WS 2023*, volume 3438 of *CEUR Workshop Proceedings*, pages 129–144. CEUR-WS.org.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. Here's my point: Joint pointer architecture for argument mining. In *Proceedings of EMNLP 2017*, pages 1364–1373. ACL.

Christian Stab and Iryna Gurevych. 2017. Parsing argumentation structures in persuasive essays. *Computational Linguistics*, 43(3):619–659.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*.

A Datasets

Corpus Statistics		Component Statistics	
Tokens	147,271	major claims	751
Sentence	7,116	claims	1,506
Paragraphs	1,833	premises	3,832
Essays	402	Total	6,089

Table 6: Statistics for PE dataset (available here).

Split	Abstracts	Components
Neo-train	350	2,291
Neo-test	100	691
Gla-test	100	615
Mix-test	100	609

Table 7: Statistics of AbstRCT dataset (available here).

Components		Relations	
policy	815	reason	1174
value	2182	evidence	46
fact	785		
testimony	1117		
reference	32		
Total	4931		1220

Table 8: Statistics for CDCP dataset (available here).

B Implementation details

We experimented with the models reported in Table 9. All experiments were carried out using the LLaMA-Factory Python library (Zheng et al., 2024), with the QLoRa fine-tuning approach (Dettmers et al., 2023). Except for Llama-3 8B, all other base models employed are 4-bit quantized checkpoints, which are all freely available from Unsloth on Hugging Face. We kept the default hyper-parameters of LLaMA-Factory (cf. Table 10), did not perform any hyper-parameter tuning, and trained the models for 5 epochs on a single NVIDIA RTX A6000 (48GB). The average training and inference time of PE dataset at the paragraph and essay levels was approximately 40 and 15 minutes, respectively.

C Task Evaluation and Error Analysis

In this section, we describe the evaluation and error handling processes for the four tasks: ACC, ARI, ARC, and ARIC. For each task, the predictions generated by the model are automatically post-processed and the corresponding classification metrics then computed. To this end, we implemented *hallucination guardrails* to filter out predictions with incorrect formats or lengths.

Checkpoint	Model
llama-3-8b-Instruct-bnb-4bit	LLaMA 3 8B (4bit)
llama-3-8b-Instruct	LLaMA 3 8B
llama-3-70b-Instruct-bnb-4bit	LLaMA 3 70B (4bit)
Meta-Llama-3.1-8B-Instruct-bnb-4bit	LLaMA 3.1 8B (4bit)
gemma-2-9b-it-bnb-4bit	Gemma 2.9 (4bit)
mistral-7b-instruct-v0.3-bnb-4bit	Mistral 7B (4bit)
Phi-3-mini-4k-instruct-bnb-4bit	Phi 3 mini (4bit)
Qwen2-7B-Instruct-bnb-4bit	Qwen 2 7B (4bit)

Table 9: Checkpoints and names of the LLMs.

Parameter	Value
num_train_epochs	5
per_device_train_batch_size	2
gradient_accumulation_steps	4
learning_rate	5e-5
lr_scheduler_type	"cosine"
warmup_ratio	0.1
max_grad_norm	1.0
finetuning_type	"lora"
lora_target	"all"
quantization_bit	4
loraplus_lr_ratio	16.0
fp16	True

Table 10: Hyper-parameters of the models (default from LLaMA-Factory).

C.1 ACC

As described in Section 2.2, for the ACC task, the model's prompt instructs the generation of a list of component types of the form

```
component_types_list = [component_type (str),...,
component_type (str)]
```

whose length corresponds to the correct number of argument components (ACs) to be predicted, and whose elements must belong to the set of possible AC types (e.g., `"claim"` or `"premise"`) (see Section E).

