Inductive Bias Extraction and Matching for LLM Prompts

Anonymous ACL submission

Abstract

All LLMs are sensitive to small changes in prompt wording, and a portion of this can be ascribed to the inductive bias that is present in the LLM. By using an LLM's output as a portion of its prompt, we can easily create satisfactory wording for prompts. This has the effect of creating a prompt that matches the inductive bias in model. Empirically, we show that using this Inductive Bias Extraction and Matching strategy improves LLM Likert ratings used for classification by up to 19% and LLM Likert ratings used for ranking by up to 27%.

1 Introduction

If you wanted to build a patio, you may ask an LLM about how to get started. Two of the steps that it could suggest are "Purchase patio blocks," and "Place patio blocks in the desired area." While placing patio blocks is a logical course of action after purchasing patio blocks, it does not necessarily follow that you must place them immediately, or you may want to build something else that also requires patio blocks. This kind of out of the ordinary application may confuse an LLM that has only seen these two segments in this order, and would be liable to label the placement of the patio blocks as a result of purchasing.

To determine the relationship between two text segments (such as in the patio block example), it will be helpful to know whether one is a direct result of the other, or if the one is a requirement of the other. These requirement and result relationships are subtly different, and it would be helpful to have a fine grained scale for both to help quantify what the relationship is. The issue is that producing high quality scales of this nature would require a knowledge of the inductive biases of the LLM for that particular task. Normally this would be determined through a lengthy prompt engineering process, but creating a method for automatically extracting these inductive biases would require less labor and would not be subject to human error.

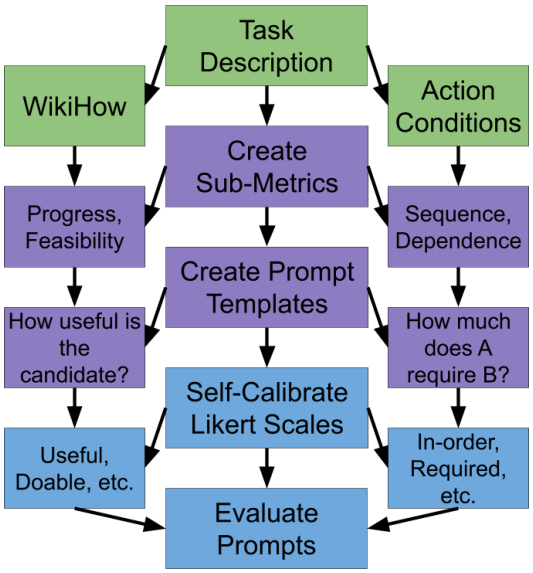

Figure 1: An overview of the IBEaM pipeline. Inputs are in green, human steps are in purple, and LLM steps are in blue.

The method that we use to improve LLMs' performance on non-comparative classification tasks is a process that we call Inductive Bias Extraction and Matching (IBEaM), shown in Figure 1. By incorporating prior knowledge we have about the task, namely what sub-tasks it can be broken down into and how we should combine them once they are solved, we can more fully take advantage of the different strengths of the LLM. When using the LLM to solve these sub-tasks, ideally we would know what the optimal prompt would be to solve the sub-task. As shown by the large body of work done in prompt engineering (Chen et al., 2025), LLMs are sensitive to small changes in prompt wording, and a portion of this can be ascribed to the inductive bias that is present in the LLM that gives it a "preference" for certain wordings. Usually these issues with prompt wording are solved through a repeti-

tive prompt engineering process, but by prompting the LLM for its preference for prompt wording, we can eliminate the need for using prompt engineering to find the specific wording that works best with an LLM's inductive biases. IBEaM is not inherently limited to non-comparative classification tasks, but due to the difficulty of quantifying performance changes in other tasks, we limit the scope of this paper to this type of task.

For each of the tasks that we examine in this paper, we prompt an LLM for one or more 10 point Likert scales that can be used to generate one or more scores for each instance for each task. These scores are then combined to generate an overall score (or multiple overall scores if the task is multiple-choice) that is then used to perform classification. For each task, we also evaluate a simple baseline where the LLM is prompted for a rating from 1 to 10 (or, again, multiple ratings if the task is multiple-choice) for each instance. Like the scores generated with our IBEaM method, these ratings are then used to perform the downstream classification task.

