Learning Procedural Dependencies from Self-Supervised Instruction Unshuffling

Anonymous ACL submission

Abstract

We develop a self-supervised method for improving the ability of language models to reason about the dependency structure of procedural texts. Previous work has explored using fine-tuned models to classify dependencies between procedure steps and construct flow-graphs using these dependencies. We improve upon these methods by introducing a self-supervised step-unshuffling training objective. By learning to map shuffled sequences of procedure steps to their correct order, our method improves the procedural reasoning abilities of language models. Through experiments we demonstrate that state-of-the-art models including GPT-4 perform poorly at the task of identifying step dependencies, and we also generate significant improvements using our step-unshuffling training objective, surpassing GPT-4 performance.

1 Introduction

Understanding procedural texts is an important goal of natural language processing research. Natural language offers a versatile and accessible means of specifying tasks to people and agents and instructional texts have been leveraged in many domains including robotics (Tellex et al., 2020), video game agents (Branavan et al., 2012), and computer vision (Ramanathan et al., 2013). Unlike commands specifying goals, procedural texts capture additional information about the manner in which a task should be performed. They are sequences of actions and subgoals, and also contain information about the necessary objects for completing a task.

For many applications, a structured representation of the procedural text is necessary. Momouchi (1980) introduces a flow-graph representation for procedural texts which consist of recipe steps and execution dependencies between the steps. These graphs specify the actions and objects of the procedure as nodes and causal dependencies between the nodes as edges. Constructing these graphs involves parsing individual steps by identifying actions and objects, and then determining which steps are dependent on which other steps and which steps can be done in any order. This requires an understanding of the preconditions and postconditions of action-steps. Recent work has utilized pretrained language models (LMs) to construct flow-graphs from recipes (Yamakata et al., 2020). They construct a dataset of recipes annotated with named entities and dependencies and then finetune a LM to classify dependencies between the entities. These approaches rely on the LM to have representations which capture the information that is necessary to reason about dependencies and to generalize this knowledge effectively to new procedural texts.

However recent work has shown that even large language models (LLMs) perform poorly at tasks that require basic procedural reasoning abilities. (Valmeekam et al., 2023) shows that state-of-the-art LLMs significantly under-perform humans at simple formal planning tasks, including generating valid plans, reasoning about plan execution, and even verifying whether provided plans are correct. This appears to indicate that current large models do not contain or cannot leverage representations for reasoning about dependencies in procedural texts.

To improve the construction of flow-graphs, we therefore seek to augment the procedural reasoning abilities of pretrained models. LLMs have demonstrated high performance on semantic parsing tasks involving single sentences (Shin et al., 2021). But generating flow-graphs involves reasoning about longer context relationships and we demonstrate that even the largest models still lack the ability to effectively identify procedure dependencies without access to significant domain-specific supervised training data. To help overcome this, we propose to use a self-supervised learning objective: *procedure step unshuffling*. We construct a self-supervised training task where the model learns to map shuf-

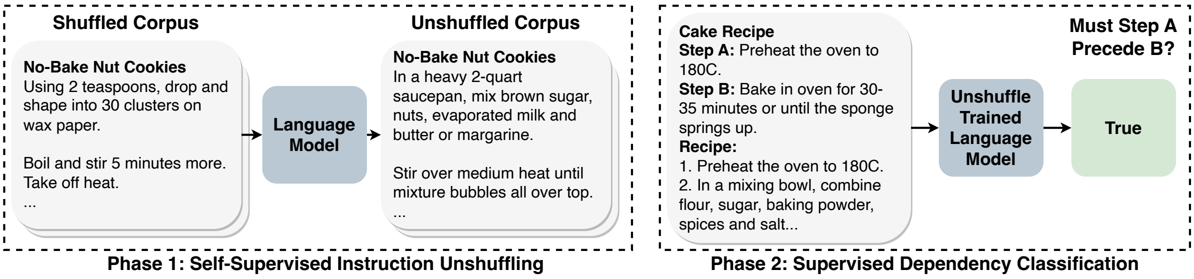

Figure 1: Our method first finetunes a pretrained LM to unshuffle recipe steps. We then finetune the model on a challenging recipe dependency classification task. This task requires determining which recipe steps must precede others in execution order and requires reasoning about the actions and objects.

fled procedure steps to their original order. This method forces the model to learn representations that are useful for reasoning about the order and dependency relationships between instruction steps. We observe that this method improves the performance of the model on the downstream task of recognizing step dependencies in cooking recipes.