Generally, the model adhered to these length and format constraint. In rare cases where the model generated a list that was too short (or too long), incorrect predictions were automatically added to (or removed from) the prediction list so that its length matches the number of ACs to be predicted. Similarly, when the model generated predictions with incorrect format (e.g. "none", 3, etc.), the latter were replaced by wrong predictions but of the correct format (e.g., `"claim"` or `"premise"`). In the extremely rare cases where the model generated a non-parsable list (e.g., incorrect JSON format due to missing closing brackets), the latter was manu-

ally corrected. With these guardrails in place, the post-processing of the results was achieved almost entirely automatically. The same remark applies for the next tasks.

Then, the list of post-processed predictions lists associated with the test set is flattened into a single list `preds_final`. Similarly, the list of ground truth lists from the test set is flattened into a single list `grounds_final`. Finally, the classification metrics are computed based on these two flattened lists, yielding F1 scores for the possible AC types (e.g., `"claim"` and `"premise"`).

C.2 ARI

For the ARI task, the prompt of the model instructs the generation of a list of pairs of the following form

```
argument_relations_list = [[source_AC (int),
target_AC (int)],..., [source_AC (int),
target_AC (int)]]
```

where each pair `[i,j]` represents a directed relation (e.g., `"support"` or `"attack"`) between the ACs `i` and `j` (see Section E).

The post-processing of the model's predictions is conducted as follows: all possible pairs `[i,j]` of distinct ACs are computed, where `i` $\neq$ `j` and $1 \leq$ `i,j` $\leq$ `nb_acs`. For each pair `[i,j]`, if the latter was generated (resp. not generated) by the model, then the triplet `[i,j,"REL"]` (resp. `[i,j,"N-REL"]`) is appended to a list `preds`. In this way, any pair `[i,j]` of incorrect format – i.e., `i` ≤ 0 or `j` ≤ 0 or `i` $>$ `nb_acs` or `j` $>$ `nb_acs` or `i` or `j` being not of type `int` – is never appended to the list `preds`, simulating the user's ability to detect and reject invalid predictions. The same process is applied to the ground truth pairs, yielding a list of triplets `grounds`.

Afterwards, the list of post-processed predictions lists associated with the test set is flattened into a single list `preds_final` composed of elements `"REL"` and `"N-REL"`. Similarly, the list of post-processed ground truth lists from the test set is flattened into a single list `grounds_final` composed of elements `"REL"` and `"N-REL"`. Finally, classification metrics are computed based on these two flattened lists, yielding F1 scores for the two classes `"REL"` and `"N-REL"`.

C.3 ARC

For the ARC task, the prompt of the model instructs the generation of a list of relation types

```
relation_types_list = [relation_type (str),...,
relation_type (str)]
```

whose length corresponds to the correct number of relations to be predicted, and whose elements must belong to the set of possible relation types (e.g., `"support"` or `"attack"`) (see Section E). The post-processing of the predictions and the computation of the classification metrics follow the exact same pattern as for the ACC task, yielding F1 scores for the possible relation types `"support"` and `"attack"` (PE and AbstRCT datasets) or `"reason"` and `"evidence"` (CDCP dataset).

C.4 ARIC

The ARIC task corresponds to the joint ARI and ARC tasks. In this case, the model's prompt instructs the generation of a list of triplets

```
arguments_relation_types_list = [[source_AC
(int), target_AC (int), relation_type (str)],...,
[source_AC (int), target_AC (int), relation_type
(str)]]
```

where each triplet `[i,j,x]` represents a directed relation between ACs `i` and `j` of type `x` (e.g. `"support"` ot `"attack"`) (see Section E).

The post-processing of the model's predictions is conducted according to the method described by (Liu et al., 2023). All possible pairs `[i,j]` of distinct ACs are computed, where `i` $\neq$ `j` and $1 \leq$ `i,j` $\leq$ `nb_acs`. For each pair `[i,j]`, if a triplet of the form `[i,j,x]` was generated by the model, where $1 \leq$ `i,j` $\leq$ `nb_abs` and `x` is of a correct format (e.g., `"reason"` or `"evidence"`), then the triplet `[i,j,x]` is appended to a list `preds`. Otherwise, the triplet `[i,j,"None"]` is appended to the list `preds`. In this way, any triplet `[i,j,x]` of incorrect format is never appended to the list `preds`, simulating the user's ability to discard invalid predictions. The same process is applied to the ground truth pairs, yielding a list of triplets `grounds`.