The contributions presented in this paper are as follows:

- We define methods for inductive bias extraction and including that extracted inductive bias in prompts to improve LLM performance.
- We demonstrate that you can describe the criteria to evaluate and let the LLM identify its preferred wording.
- We show that an LLM's ability to generate numeric scores is limited, and we demonstrate that IBEaM can be used to generate improved numeric scores from LLMs.

2 Related Work

With the advent of generative large language models (LLMs) (Achiam et al., 2023; Touvron et al., 2023; Brown et al., 2020), a large body of research has been conducted to determine what tasks they can be applied to and what their efficacy is for those tasks (Zhao et al., 2025). While LLMs exhibit emergent behaviors that allow them to perform tasks that machine learning methods were previously ineffective for, they can also be applied to tasks that existed prior to LLMs being introduced. The motivation for applying LLMs to these older tasks may be the desire to improve efficiency in execution speed, improve accessibility to non-technical users, or improve the state-of-the-art in accuracy. This broad spectrum of LLM applications is what has motivated us to evaluate IBEaM by generating numeric scores and using them as a proxy for performing classification and ranking tasks .

One method of improving LLM performance on tasks is to perform chain of thought reasoning, where the LLM is walked through the process of solving one or more examples in the prompt prior to being asked to evaluate an instance (Wei et al., 2022). Another method of improving LLM performance is to perform extensive prompt engineering (Chen et al., 2025), which is typically a repetitive process that optimizes an LLM's downstream performance by making small adjustments to its prompt. All prompt engineering techniques fundamentally are attempting to make the prompt work better with the parameters learned by the LLM, or in other words, are attempting to find the prompt that is most compatible with the LLM's inductive biases.

All of the datasets examined in this paper have some concept of a world state implied by the text. LLMs are able to reason about the hypothetical world state implied in a prompt, and also the real world state implied by their training data (Zhu et al., 2023). The encoding of this real world state is a form of inductive bias in LLMs.

3 Method

Applying IBEaM to a new task is a four step process. First, the user must create some number of component metrics to enable the extraction of additional inductive bias. Second, the user must create a prompt template that can integrate the extracted inductive bias into an evaluation prompt. Third, the user must prompt the LLM to extract the inductive bias for each component metric, and finally, the user must devise some method of combining the component scores into a final evaluation. Once all of these steps have been performed, the result is a pipeline that can be used to improve the performance of an LLM on the desired task.

3.1 Creation of Component Metrics

The first step in the process of applying IBEaM to a problem is to break the problem into a reasonable number of fine-grained sub-problems. The goal here is not to break the problem into overly small

problems, but rather to determine the high-level dimensions that define the task. For example, if our task is to determine how well "Putting a potato in the oven" continues the process of "Baking a potato" given a previously completed step of "Preheat the oven," two logical dimensions of this task are progress and feasibility. Ideally, when determining whether or not a step is a good continuation of a process when working towards a goal, we want the candidate step to both make substantial progress and to be substantially feasible. Other tasks will break into different component metrics, and this is a determination that needs to be made by the IBEaM user.

Splitting a task into component sub-tasks is not unique to IBEaM, but IBEaM obtains outsized benefits from this process. This is because the more Likert scales obtained from an LLM across unique dimensions, the more inductive bias can be extracted from the LLM. The benefit that IBEaM gets from this additional inductive bias extraction is in addition to the intuitive benefits from breaking a task into logical sub-problems. These intuitive benefits include getting more detailed output from the LLM, getting an evaluation across multiple dimensions, and, if applicable, getting the ability to unevenly weight the answers to the different sub-problems.

3.2 Inductive Bias Extraction

The next step in the process of applying IBEaM to a problem is to extract the LLM's inductive bias for each component metric. For all of the tasks that we study in this paper, this involves prompting the LLM for a Likert scale for each component metric. For example, for the feasibility metric mentioned previously, the LLM may produce a 10 point scale along the lines of the scale seen in Figure 2.

As shown above, we now have extracted the inductive bias of the model in the form of its preferred wording for each rating in the Likert scale. If we were to attempt to create this scale manually as opposed to prompting the LLM for it, we would have to go through a labor intensive prompt engineering process.

