Actor Identification in Discourse: A Challenge for LLMs?

Ana Barić[*†] and **Sebastian Padó**[*] and **Sean Papay**[*]
[*]: IMS, University of Stuttgart, Stuttgart, Germany
[†]: TakeLab, FER, University of Zagreb, Croatia
{ana.baric,sebastian.pado,sean.papay}@ims.uni-stuttgart.de

Abstract

The identification of political actors who put forward claims in public debate is a crucial step in the construction of *discourse networks*, which are helpful to analyze societal debates. Actor identification is, however, rather challenging: Often, the locally mentioned speaker of a claim is only a pronoun (*"He proposed that [claim]"*), so recovering the *canonical* actor name requires discourse understanding. We compare a traditional pipeline of dedicated NLP components (similar to those applied to the related task of coreference) with a LLM, which appears a good match for this generation task. Evaluating on a corpus of German actors in newspaper reports, we find surprisingly that the LLM performs worse. Further analysis reveals that the LLM is very good at identifying the right reference, but struggles to generate the correct *canonical form*. This points to an underlying issue in LLMs with controlling generated output. Indeed, a hybrid model combining the LLM with a classifier to normalize its output substantially outperforms both initial models.

1 Introduction

Political decision-making in democracies is generally preceded by political debates taking place in parliamentary forums (committees, plenary debates) or different public spheres (e.g., newspapers, television, social media). One way in which political scientists have analyzed such processes is to adopt the framework of political claims analysis (Koopmans and Statham, 1999), identifying the *claims* (i.e., calls for or against specific courses of action) and *actors* involved in a given debate. Actors, claims, and the relations between them can then be represented as bipartite *discourse networks* (Leifeld and Haunss, 2012; Leifeld, 2016), such as shown in Figure 1. Such networks permit researchers to investigate debates on a fine-grained level, identifying, e.g., discourse coalitions, decision makers, or argumentative clusters.

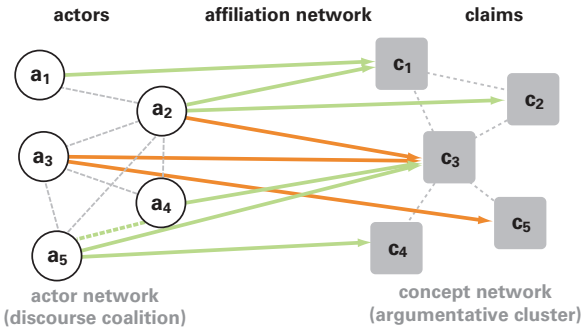

Figure 1: Discourse network with actors as circles and claims as squares (adapted from Padó et al., 2019)

While early work on discourse networks was based on manual analysis, widespread use of discourse networks requires quick, ideally automatic, methods to construct them from text. This calls for NLP methods to (1) detect instances of claims, assign them to their categories (c_i in Figure 1), and (2) identify actors for these claims in terms of some canonical representation (a_i), cf. Padó et al. (2019).

At least for newswire, there are several NLP models for claim detection and categorization (Subramanian et al., 2018; Padó et al., 2019). In contrast, there is little work on actor identification. Arguably, this is because claims are easier to handle: Both detection and categorization are sentence-level classification tasks which can be modeled based on predominantly sentence-internal features. In contrast, actor identification calls for a substantial amount of discourse understanding: models must *locally* identify an actor for the claim, but since these are often just a pronoun or a definite description (cf. Table 1), they must *globally* find a reasonable canonical representation for that actor.

This paper asks whether this situation has improved with the emergence of prompt-based LLMs (Liu et al., 2023) and their promise for text-to-text generation, which appears to be a good match for the actor identification task. We contrast an LLM-based architecture with a traditionally trained

Proceedings of the 5th Workshop on Computational Approaches to Discourse (CODI 2024), pages 64–70

	Local mention of actor	Canonical version
1	*President Joe Biden* pleaded with Republicans …	Joe Biden
2	*Biden* signaled a willingness to make significant changes …	Joe Biden
3	"We can't let Putin win", *he* said.	Joe Biden
4	However, *Senate Republicans* later on Wednesday blocked …	Senate Republicans
5	A *U.S. official* said Washington had less than $1B …	U.S. official

Table 1: Actor mentions and their canonicalizations in newswire article (https://shorturl.at/WZ159)

pipeline of dedicated NLP components on a German dataset with actor-claim annotation (Blokker et al., 2023). We find that, surprisingly, the traditional architecture outperforms the LLM. Our error analysis shows that the LLM often identifies the correct actor entity, but fails to generate the canonical actor name. We attribute this to the general difficulty in controling what exactly LLMs generate, a problem which has given rise to a substantial body of work (Zheng et al., 2023). In line with this interpretation, we show that combining the LLM with the traditional model (for post-processing) achieves substantially better performance on the actor identification task than either model alone.