Afterwards, the list of post-processed predictions lists associated with the test set is flattened into a single list `preds_final` composed of elements `"reason"`, `"evidence"` and `"None"` (CDCP dataset). Similarly, the list of post-processed ground truth lists from the test set is flattened into a single list `grounds_final` composed of elements `"reason"`, `"evidence"` and `"None"`. Finally, classification metrics are computed based on these two flattened lists, yielding F1 scores for the classes `"reason"`, `"evidence"` and `"None"` (CDCP dataset).

D Ablation Study for PE Dataset

As described in Section 2, for PE dataset, the two following fine-tuning modalities are considered: (i) Paragraph/Essay level: The LLMs are trained and tested on data samples consisting of either individual paragraphs or full essays, respectively; (ii) With/Without structural tags: At the essay level, markup tags of the form

`<paragraph_type>. . . </paragraph_type>`

delimiting the topic, introduction, body paragraphs and conclusion of the essays are inserted in the text. For the ARI and ARC tasks, the argument components' types ('major claim', 'claim' or 'premise') can further be given as tags of the form

`<ACn, type>. . . </ACn, type>`.

We conducted an ablation study to evaluate the impact of using an essay-level versus a paragraph-level approach (essay/paragraph) and the effect of including versus excluding (1/0) structural tags and component type tags (the latter applying only to ARI and ARC). The results are reported in Table 5.

Both paragraph and essay modalities capture the sequentiality of ACs in the argumentative flow, but at different scales. The inclusion of markup tags provides contextual and structural features, which are known to be crucial for enhancing performance on AM tasks (Stab and Gurevych, 2017; Kuribayashi et al., 2019).

ACC. There is no clear pattern indicating which contextual scale performs best. We conjecture that the essay level is beneficial for this task, as the argumentative flow extends throughout the entire essay. The injection of markup tags seems to improve results at the paragraph level, but not at the essay level. The structural information conveyed by the tags would thus be able to boost performance in the case of limited contextual scale.

ARI. In its original formulation, the ARI task involves the identification of argument relations within paragraphs, where these relations occur. Naturally, solving the ARI task at the essay level is more challenging than at the paragraph level, as it involves more pairs of ACs. These considerations explain the lower scores obtained at the essay level.

At the paragraph level, the addition of markup tags significantly improves the results. Notably, the inclusion of structural tags alone achieves SOTA results of 83.5, highlighting the importance of structural information for this task. Furthermore, the addition of AC type tags alone results in a remarkable boost to 92.8, indicating that the related/non-related nature of ARs strongly depends on the types of their constituent ACs. In a practical AM pipeline, these true AC types, which are typically unknown, could be replaced by predictions from the previous ACC task to further enhance performance. Finally, combining both types of tags leads to an additional improvement to 93.7, representing a significant advance over the previous SOTA of 82.7.

ARC. The ARC task is also inherently modeled at the paragraph level, which explains the significantly poorer results obtained at the essay level.

At the paragraph level, we achieve SOTA results of 89.6, which substantial improves over the previous SOTA of 81.0. In this case, there is no clear evidence that adding tags improves the results. This suggests that, once ARs are identified, their supporting or attacking nature depends primarily on the textual content of their constituent ACs, rather than on the types of these ACs or the types of paragraphs in which they are located.

E Prompts

Below, we present examples of training samples used in the fine-tuning of our models, covering PE, CDCP, and AbstRCT datasets and their corresponding tasks ACC, ARI, ARC and ARIC and joint. For each task, the prompt instructs the model to generate predictions list(s) with correct formats and lengths.