An additional benefit of this inductive bias extraction process is that we obtain the LLM's preferred wording for each 10 point Likert scale without having to go through a labor intensive prompt engineering process.

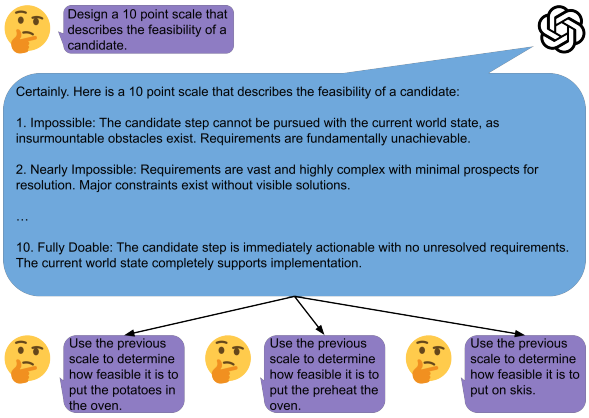

Figure 2: For each of our trials, we use a consistent Likert scale that has been defined by an LLM across all instances. To eliminate the need for the curation of these scales, we include the scale as part of the conversation history when making future calls to the LLM. Keeping the scale the same across all instances improves rating consistency and reduces the cost and computation time when compared to regenerating the scales for each instance.

3.3 Inductive Bias Matching

The third step in the process of applying IBEaM to a problem is to match the LLM's inductive bias when prompting. This involves creating a prompt to evaluate each metric which includes the generated Likert scale. As we demonstrate with our experiments, the prompts for evaluating each metric do not need to be complicated, and can include the generated Likert scale as part of the LLM's conversation history, which allows us to skip any postprocessing of the Likert scales in our experiments. This conversation history is utilized in the prompt by using wording such as, "Rate the following candidate from 1 to 10 on the previous feasibility scale." A simplified version of this process is shown in Figure 2.

An additional benefit of this inductive bias matching process is that we obtain a consistent rating scale that we can apply to all instances in a task, which allows the LLM to apply consistent and more specific criteria when compared to intuitive baseline approaches.

3.4 Combine Component Metrics

As a result of splitting a task into multiple sub-problems, we have the issue of there being multiple component scores that need to be combined in some way to make an overall score. The exact method by which we combine the scores varies depending on the type of task. For ranking tasks

like WikiHow, a simple summation of the scores is usually an effective and intuitive starting point. We tested summation, product, summation of score ranks, and product of score ranks. Of these we found that the product of score ranks was the most effective combination method while not requiring any learned parameters. Assuming n is the number of multiple choice options, this method works by assigning a rank from 0 to $n-1$ to each candidate for each metric. The final combined score is the product of these ranking numbers. The benefit of this method is that it promotes candidates that have more balanced scores compared to the other candidates while still promoting candidates that have high scores overall.

Our other two tasks, SAGA Task 2 and the Action Conditions task, are classification tasks as opposed to ranking tasks. This requires us to create some sort of decision boundary depending on the values of the scores. Empirically, we found that feeding one-hot encoded vectors for all of the submetrics into logistic regression was the most effective way of creating a decision boundary. This, however, necessitates the use of a training set for the logistic regression model which is not required for ranking tasks. As such, for SAGA Task 2 and the Action Conditions task, we take a small sample of the training set and feed that through the IBEaM pipeline. The labels and the LLM's responses are then used to fit the logistic regression models for both of these tasks.

4 Datasets

We use three different datasets to evaluate the efficacy of IBEaM when compared to intuitive baselines. These datasets include SAGA, the Action Conditions dataset, and our own WikiHow-based dataset. SAGA and the Action Conditions dataset allow us to evaluate IBEaM's ability to enable classification while our WikiHow-based dataset allows us to evaluate IBEaM's ability to perform ranking.

4.1 Custom WikiHow-based Dataset

WikiHow was selected as one of our datasets because we wanted one of our tasks to require the LLM to have an understanding of implied world state. Any task requiring this will test the LLM's capacity for reasoning, and we wanted to evaluate IBEaM's effect on the LLM's reasoning capabilities. The sheer volume of procedural text within WikiHow makes it an invaluable resource when evaluating an LLM's ability to perform world state reasoning.