We make the following contributions:

- Apply step unshuffling to improve reasoning about the dependency structure of cooking recipes, significantly improving the performance of finetuned models. To the best of our knowledge, this is the first time this objective has been applied to improving the understanding of natural language instructions.

- Show that state-of-the-art language models struggle to reason about instruction step-dependencies without supervised training.

2 Method

We use the English Recipe Flow Graph Corpus[1] (Yamakata et al., 2020) which contains 300 English language cooking recipes annotated with named entities and substep procedure dependencies. We are primarily interested in assessing the ability of LLMs to reason about dependencies and not the parsing of named entities within individual steps. Therefore, we modify the original dataset to construct a new *sentence-level* dependency corpus based on dependencies between sentences in the recipe. For each recipe, a directed acyclic graph (DAG) is constructed where nodes are recipe sentences and edges indicate dependencies between those steps. If two steps are not linked by an edge, they may be performed in any order without changing the recipe result. We divide this corpus into a 70% train, 10% validation, 20% test set split.

[1] https://sites.google.com/view/yy-lab/resource/english-recipe-flowgraph

2.1 Instruction Unshuffling

We improve the pretrained representations of the language models to enable better reasoning about step dependencies. For pretrained language representations we utilize the Flan-T5 models (Chung et al., 2022) which perform at or near state-of-the-art across a variety of NLP tasks including classification and natural language reasoning tasks. Starting with a Flan-T5 model, we finetune this model on the additional training task of unshuffling recipe steps from the RecipeNLG corpus (Bień et al., 2020; Marin et al., 2019; Salvador et al., 2017). We use a randomly select a subset of one million recipes out of the 2,231,150 available. Figure 1 shows an example of the training stages and input format used in both the recipe unshuffling and dependency classification tasks. The hyperparameters for all finetuning tasks are found in Table 3. The model maps a randomly shuffled order of the recipe steps to its ground-truth order in the original recipe using the standard autoregressive sequence modeling loss.

2.2 Finetuning Dependency Classifiers

We formulate the step-dependency recognition problem as a Boolean classification problem. For each pair of ordered steps in the recipe, we classify whether or not there should be a directional dependency between them. We supply the steps, complete recipe text, and title to the classifier to provide the necessary context.

Our approach differs from past work including (Yamakata et al., 2020) in that our method only detects dependencies between pairs of ordered steps and not all possible pairs of steps. From examining the English Recipe Flow Graph Corpus and

Model	Parameter Count	AUC-ROC	AUC-PR Pos	AUC-PR Neg	Accuracy	Error Rate ↓
Dependency Finetune						
Flan-T5 Small	80M	87.6	89.1	84.7	77.5	22.5
Flan-T5 Base	250M	93.3	93.2	93.1	85.4	14.6
Unshuffle + Dependency Finetune						
Flan-T5 Small	80M	91.6	91.5	91.1	83.0	17.0
Flan-T5 Base	250M	**94.2**	**94.3**	**93.9**	**85.6**	**14.4**

Table 1: Performance of finetuned models on a balanced evaluation set of 1420 step dependencies from 60 recipes: a 20% random split of all recipes. The unshuffle trained models outperform the original Flan-T5 models.

RecipeNLG corpus, it is rare that a later step must be done before an earlier step. Later steps can sometimes be done before earlier steps, but this is usually not causally required. This simplifying assumption improves dependency step recognition while not sacrificing applicability to our dataset. However, this simplification may not hold for certain domains.