Example 1. Training sample for PE dataset, ACC task, with structural tags at the essay level.

instruction ### You are an expert in Argument Mining. You are given an essay which contains numbered argument components enclosed by <AC></AC> tags. Your task is to classify each argument components in the essay as either "MajorClaim", "Claim" or "Premise". You must return a list of argument component types in following JSON format: "component_types": [component_type (str), component_type (str), ..., component_type (str)]

input ### Here is the essay text: <topic> Technology inspires children ' s creativity instead of weakening it </topic><para-intro> Technology accelerates human ' s evolving pace . With advanced technology , many things that seemed impossible in the past , have become realities . For example , people in the past never dreamed of talking to anyone whenever they wanted or see someone overseas on a computer screen , both of these can be achieved with cellphone and internet . However , some people point that technology has made children less creative . I don ' t deny that <AC0> technology may have some negative effects on children </AC0> , but I think <AC1> technology makes children even more creative </AC1> . </para-intro><para-body> First , <AC2> technology inspires children to create new things </AC2> . <AC3> Children are curious about everything around them </AC3> , so <AC4> when they come across a high - tech

product like a cellphone , they will be obsessed with its mysterious functions and eager to know how it works </AC4> . For example , <AC5> Bill Gates was attracted by the original huge computer , then he did everything he could to understand how it worked </AC5> . <AC6> After he had figured out all the stuff , he then began to promote the computer </AC6> . <AC7> In the end , he successfully invented a computer that was easy to use and lightweight </AC7> . Therefore , <AC8> instead of interfering children ' s creativity , technology actually encourages children to learn and to create </AC8> . </para-body><para-body> Second , <AC9> technology widen children ' s knowledge </AC9> . <AC10> In the past , children were only able to see things from one perspective </AC10> . However , <AC11> with highly advanced technology , children are able to get information from foreign countries and even communicate with foreign friends </AC11> . Therefore , <AC12> they will be able to learn about different cultures and different ways of thinking </AC12> . <AC13> Knowledge is the base of creativity </AC13> . <AC14> The diverse knowledge that children gained from different parts of the world , inspires children and encourage them to create new things </AC14> . </para-body><para-conclusion> Admittedly , <AC15> technology may have negative effects on children </AC15> . <AC16> Technology has made life too easy for most children </AC16> . <AC17> In comfortable conditions , children may be satisfied with the world around them and refuse to make a change </AC17> . Nonetheless , <AC18> with the development of technology , increasingly more young people become obsessed with high - tech products </AC18> . <AC19> They kept researching new technology and became successful at a very young age </AC19> . </para-conclusion>

output "component_types": ["Claim", "MajorClaim", "Premise", "Premise", "Premise", "Premise", "Premise", "Premise", "Claim", "Premise", "Premise", "Premise", "Premise", "Premise", "Claim", "Claim", "Premise", "Premise", "Premise", "MajorClaim"]

Example 2. Training sample for CDCP dataset, ARI task.

instruction ### You are an expert in Argument Mining. You are given a text which contains numbered argument components enclosed by <AC></AC> tags. Your task is to identify argument relations between argument components in the text. You must return a list of pairs in the following JSON format: "list_argument_relations": [[source AC (int), target AC (int)], ..., [source AC (int), target AC (int)]]

input ### Here is the text: <AC0>Calling a debtor at work is counter-intuitive;</AC0><AC1> if collectors are continuously calling someone at work, other employees may report it to the debtor's supervisor.</AC1><AC2> Most companies have established rules about receiving or making personal calls from company or cell phones during an employee's working hours.</AC2><AC3> If a collector or creditor calls a debtor on his/her cell phone and is informed that the debtor is at work, the call should be terminated.</AC3><AC4> No calls to employers should be allowed </AC4><AC5> as this jeopardizes the debtor's job.</AC5><AC6> How does that help in debt collection? It doesn't.</AC6>

output "list_argument_relations": [[0, 1], [1, 2], [3, 2], [4, 5]]

Example 3. Training sample for CDCP dataset, ARC task.