We created a 5-way multiple choice task from WikiHow where each instance contains an in-progress goal, some number of completed steps, and five candidate continuations for the procedure. The five candidates are composed of the target step and four distractors. The four distractors include a duplicate step that has already been performed, two out-of-document steps, and an out-of-order future step. In particular, the out-of-order future steps is part of the same article as the goal and the target, but there is another step that needs to be completed first before the out-of-order future step can be performed. We annotated 66 instances with the target, distractors, previously completed steps, and overall goal. Since a training set is not required for our ranking method, this dataset is exclusively a test set and does not contain training or validation splits.

Of these distractors, the duplicate step is particularly difficult to rank correctly because it is usually feasible to repeat a step (e.g. preheat the oven) when it already has been done, but it does not make any progress towards the goal. However, the out-of-order future step is even more difficult because it makes progress towards the goal (e.g. leave the potatoes in the oven for an hour), but the LLM needs to identify the gap between its requirements (e.g. potatoes in the oven) and the world state (e.g. potatoes on the table) in order to determine that it is not a viable continuation to the current process.

4.2 SAGA Dataset (Task 2)

The ROCStories dataset is composed of instances referred to as stories, which have five events in a sequence that can be summarized by a single sentence (Mostafazadeh et al., 2016). The the PASTA dataset is an extension of ROCStories that substitutes alternative events into the stories, which may or may not match up with the original summary sentence (Ghosh et al., 2023). The Story Alternatives and Goal Applicability (SAGA) dataset is an annotated extension to PASTA, which among other things, contains human evaluations of how applicable these alternative stories in PASTA are to the original summary (Vallurupalli et al., 2024). The paper for SAGA defines multiple tasks that can be tested on SAGA, and in particular we will be performing Task 2. Task 2 is a binary classification task that tests a model's ability to determine if a summary is applicable to a story or not. We chose this dataset because it is nontrivial to perform, but

it also has the simple evaluation criteria of Macro and Micro F1 score.

4.3 Action Conditions Dataset

The Action Conditions dataset is also derived from WikiHow (Wu et al., 2023), but instead of focusing on ranking next steps in a process, it instead defines segments of text within WikiHow articles as pre-conditions and post-conditions. In this context, a pre-condition relationship indicates that one text segment is a requirement for the other, while a post-condition relationship indicates that one text segment is the result of another. Any pair of text segments within a WikiHow article that does not have an indicated pre-condition or post-condition relationship has what they refer to as a NULL relationship. These relationships are not necessarily bidirectional, so if one text segment is a pre-condition of another, the second text segment is not necessarily a pre-condition or post-condition of the first unless labeled as such. For our evaluations, we define an instance as a pair of text segments, a label, and the minimum body of text that contains both text segments. We chose this dataset because it is also nontrivial to perform, has simple evaluation criteria of Macro and Micro F1 scores, and is sufficiently different from our other two tasks.

5 Experimental Setup

For all of our experiments, each trial uses a freshly generated set of Likert scales both for IBEaM, and both IBEaM and the baselines re-prompt the LLM even in cases where the prompt would be identical to a prompt from another trial. The reason for this is that we observed that the LLM tended to have small variances in their responses even when given an identical prompt. Within each trial, LLM responses are cached to reduce inference time and cost. The LLM used for both IBEaM and the baselines is the 2024-08-06 snapshot of GPT-4o (Hurst et al., 2024).

5.1 Custom WikiHow-based Dataset

For IBEaM, we split the task into progress and feasibility sub-tasks. Each of these sub-tasks has a Likert scale generated for it which is then included as part of the conversation history for the LLM. Each Likert scale is regenerated for each trial, but is consistent for each instance within a trial. The most effective score aggregator that we found was to rank the scores for progress and feasibility individually, and then take the product of the two numeric rankings. Whichever candidate has the highest product is selected as the prediction.

For our baseline, we simply prompt the LLM for a rating from 1 to 10 for each candidate given the in-progress goal and the completed steps. Whichever candidate has the highest rating is chosen as the prediction.