2 Methods

2.1 Actor Identification: Task Definition

Table 1 shows mentions of actors making claims in a newswire article and the canonical actors they refer to, i.e., input–output pairs for actor mapping.

One possible approach is to treat this task as entity linking (Sevgili et al., 2022), typically realized as classification where the classes are the set of entities from a knowledge base (KB) such as Wikidata. While frequent actors (cf. lines 1–3) are mostly represented in such KBs, texts also introduce ad-hoc actors through plurals (line 4) or unspecific descriptions (line 5) which are generally not part of KBs. That rules out pure entity linking.

Instead, we formalize actor identification directly as *canonical name string prediction*: Models are presented with a claim, along with its context within an article, and are tasked with predicting a string representing that actor. For actors which commonly recur across claims, this string will be a canonical form of the actor's full name, while for singleton actors, this string will be the verbatim realization of an actor mention from the article.

While this formalization seems to ignore much of the structure of the task (after all, actor names are not fundamentally arbitrary strings), it has the benefit of allowing fair comparisons between vastly different model architectures: Text generation models can produce short strings directly, and other modeling approaches can take advantage of task structure internally, while still outputting a string. For example, we could approach the task with a coreference model, extended with a component which chooses the most canonical realization in each coreference chain from among the mentions.[1]

2.2 A Traditional Pipeline Architecture

The first method we apply to this task is a pipeline of two "traditional" NLP approaches: an entity extractor for actor mentions, and a classifier for associating mentions with canonical actor names.

Our mention extractor is a CRF-based sequence labeler. As input, we provide full articles in which the target claim has been marked and encode the input with a pretrained XLM-RoBERTa encoder (Conneau et al., 2020), which we fine-tune during training. The CRF's task is to extract mentions of the actor for the marked claim. As each claim must have at least one actor mention, we constrain (Papay et al., 2022) our CRF to always predict at least one actor mention. In order to map actor mentions to canonical forms, we employ a simple neural classifier based on the same XLM-RoBERTa encoder as above. As classes, we use the set of all canonical actor names which occur at least twice in the training partition of our data (see Section 3.1), along with a special 'verbatim' class for the remaining cases. In these cases, the string output we predict is the exact text of the actor mention.

2.3 An LLM-Based Architecture

In our LLM-based approach, we treat actor identification as an end-to-end task by combining the subtasks of actor detection and mapping within the prompt to directly predict the canonicalized actor.

[1]We do not evaluate a coreference model since full coreference is known to be a very hard task (see, e.g., Peng et al., 2015) and actor identification only requires solving a subpart.

Due to the limited availability of language-specific LLMs, we opted to experiment with the Llama 2 language model (Touvron et al., 2023) for both base- and instruction model options in all available size variants. This model family could be used on German, despite being predominantly trained on English corpora, because of the cross-lingual transferability that is shown to occur in such multilingual LLMs (Choenni et al., 2023).

We assess this task in zero- and few-shot settings, employing current best practices for robust prompt construction. These include: (1) using different instruction paraphrases for prompt templates, given the fact that 'canonical name' is not a very established concept (cf. Appendix A); (2) selecting exemplars semantically similar to the input (Margatina et al., 2023); and (3) varying exemplar quantity and order within the prompt (Lu et al., 2022). We construct the prompts by combining the English task description as prompt instruction with the preprocessed article in German (again, cf. Appendix A). Due to the context length limitation, we preprocess articles by extracting the target claim, marked with special tags, with its surrounding context at the sentence level. We use greedy decoding.

In these trials, zero-shot Llama-2-70b-chat outperforms all few-shot settings. We choose this setting for the rest of the paper.