instruction ### You are an expert in Argument Mining. You are given a text which contains numbered argument components enclosed by <AC></AC> tags. You are also given a list of pairs of related argument components in the form: [(target AC (int), source AC (int)), (target AC (int), source AC (int)), ..., (target AC (int), source AC (int))]. Your task is to classify each pair of related argument components in the list as either "reason" or "evidence". You must return a list of argument relation types, strictly of length 5, in following JSON format: "relation_types": ["relation_type (str)", "relation_type (str)", "relation_type (str)", "relation_type (str)", "relation_type (str)"] where each element "relation_type (str)" is replaced by either "reason" or "evidence".

input ### Here is the text: <AC1>Debt collectors continuously resell the debt to other debt collectors who start the clock all over again.</AC1><AC2> Then that debt collector resells the debt </AC2><AC3> and the 3rd debt collector restarts the clock again, and so on.</AC3><AC4> The burden of proof is put on the consumer to prove it is an old debt.</AC4><AC5> The credit reporting agencies don't automatically remove old debts </AC5><AC6> nor do they check to see if a newly reported debt is in fact a 9 year old debt that has been resold numerous times.</AC6><AC7> The credit agencies (CRA) are more of a problem for consumers than the debt collectors.</AC7><AC8> The CRAs are paid by the credit card companies</AC8><AC9> and the credit card companies have bigger profits when they can charge higher rates based on poor credit scores.</AC9><AC10> So there is inherently a huge conflict of interest here.</AC10><AC11> If the CRAs can keep our credit scores down the card companies can make more money and don't mind paying the CRAs a piece of the action.</AC11><AC12> Seems like unspoken collusion to me.</AC12> ### Here is the list of pairs of related argument components in this paragraph: [[3, 4], [3, 5], [6, 3], [9, 7], [9, 8]]

output "relation_types": ["reason", "reason", "reason", "reason", "reason"]

Example 4. Training sample for AbstRCT dataset, ARIC task.

instruction ### You are an expert in Argument Mining. You are given a biomedical abstract text which contains numbered argument components enclosed by <AC></AC> tags. Your task is to identify argument relations between argument components in the abstract text and classify their relation type as either "support" or "attack". You must return a list of triplets in the following JSON format: "list_argument_relation_types": [[source AC (int), target AC (int), relation_type (str)], ..., [source AC (int), target AC (int), relation_type (str)]] where each element "relation_type (str)" is replaced by either "support" or "attack".

input ### Here is the abstract text: <AC1> Treatment with cisplatin-based chemotherapy provides a modest survival advantage over supportive care alone in advanced non-small-cell lung cancer (NSCLC). </AC1>To determine whether a new agent, paclitaxel, would further improve survival in NSCLC, the Eastern Cooperative Oncology Group conducted a randomized trial comparing paclitaxel plus cisplatin to a standard chemotherapy regimen consisting of cisplatin and etoposide. The study was carried out by a multi-institutional cooperative group in chemotherapy-naive stage IIIB to IV NSCLC patients randomized to receive paclitaxel plus cisplatin or etoposide plus cisplatin. Paclitaxel was administered at two different dose levels (135 mg/m(2) and 250 mg/m(2)), and etoposide was given at a dose of 100 mg/m(2) daily on days 1 to 3. Each regimen was repeated every 21 days and each included cisplatin (75 mg/m(2)). The characteristics of the 599 patients were well-balanced across the three treatment groups.<AC2> Superior survival was observed with the combined paclitaxel regimens (median survival time, 9.9 months; 1-year survival rate, 38.9%) compared with etoposide plus cisplatin (median survival time, 7.6 months; 1-year survival rate, 31.8%; P =. 048). </AC2><AC3> Comparing survival for the two dose levels of paclitaxel revealed no significant difference. </AC3><AC4> The median survival duration for