5.2 SAGA Dataset (Task 2)

For IBEaM, we split the task into summarization, objective, and accomplishment sub-tasks. Like the WikiHow task, each of these subtasks has a Likert scale regenerated for it for each trial. However, unlike the WikiHow task, SAGA Task 2 is a binary classification task as opposed to a ranking task, so we cannot rank the scores to determine what the best prediction is. Instead, we get a number of training instances for a logistic regression model. This logistic regression model takes in the numeric ratings as one-hot encoded vectors and predicts either true or false for applicability, and we fit separate logistic regression models for IBEaM and the baseline. The complete pipeline is shown in Figure 4.

While the logistic regression component is not a part of IBEaM, it does aid in evaluating the improvement in the ratings produced by IBEaM over the ratings produced by the baseline. As a result of us using the LLM to produce ratings rather than direct classification predictions, we need a way to convert the ratings into a binary classification. Logistic regression does this for us while still being informative because it will perform better when the ratings are closer to being linearly separable. Whichever prompting strategy that produces more linearly separable results is the prompting strategy that is more certain about its predictions. While certainty is not a direct proxy for downstream results, using a separate training set for logistic regression will also translate any decreased quality in the ratings to decreased accuracy.

The test set for SAGA is composed of 512 instances, and each trial uses the same test set, of these 512 instances, 419 have a positive label while only 93 have a negative label. This imbalance in labeling is representative of the training set, so we must apply a modifier to the class weights when fitting the logistic regression model. For the purposes of fitting the logistic regression model, a different 512 instance sample is taken from the training set for each trial.

For our baseline, we again prompt the LLM for

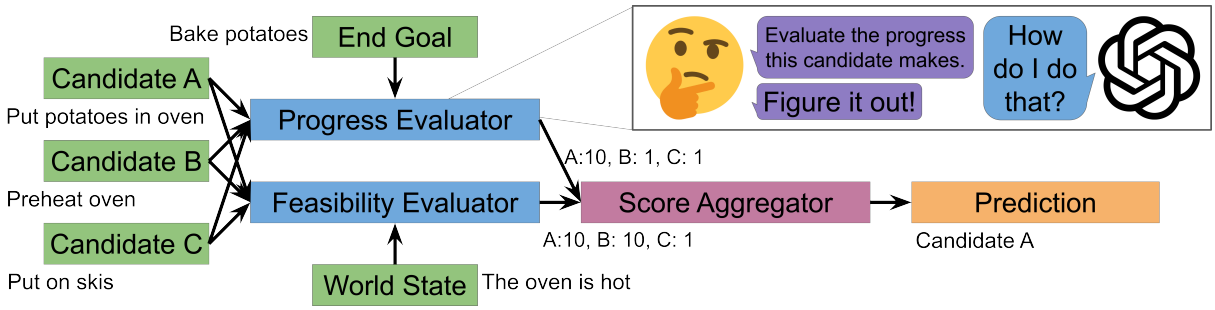

Figure 3: An overview of IBEaM in use for our WikiHow task.

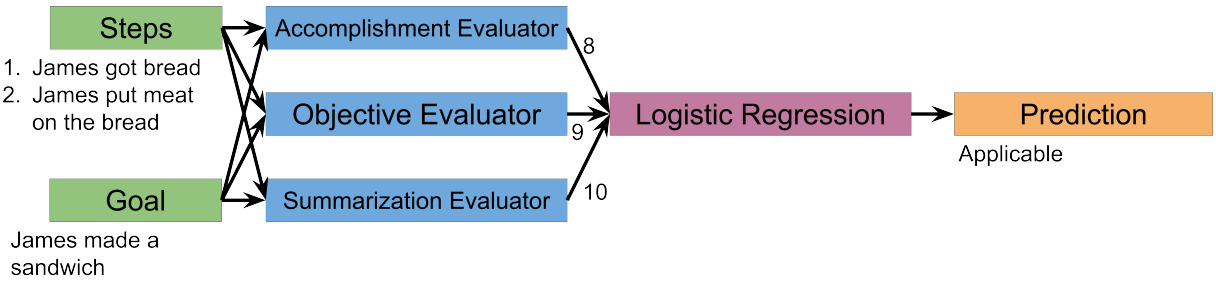

Figure 4: An overview of IBEaM in use for SAGA Task 2.

a rating from 1 to 10 for how applicable a goal is to a sequence of steps. Like the method we used for IBEaM, we again fit a logistic regression model to determine which ratings predict true or false for applicability. For each trial, the training set used to fit the logistic regression model for the baseline is the same one that is used for IBEaM.