2.3 Recipe Flow-Graph Construction

Recipe flow-graph construction uses a trained dependency prediction model to predict dependencies for all ordered pairs of steps in a recipe. Similar to (Yamakata et al., 2020), the dependency flow-graph is then constructed greedily starting with the last step in the recipe until all nodes with predicted dependencies are incorporated into the graph. An example of a recipe graph is provided in Appendix A, Figure 2.

3 Results

We perform finetuning experiments using two Flan-T5 model sizes as shown in Table 1 and evaluate in-context learning using GPT-4 (OpenAI, 2023), GPT-3.5 Turbo (Brown et al., 2020), and Mistral 7B Instruct (Jiang et al., 2023). Table 1 reports accuracy, error rate, and area-under-the-curve (AUC) for both receiver operating characteristic (ROC) (Fawcett, 2006) and precision-recall (PR). The AUC-PR is reported for both the positive and negative classes, where the positive class indicates that the first step needs to come before the second i.e. there is a dependency between the steps. Additional training details are available in Appendix A.

We observe that the unshuffling objective significantly improves the performance of the finetuned classification models. For the Flan-T5 Small model, the accuracy increases by 5.5%, which corresponds to a reduction in the error rate of 24.4%. The accuracy improvements to the Flan-T5 Base model are proportionally smaller, but this may be due to a ceiling effect. The improvements in the AUC metrics are comparatively larger.

Model	Accuracy
Zero-Shot	
Mistral 7B Instruct	57.4
GPT-3.5 Turbo	52.0
GPT-4	70.2
Few-Shot	
Mistral 7B Instruct (N=5)	64.2
GPT-3.5 Turbo (N=5)	57.8
GPT-4 (N=5)	73.7

Table 2: Accuracy of LLMs with in-context learning. N indicates the number of in-context examples used. GPT-3.5 does no better than random and GPT-4 underperforms the smaller finetuned models.

For the Flan-T5 Small model, the accuracy improvements are primarily explained by reductions in the number of false positives. As shown in the *AUC-PR Neg* results, the model significantly increases its precision and recall with respect to the negative class. This is the more challenging class for the original Flan-T5 Small model, and as expected representations which allow for better reasoning about step dependencies improve the model's ability to classify which steps are not linked by a dependency.

For the LLMs evaluated, we utilize in-context learning with examples selected from the balanced training set using the BM25 (Robertson et al., 2009) algorithm. Because GPT-4 and GPT-3.5 Turbo do not return token probabilities, AUC-ROC and AUC-PR cannot be calculated. During training we select the model with the highest validation accuracy and report results on the test-set using this model. As shown in Table 2, GPT-4 exhibits poor performance on this task, despite its state-of-the-art performance on other NLP tasks. Their performance also does not significantly improve with the introduction of relevant in-context examples.

4 Discussion

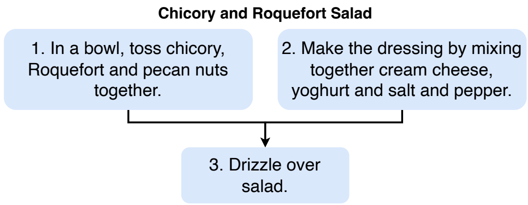

Figure 2: An example dependency graph. The first two steps can be done in any order, but both must be done before the third step.

The autoregressive (Radford et al., 2018) and span-masking (Raffel et al., 2020) pretraining objectives are not well suited to learning language representations that are useful for reasoning about procedural actions and their dependencies (Lin et al., 2021; Bubeck et al., 2023). At a basic level, these objectives encourage the model to learn representations necessary for modeling the conditional generation of text. In the case of the autoregressive objective the model learns how later text depends on earlier text; and for the span-masking objective, the model learns how text depends on adjacent text. For example, given a recipe title, the model learns what text sequence represents a valid recipe for making that food dish. However, these objectives do not explicitly encourage representations which capture relationships between constituent actions of a text and their relationship to the whole text. For example, given a *set* of steps, how can a valid recipe be constructed from them? Even a bidirectional language modeling objective does not fully capture this kind of representation.