3 Experimental Setup

3.1 Data

As gold standard for our studies we use DEbateNet (Blokker et al., 2023), a German large corpus resource for the analysis of the domestic debate on migration in Germany in 2015. After domain experts from political science developed a codebook for the policy domain, roughly 700 newspaper articles from the German left-wing quality newspaper "taz – die tageszeitung" with a total of over 550,000 tokens were annotated for actors, claims, and their relations. For each article, all claims are marked and labeled, and each claim is associated with a canonical actor (our gold standard), yielding a collection of about 1,800 actor-attributed claims. Most claims are also associated with a named entity mention from the vicinity of the claim, though this may not be the nearest mention, cf. Table 1. We use the established DEbateNet train–dev–test split, with 1383 claims in train, 220 in dev, and 207 in test.

	Evaluation	Pr	Re	F_1
LLM	exact match	42.66	43.46	43.06
	up to formatting	43.56	44.39	43.98
	up to canonic.	62.39	63.55	62.96
dedicated pipeline	exact match	48.66	59.35	53.47
	up to formatting	48.66	59.35	53.47
	up to canonic.	54.79	66.82	60.21

Table 2: Results for the LLM and traditional pipeline models in the different evaluation settings

3.2 Evaluation

Both models are evaluated and compared via F_1-score. In order to gain a more detailed understanding, we use three evaluation settings:

In the strictest *exact-match* setting, predictions are counted as correct only if they exactly match the gold-standard actor string. This setting can be performed automatically.

In our *correct-up-to-formatting* setting, predictions are counted as correct if they match the gold standard string modulo text formatting differences (e.g. whitespace differences, capitalization, punctuation). This setting tells us how often a model is "almost right" but receives no credit in the strict setting. We carry out this evaluation manually.

Finally, our *correct-up-to-canonicalization* setting counts predictions as correct if they predict the correct entity, even if a different referring expression is generated. For example, "the chancellor" or "Merkel" would be considered correct predictions for the gold-standard actor "Angela Merkel." As with before, this evaluation is performed manually.

4 Results and Analysis

Main results. Table 2 summarizes the performance of our two models under our three evaluation settings. We first consider our strictest setting, exact match. We find results in the range of 40–50 points F_1 score, in line with the assumption that actor mapping is a difficult task. Both models have somewhat higher recall than precision, and the dedicated pipeline outperforms the LLM by 10 point F_1 score. This is somewhat surprising, given LLMs' well-known capabilities in instruction-following text generation (Brown et al., 2020; Webson and Pavlick, 2022; Zhou et al., 2023).

We form two non-mutually exclusive hypotheses for this performance gap: either that the traditional model, through its supervised training, came to be

more competent at predicting the *correct* political actor, or, through virtue of its inductive biases, it came to better and predicting the *exact* canonical name. We examine these hypotheses by evaluating the model with the other two settings. We also carry out a qualitative analysis of errors made by the LLM-based model (see Table 3).

One simple factor that would lead an essentially correct LLM to be inexact is formatting errors in its output – either mismatched spacing, punctuation, or capitalization, or natural language responses that could not be correctly post-processed. Such effects should show up as a difference between the 'exact match' and the 'up to formatting' setting. However, the numbers (43.06 F_1 vs. 43.98 F_1) show that these types of error account for less than one percentage point. Our qualitative error analysis (Table 3, top part) finds (few) cases of formatting errors, which often co-occur with other problems (unexpected LLM responses, gold standard errors). We conclude that such errors have a relatively minor effect on performance.

The reliance of our exact evaluation metric on gold-standard canonical forms provides another opportunity for a largely correct model to show low performance due to an inability to pick the exact canonical form required. This factor should come to the fore when we compare exact match results to the 'up-to-canonicalization' setting. Indeed, for this setting, both models show a substantial increase in performance – which implies that canonicalization represents a large part of the difficulty for this task. Interestingly, the LLM shows a much larger improvement, ultimately outperforming the traditional pipeline by about 2.5 points F_1. Our qualitative error analysis in Table 3 (center part) indicates that our LLM predictions have a hard time hitting the right level of verbosity: they are either too verbose, spuriously including government positions (e.g. [*Interior Minister*] *Thomas de Maizière*), or not verbose enough, omitting first names (e.g. [*Angela*] *Merkel*).