5.3 Action Conditions Dataset

For IBEaM, we split the task into sequence, dependence, and independence sub-tasks. Like the WikiHow task and SAGA Task 2, each of these sub-tasks has a Likert scale regenerated for it for each trial. The Action Conditions task is a ternary classification task as opposed to a binary classification task, but we can still apply the same logistic regression method that we used for SAGA Task 2. Again, we get a number of training instances for our logistic regression model, but instead of predicting true or false, it predicts pre-condition, post-condition, or NULL. The complete pipeline is shown in Figure 5.

The test set for the Action Conditions dataset is far too large to evaluate in its entirety, so instead we take a different 512 instance sample from the test set for each trial. Like SAGA Task 2, a different 512 instance sample is taken from the training set for the purposes of fitting the logistic regression models for each trial.

For our baseline, we prompt the LLM for two different scores on a scale from 1 to 10. The first score is a pre-condition score and the second score is a post-condition score. Like SAGA Task 2, these two scores are given to a trained logistic regression model to determine whether the prediction should be pre-condition, post-condition, or NULL. Again for each trial, the training set for the baseline is the same one that is used for IBEaM, and the test set for the baseline is the same one that is used for IBEaM.

6 Discussion

For all of our experiments, we see at least a small improvement in the relevant evaluation metrics on average. The amount of improvement varies widely depending on the task, and that can likely be attributed to the LLM's ability to perform well with the baseline prompt. In particular, the LLM struggles to perform well on the Action Conditions baseline, which leads to the largest increase in performance when applying IBEaM.

6.1 Custom WikiHow-based Dataset

As shown in Figure 6 and Table 1, we show an almost 12 percentage point improvement in accuracy and more than a 0.15 improvement in MRR over the baseline when averaged across all trials. For this dataset, all trials show consistent improvement

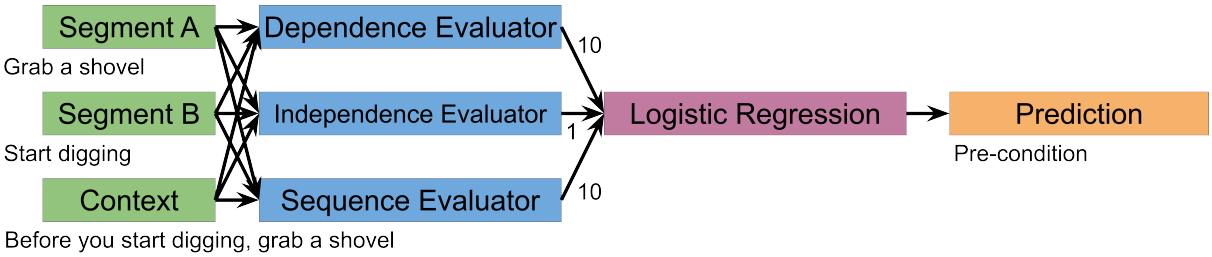

Figure 5: An overview of IBEaM in use for the Action Conditions Task.

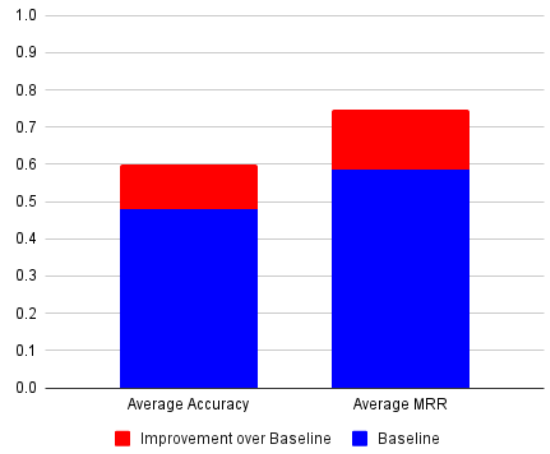

Figure 6: A representation of the overall improvement in performance when applying IBEaM to our WikiHow task. The combined height of the blue and red bars is the performance of IBEaM.