The procedure step unshuffling objective aims to address this limitation while building on the powerful existing pretraining objectives which have proven successful at diverse downstream tasks. However, it fills in a missing component of the autoregressive and span masking objectives and provides a means by which the model can learn to represent dependency relationships between steps in the procedure. Our results show that the finetuned classification models and state-of-the-art LMs like GPT-4 fail to capture these dependencies, echoing previous work that has found reasoning deficits in out of distribution tasks (Wu et al., 2023).

5 Related Work

Sentence Unshuffling: Previous work has utilized unshuffling to improve the representations of language models for various applications. Lee et al. (2020) train sentence embeddings using a sentence unshuffling objective, but requires modifications to the underlying model and a specialized decoder architecture. Our approach also differs as it improves the underlying LM representations instead of learning sentence embeddings. As noted by Lee et al. (2020), various language models have proposed using sentence order prediction and unshuffling to improve their language representations, but this does not result in significant improvements on downstream tasks (Lewis et al., 2020; Devlin et al., 2019; Lan et al., 2020). Our approach differs in that we do not consider shuffling of sentences, but rather procedure steps parsed from recipes. Unlike autoregressive sentence order reconstruction, our method results in clear improvements on our downstream task of classifying procedure dependencies. Other work has improved language models to better handle sequential events by finetuning on perturbed sequence orders (Koupaee et al., 2021) but does not explore unshuffling.

Procedural Text Understanding: Papadopoulos et al. (2022); Kiddon et al. (2015) explore predicting dependencies in cooking recipes and related tasks like temporal step ordering of WikiHow instructions (Zhang et al., 2020). We utilize the recipe dependency dataset of Yamakata et al. (2020) but modify the dataset by extracting sentence-level dependencies from their entity level dependencies and therefore do not compare to their results. Wu et al. (2022); Pan et al. (2020) investigate identifying dependencies in multimodal instructions with images and text.

6 Conclusion

We introduce a self-supervised learning objective based on unshuffling procedure steps. This improves the language models' ability to reason abilities about dependency relationships between steps in a procedure. Our method results in significant improvements in the ability of finetuned state-of-the-art models to classify these relationships. Additionally we show that larger state-of-the-art models do not perform significantly better than these smaller models despite using many orders of magnitude more computation at training and inference time. This points to underlying deficits in procedural reasoning abilities that our objective aims to improve.

7 Limitations

The datasets investigated are all English-language datasets and this limits our results and improvements. In future work we plan on investigating whether these techniques can be applied to other languages, particularly low-resource languages where supervised training data is limited. Performance on these languages could benefit from better pretrained representations. While our work only considers the cooking recipe domain for procedural texts, this method can in principle be applied to many other domains. Medical practice guidelines, repair manuals, and software tutorials among others are domains worth investigating. Given that most previous work has found negligible benefit to utilizing sentence unshuffling as a pretraining objective, its worth investigating whether procedure step unshuffling could be incorporated into language model pretraining as a general objective to improve downstream performance on natural language reasoning tasks beyond step dependency classification. In our work we focus on the more narrow case of procedural text understanding, and only train on procedural texts. Given its success in predicting procedure step dependencies, step unshuffling could potentially be applied to other sequential reasoning tasks like planning and we hope to investigate these other domains in future work.

8 Ethical Considerations

As noted, our method seeks to improve machine understanding of procedural texts, but is only demonstrated on a corpus English recipe dependencies and pretrained on a corpus of English recipes. We seek to remedy this in future work. While we believe these datasets form an important domain and test for validating our method, only training for English cooking texts disproportionately benefits those who use English and are able to cook or otherwise take part in cooking-related activities. As this method improves machine understanding of procedural texts and could in principle be used to augment the capabilities of autonomous agents, particularly those which need to follow instructions in the real world, which could be unsafe or promote bias and other social harms.

References

Michał Bień, Michał Gilski, Martyna Maciejewska, Wojciech Taisner, Dawid Wisniewski, and Agnieszka Lawrynowicz. 2020. RecipeNLG: A cooking recipes dataset for semi-structured text generation. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 22–28, Dublin, Ireland. Association for Computational Linguistics.