We take this as evidence that our LLM-based model is adept at selecting the correct actor, but struggles to select the canonical form. This is somewhat to be expected, as our LLM-based model has neither a training signal nor a strong inductive bias to prefer any particular canonical form. However, as mentioned in Section 2.3, preliminary experiments with a few-shot setting where we included canonical forms in prompts showed no improvements over our proposed model. We believe that this indicates that the task of predicting 'canonical names' remains a non-straightforward task for LLMs even in the presence of training data.

Error Type	**Model output**	**Ground Truth**
Format	Bayern The claim is	Bayern *(Bavaria)*
	EU-Kommission *(EU commission)*	EU-Kommision [*sic*]
Canonicalization	Bundesinnenminister *(federal minister of the interior)* Thomas de Maizière	Thomas de Maizière
	Kommissions-präsident (*commission president*) Jean-Claude Juncker	Jean-Claude Juncker
	Zimmermann	Klaus F. Zimmermann
	Merkel	Angela Merkel
Wrong Actor	EU-Kommission *(EU commission)*	Jean-Claude Juncker
	Germany	Thomas Bauer

Table 3: Some illustrative examples of the errors exhibited by the LLM-based actor identification model: German outputs with English translations

Finally, responses which bungled the reference completely (Table 3, bottom part) sometimes tended to be plausible, e.g. metonymyic, mistakes, such as predicting the EU commission instead of Jean-Claude Juncker, its president.

Hybrid model. The observations on the errors motivate a follow-up experiment with a hybrid approach combining both our traditional and LLM-based models. This hybrid is structurally similar to our traditional model, but it is provided the LLM's prediction in addition to its other inputs. In this way, the LLM can decide which actor made the claim, while the traditional pipeline can be responsible for predicting that actor in a canonical form. Table 4 shows that this approach has similar properties to the individual models (no effect of formatting, but a large effect of canonicalization) but that it represents, crucially, a substantial improvement

Evaluation	Pr	Re	F_1
exact match	54.33	64.49	58.97
up to formatting	54.33	64.49	58.97
up to canonic.	64.96	76.39	70.21

Table 4: Results for the hybrid model in the different evaluation settings

in terms of quality: In the strictest setting (exact match), it achieves an F_1 score of 59 points (previous best: 53 F_1), and in the laxest setting it obtains 70 points F_1 (previous best: 63 F_1).

5 Conclusion

In this work, we investigate alternative approaches to tackling the discourse-level actor identification task, comparing LLM prompting with a conventional NLP pipeline. We find that our LLM better recognize the appropriate actor entities compared to the traditional pipeline, but has a harder time controlling the exact output. This problem cannot be solved easily with tuning, as the failure of our few-shot setup shows, which is also in line with recent studies on the controllability of LLM output (Reif et al., 2022; Sun et al., 2023). Our solution is a hybrid model which integrates the LLM-generated output as a cue in the pipeline approach, resulting in a clear improvement over the individual models.

The current study is limited in several respects: It only considers one LLM, one corpus, and one evaluation. In the future, we also plan to carry out an extrinsic evaluation of our actor identifier on generating full discourse networks. In terms of future directions, we believe that actor identification is a task which could plausibly profit from retrieval-augmented generation (RAG) proposed by Lewis et al. (2020) which would give the LLM access to information beyond the current discourse.

Acknowledgements

We acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG) for the project MARDY 2 (375875969) within the priority program RATIO.

References

Nico Blokker, Andre Blessing, Erenay Dayanik, Jonas Kuhn, Sebastian Padó, and Gabriella Lapesa. 2023. Between welcome culture and border fence: The European refugee crisis in German newspaper reports. *Language Resources and Evaluation* 57:121–153. https://doi.org/10.1007/s10579-023-09641-8.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*. volume 33, pages 1877–1901. https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Rochelle Choenni, Dan Garrette, and Ekaterina Shutova. 2023. How do languages influence each other? studying cross-lingual data sharing during llm fine-tuning.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, pages 8440–8451. https://doi.org/10.18653/v1/2020.acl-main.747.

Ruud Koopmans and Paul Statham. 1999. Political Claims Analysis: Integrating Protest Event And Political Discourse Approaches. *Mobilization* 4(2):203–221.