	Accuracy		MRR	
	IBEaM	Baseline	IBEaM	Baseline
Trial 1	**0.573**	0.500	**0.718**	0.598
Trial 2	**0.591**	0.495	**0.751**	0.610
Trial 3	**0.636**	0.492	**0.771**	0.605
Trial 4	**0.583**	0.477	**0.728**	0.561
Trial 5	**0.614**	0.442	**0.764**	0.560
Average	**0.599**	0.481	**0.746**	0.587

Table 1: An overview of the performance of IBEaM and the baseline for each WikiHow trial. Accuracy values give partial credit for ties. If the target has the same score as one distractor, half credit is given, two distractors result in one third credit, and so on.

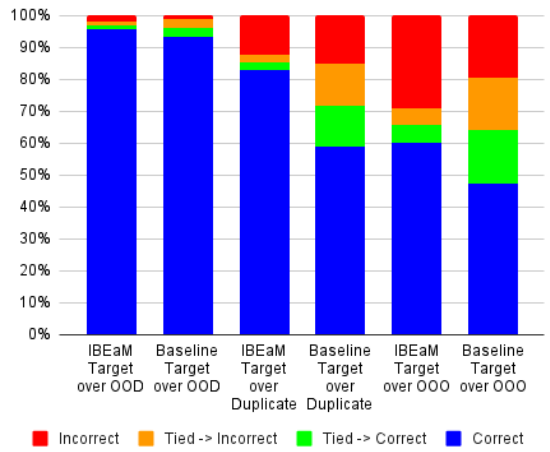

Figure 7: A breakdown of IBEaM and the baseline's performance on the WikiHow dataset. Since all assigned scores for both methods are integers, there are a non-trivial number of ties for both methods. This figure shows the performance of both models when counting half of the ties as correct and the other half of the ties as incorrect, which would simulate the model randomly choosing one of the highest scoring candidates if multiple candidates were assigned the highest score.

over the baseline as well. We also performed an analysis of the performance on each individual distractor to see if each category of distractor was as difficult as we expected. As shown in Figure 7, the performance on each category is about what we expected for both IBEaM and the baseline. Of particular interest is the number of scoring ties produced by each method. The baseline, likely as a result of its predictions being derived from a single 10 point scale, had many ties when evaluating both the duplicate distractor and the out-of-order distractor. IBEaM had substantially fewer ties, and this was likely in part because it was combining multiple metrics, which allows for finer grained scoring predictions than the baseline. Even with its increased certainty, it still outperformed the baseline in all categories both when counting all ties as incorrect predictions and when counting 50% of ties as incorrect predictions.

6.2 SAGA Dataset (Task 2)

As shown in Table 2, out of the five trials we performed, all macro F1 scores showed improvement, and four out of five micro F1 scores showed improvement. For Trial 4 where the Micro F1 score was lower than the baseline, we need to recall that there is a substantial labeling imbalance in the test

	SAGA Task 2				Action Conditions			
	Micro F1		Macro F1		Micro F1		Macro F1	
	IBEaM	Baseline	IBEaM	Baseline	IBEaM	Baseline	IBEaM	Baseline
Trial 1	**0.717**	0.674	**0.660**	0.616	**0.469**	0.465	**0.423**	0.405
Trial 2	**0.697**	0.643	**0.639**	0.595	**0.477**	0.447	**0.431**	0.420
Trial 3	**0.662**	0.643	**0.607**	0.585	**0.533**	0.402	**0.467**	0.362
Trial 4	0.674	**0.689**	**0.634**	0.620	**0.496**	0.455	**0.441**	0.387
Trial 5	**0.736**	0.656	**0.644**	0.603	**0.514**	0.328	**0.479**	0.299
Average	**0.697**	0.661	**0.637**	0.604	**0.498**	0.420	**0.448**	0.375

Table 2: An overview of the performance of IBEaM and the baseline for the classification tasks. IBEaM consistently performs better than the baseline with the exception of one SAGA Task 2 trial.