SRK Branavan, David Silver, and Regina Barzilay. 2012. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. 2015. Mise en place: Unsupervised interpretation of instructional recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 982–992, Lisbon, Portugal. Association for Computational Linguistics.

Mahnaz Koupaee, Greg Durrett, Nathanael Chambers, and Niranjan Balasubramanian. 2021. Don't let discourse confine your model: Sequence perturbations for improved event language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 599–604.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations.

Haejun Lee, Drew A Hudson, Kangwook Lee, and Christopher D Manning. 2020. SLM: Learning a discourse language representation with sentence unshuffling. *arXiv preprint arXiv:2010.16249*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Chu-Cheng Lin, Aaron Jaech, Xin Li, Matthew R. Gormley, and Jason Eisner. 2021. Limitations of autoregressive models and their alternatives. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5147–5173, Online. Association for Computational Linguistics.

Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. 2019. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Trans. Pattern Anal. Mach. Intell.*

Yoshio Momouchi. 1980. Control structures for actions in procedural texts and PT-chart. In *COLING 1980 Volume 1: The 8th International Conference on Computational Linguistics*.

OpenAI. 2023. GPT-4 technical report.

Liang-Ming Pan, Jingjing Chen, Jianlong Wu, Shaoteng Liu, Chong-Wah Ngo, Min-Yen Kan, Yugang Jiang, and Tat-Seng Chua. 2020. Multi-modal cooking workflow construction for food recipes. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20. ACM.

Dim P Papadopoulos, Enrique Mora, Nadiia Chepurko, Kuan Wei Huang, Ferda Ofli, and Antonio Torralba. 2022. Learning program representations for food images and cooking recipes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16559–16569.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Vignesh Ramanathan, Percy Liang, and Li Fei-Fei. 2013. Video event understanding using natural language descriptions. In *Proceedings of the IEEE international conference on computer vision*, pages 905–912.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. 2017. Learning cross-modal embeddings for cooking recipes and food images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.

Richard Shin, Christopher H. Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Ben Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *2021 Empirical Methods in Natural Language Processing*.

Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. 2020. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:25–55.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Te-Lin Wu, Alex Spangher, Pegah Alipoormolabashi, Marjorie Freedman, Ralph Weischedel, and Nanyun Peng. 2022. Understanding multimodal procedural knowledge by sequencing multimodal instructional manuals.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2023. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks.

Yoko Yamakata, Shinsuke Mori, and John A Carroll. 2020. English recipe flow graph corpus. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5187–5194.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020. Reasoning about goals, steps, and temporal ordering with WikiHow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4630–4639, Online. Association for Computational Linguistics.

A Appendix

A.1 Implementation

For training and evaluation we utilize the HuggingFace Transformers library (Wolf et al., 2020) (Apache 2.0) and scikit-learn (Pedregosa et al., 2011) (BSD 3-Clause). All experiments are performed on a machine with a NVIDIA A100 40G and take approximately 40 hours to run in total. The Flan-T5 models (Chung et al., 2022) and Mistral 7B Instruct (Jiang et al., 2023) are available under an Apache 2.0 license. The OpenAI platform terms of service are available at https://openai.com/policies/terms-of-use. The RecipeNLG dataset's (Bień et al., 2020; Marin et al., 2019; Salvador et al., 2017) license is not provided to our knowledge, but is a derivative of the Recipe1M+ dataset which is available under an MIT license. For all datasets used, we checked a random samples of approximately 200 data-points for malicious content and personal information. For in-context example selection for the LLMs we utilize the rank_BM25 library [2] available under the Apache 2.0 license.

Hyperparameters	
Optimizer	Adafactor
Learning rate	5e-4
Batch Size	16

Table 3: Hyperparameters used for training were found by grid search. The Adafactor optimizer was introduced by (Shazeer and Stern, 2018) and was selected for its use in Chung et al. (2022).

[2] https://github.com/dorianbrown/rank_bm25