Philip Leifeld. 2016. Discourse Network Analysis: Policy debates as dynamic networks. In *The Oxford Handbook of Political Networks*, Oxford University Press.

Philip Leifeld and Sebastian Haunss. 2012. Political discourse networks and the conflict over software patents in Europe. *European Journal of Political Research* 51(3):382–409.

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 Conference on Neural Information Processing Systems*. Vancouver, Canada.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.* 55(9). https://doi.org/10.1145/3560815.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, pages 8086–8098. https://doi.org/10.18653/v1/2022.acl-long.556.

Katerina Margatina, Timo Schick, Nikolaos Aletras, and Jane Dwivedi-Yu. 2023. Active learning principles for in-context learning with large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, Singapore, pages 5011–5034. https://doi.org/10.18653/v1/2023.findings-emnlp.334.

Sebastian Padó, Andre Blessing, Nico Blokker, Erenay Dayanik, Sebastian Haunss, and Jonas Kuhn. 2019. Who sides with whom? Towards computational construction of discourse networks for political debates. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, pages 2841–2847. https://doi.org/10.18653/v1/P19-1273.

Sean Papay, Roman Klinger, and Sebastian Pado. 2022. Constraining linear-chain CRFs to regular languages. In *Proceedings of the International Conference on Learning Representations*. https://openreview.net/forum?id=jbrgwbv8nD.

Haoruo Peng, Daniel Khashabi, and Dan Roth. 2015. Solving hard coreference problems. In Rada Mihalcea, Joyce Chai, and Anoop Sarkar, editors, *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 809–819. https://doi.org/10.3115/v1/N15-1082.

Emily Reif, Daphne Ippolito, Ann Yuan, Andy Coenen, Chris Callison-Burch, and Jason Wei. 2022. A recipe for arbitrary text style transfer with large language models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Dublin, Ireland, pages 837–848. https://doi.org/10.18653/v1/2022.acl-short.94.

Özge Sevgili, Artem Shelmanov, Mikhail Arkhipov, Alexander Panchenko, and Chris Biemann. 2022. Neural entity linking: A survey of models based on deep learning. *Semantic Web* 13(3).

Shivashankar Subramanian, Trevor Cohn, and Timothy Baldwin. 2018. Hierarchical structured model for fine-to-coarse manifesto text ana lysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, pages 1964–1974. https://doi.org/10.18653/v1/N18-1178.

Jiao Sun, Yufei Tian, Wangchunshu Zhou, Nan Xu, Qian Hu, Rahul Gupta, John Wieting, Nanyun Peng, and Xuezhe Ma. 2023. Evaluating large language models on controlled generation tasks. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, pages 3155–3168. https://doi.org/10.18653/v1/2023.emnlp-main.190.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* .

Albert Webson and Ellie Pavlick. 2022. Do prompt-based models really understand the meaning of their prompts? In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Seattle, United States, pages 2300–2344. https://doi.org/10.18653/v1/2022.naacl-main.167.

Carolina Zheng, Claudia Shi, Keyon Vafa, Amir Feder, and David Blei. 2023. An invariant learning characterization of controlled text generation. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Toronto, Canada, pages 3186–3206. https://doi.org/10.18653/v1/2023.acl-long.179.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. https://doi.org/10.48550/arXiv.2311.07911.

A Prompt Templates

#	Instruction templates
1	*"Extract only the entity that made the claim in the article. The claim is surrounded with <claim>and <\claim>tags. Output only the entity without any additional explanation. Article: [ARTICLE]"*
2	*"Extract and standardize only the entity that made the marked claim in the article. The claim is surrounded with <claim>and <\claim>tags. Output only the standardized entity without any additional explanation. Article: [ARTICLE]"*
3	*"Retrieve the party or parties responsible for the statement in the given article, contained within <claim>and <\claim>tags. Output only the entity without further elaboration. Article:[ARTICLE]"*
4	*"Identify and output the entity or entities that made the claim within the specified article, enclosed by <claim>and <\claim>tags. Do not include any supplementary information. Article: [ARTICLE]"*

Table 5: Prompt template instruction paraphrases used for robustness check for zero- and few-shot setting.