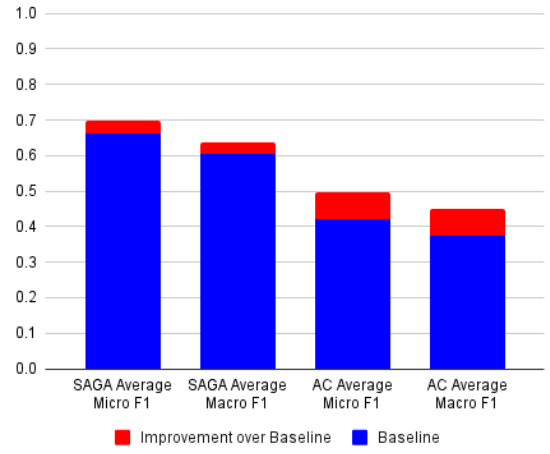

Figure 8: A representation of the overall improvement in performance when applying IBEaM to our classification tasks. The combined height of the blue and red bars is the performance of IBEaM.

set, meaning that if IBEaM or the baseline is more biased towards labeling instances as applicable, the Micro F1 score will increase and the Macro F1 score will decrease. The inverse relationship is true as well, so this is an indication that IBEaM and the baseline performed about the same overall in Trial 4. However, the other four trials show a consistent increase in both Macro and Micro F1 scores when compared to the baseline, so IBEaM performs better overall. This overall increase in performance is also shown in Figure 8.

6.3 Action Conditions Dataset

As shown in Table 2 and 8, IBEaM performs better than the baseline in all trials, but considering that the Action Conditions task is a ternary classification problem, both IBEaM and the baseline struggle to perform well on this task. The range of F1 scores is also much higher for the baseline than it is for SAGA Task 2, and this can be partially attributed to a different test sample being taken from the Action Conditions dataset for each trial, which was not done for SAGA Task 2. Of note is that IBEaM has a lower variance in its F1 scores even with the different test sample for each trial. The baseline's lowest Micro F1 score is 0.328 and its highest Micro F1 score is 0.465, giving it a range of 0.137. Also its lowest Macro F1 score is 0.299 and its highest Macro F1 score is 0.420, giving it a range of 0.121. This is in contrast to IBEaM's Micro and Macro F1 ranges being 0.064 and 0.056, respectively, so in addition to performing better than the baseline, it is also much more consistent.

7 Conclusion

In this paper, we presented IBEaM, which is a method for writing prompts that automatically extract and apply the LLM's inductive biases in a way that improves its ability to assign scores that can be used for downstream ranking and classification. This ability of LLMs to extract their own inductive biases likely has broad applications, but we quantified the possible performance improvements in multiple contexts with concrete evaluation criteria. When compared to our baselines that prompt the LLM for a rating from 1 to 10 with a short description of the task, the increased complexity of IBEaM is a worthwhile investment in contexts where a numeric score is required in addition to an overall prediction of ranking or classification.

Limitations

All of our experimentation was performed using GPT-4o, which is a closed source LLM trained on a proprietary dataset. IBEaM is not dependent specifically on GPT-4o, but it is a technique specifically designed for interacting with LLMs, and as such,

is subject to the same limitations as whatever LLM it is being used with.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. 2025. Unleashing the potential of prompt engineering for large language models. *Patterns*, page 101260.

Sayontan Ghosh, Mahnaz Koupaee, Isabella Chen, Francis Ferraro, Nathanael Chambers, and Niranjan Balasubramanian. 2023. Pasta: A dataset for modeling participant states in narratives. *Preprint*, arXiv:2208.00329.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

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.

Sai Vallurupalli, Katrin Erk, and Francis Ferraro. 2024. Saga: A participant-specific examination of story alternatives and goal applicability for a deeper understanding of complex events. *Preprint*, arXiv:2408.05793.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Te-Lin Wu, Caiqi Zhang, Qingyuan Hu, Alexander Spangher, and Nanyun Peng. 2023. Learning action conditions from instructional manuals for instruction understanding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3023–3043, Toronto, Canada. Association for Computational Linguistics.

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, and 3 others. 2025. A survey of large language models. *Preprint*, arXiv:2303.18223.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zheng Liu, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*.