the stage IIIB subgroup was 7.9 months for etoposide plus cisplatin patients versus 13.1 months for all paclitaxel patients (P =.152). </AC4><AC5> For the stage IV subgroup, the median survival time for etoposide plus cisplatin was 7.6 months compared with 8.9 months for paclitaxel (P =.246). </AC5><AC6> With the exceptions of increased granulocytopenia on the low-dose paclitaxel regimen and increased myalgias, neurotoxicity, and, possibly, increased treatment-related cardiac events with high-dose paclitaxel, toxicity was similar across all three arms. </AC6><AC7> Quality of life (QOL) declined significantly over the 6 months. </AC7><AC8> QOL scores were not significantly different among the regimens. </AC8>As a result of these observations,<AC9> paclitaxel (135 mg/m(2)) combined with cisplatin has replaced etoposide plus cisplatin as the reference regimen in our recently completed phase III trial. </AC9>

output "list_argument_relation_types": [[2, 9, "support"], [4, 9, "support"], [5, 9, "support"], [6, 9, "support"], [8, 9, "support"]]"

Example 5. Training sample for CDCP dataset, joint ACC–ARI–ARC task.

instruction ### You are an expert in Argument Mining. You are given a text which contains numbered argument components enclosed by <AC></AC> tags. Your task is to classify the argument components in the text as well as to identify and classify argument relations between the argument components. For each argument component, its component_type (str) is either "fact", "policy", "reference", "testimony" or "value". For each argument relation (source_component (int), target_component (int)), its relation_type (str) is either "evidence" or "reason". You must return two lists called "component_types" and "argument_relations_and_types". The first list "component_types", strictly of length 16, must be in following JSON format: "component_types": ["component_type (str)", "component_type (str)", "component_type (str)", "component_type (str)", "component_type (str)", "component_type (str)", "component_type (str)", "component_type (str)", "component_type (str)", "component_type (str)", "component_type (str)", "component_type (str)", "component_type (str)", "component_type (str)", "component_type (str)", "component_type (str)"] where each element "component_type (str)" is replaced by either "fact", "policy", "reference", "testimony" or "value". The second list "argument_relations_and_types" must be a list of triplets in the following JSON format: "argument_relations_and_types": [[source_component (int), target_component (int), relation_type (str)], ..., [source_component (int), target_component (int), relation_type (str)]] where each element "relation_type (str)" is replaced by either "evidence" or "reason".

input ### Here is the text: <AC1>The agency needs to be careful of new rules that will, though unintended, be harmful to the free market.</AC1><AC2> However, it is important that litigation is monitored to ensure that the system does not allow reward with no risk.</AC2><AC3> A plaintiff in a lawsuit cannot be placed in a situation where it is rewarding to sue with insufficient cause and minimal risk.</AC3><AC4> Creditors and those following in the process of recovery do not need additional regulation in this area</AC4><AC5> since their efforts are monitored by the CFPB.</AC5><AC6> The CFPB does not need to set standards for the industry,</AC6><AC7> but make it clear that only those documents provided to the court in the original filing will be considered in a CFPB review.</AC7><AC8> If in the course of a CFPB review, it is found that litigation was filed without a successful verification of the facts, the CFPB could award the amount sought by the filing to the defendant of the case and a fine to the CFPB of no less than $2,000 but no more than the amount sought.</AC8><AC9> This position would motivate the litigant to ensure their facts were correct before any filing took place.</AC9><AC10> This position would allow CFPB to review cases filed no matter how they were adjudicated by the involved court.</AC10><AC11> This review could be conducted at CFPB offices</AC11><AC12> since it only requires the original filings.</AC12><AC13> This review could be initiated simply based on a complaint to the agency's offices.</AC13><AC14> The penalty could be adjustable based on a variety of agency standards.</AC14><AC15> I would have to take more time to consider when the plaintiff was not the original creditor or a member of the recovery industry,</AC15><AC16> but, again, there must be a risk associated with the reward to stop nuisance suits.</AC16>

output "component_types": ["policy", "value", "fact", "value", "fact", "value", "policy", "value", "value", "value", "value", "value", "value", "value", "value", "value"], "argument_relations_and_types": [[4, 5, "reason"], [8, 9, "reason"], [8, 10, "reason"], [11, 12, "reason"]]