Optimizing Language Models for Argumentative Reasoning

Luke Thorburn[1,2,*], Ariel Kruger[1]

[1]Hunt Lab, University of Melbourne, Parkville, Victoria 3010, Australia
[2]King's College London, Strand, London WC2R 2LS

Abstract

Large transformer-based causal language models are capable of strong performance on many natural language processing tasks. Here, we systematically evaluate the performance of the 2.7 billion parameter GPT Neo pre-trained language model on 6 argumentative reasoning tasks under 5 different optimization strategies, including prompt programming, soft prompts, and parameter tuning. We report both intrinsic evaluation metrics (perplexity), and extrinsic measures of the coherence of model outputs, as judged by an expert human rater. With a few exceptions, the rate at which models produced coherent responses ranged from 15-50%. In contrast, human performance (users of the *Kialo* argument mapping platform) ranged from 65-82% coherent, depending on the task. These results suggest that larger, suitably optimized language models may be capable of supporting authors and auditors of natural language argument maps in human-in-the-loop settings. We share our finetuned models and code.

Keywords

language model, argument generation, finetuning, soft prompt

1. Introduction

Since Douglas Engelbart first envisaged software for authoring structured argumentation [1], the goal of an algorithm that can automatically check human reasoning—a "spell checker for logic"—has been discussed. More broadly, knowledge workers of many types (including academics, risk analysts, and intelligence analysts) stand to benefit from tools that help them reliably and efficiently construct coherent arguments. One approach to realizing such tools is to integrate automated reasoning algorithms with argument mapping software, an approach that we have taken recently[1].

There are many specific argumentative tasks that it may be useful to automate, such as the generation of reasons and objections, the identification of unstated premises, and the process of "tightening up" an argument by rewording premises to best represent the most argumentatively appropriate claim. In resource-constrained settings, it may only be possible to automate these

1st International Workshop on Argumentation and Machine Learning @COMMA 2022, September 13, 2022, Cardiff, Wales

*Corresponding author.
luke.thorburn@kcl.ac.uk (L. Thorburn); ariel.kruger@unimelb.edu.au (A. Kruger)
https://lukethorburn.com/ (L. Thorburn)
0000-0003-4120-5056 (L. Thorburn); 0000-0002-0121-2780 (A. Kruger)

CEUR Workshop Proceedings (CEUR-WS.org)

[1]Our prototype "argument processor", which integrates with the commercial OpenAI language model API, can be found at https://luke-thorburn.github.io/argument-processor/.

numerous tasks if there is a common, accessible method that can be applied to all of them. In this paper, we investigate the extent to which an open-source pre-trained language model, the 2.7 billion parameter version of GPT Neo [2], can be optimized to perform such argumentative reasoning.

1.1. Background

Large transformer-based causal language models are capable of strong performance on many natural language processing (NLP) tasks [3, 4]. This flexibility is made possible by the generality of causal language modeling—the task of predicting what text comes next, conditioned on the text that has come before. Any task that can be articulated as a natural language *prompt* followed by a *response* can be posed to a causal language model. For this reason, pre-trained language models can in some cases serve as few-shot or zero-shot learners [5, 6, 7, 8] by including instructions or examples of the task as part of the prompt [8]. Performance can often be improved further by tuning some or all of the model weights [9, 10].

Previous academic work has investigated whether language models can emulate different types of logical reasoning. For example, El Baff et al. [11] use language models to synthesize arguments for or against a given claim by training a language model over a vocabulary of curated argumentative discourse units. Clark et al. [12] find that when facts and rules are presented in natural language, a language model can reason over a knowledge base with high accuracy. Gurcke et al. [13] evaluate whether premises are sufficient to draw a conclusion by comparing the stated conclusion with one generated by a language model. Skitalinskaya et al. [14] tune a language model to evaluate the quality of argumentative claims. Other work has investigated the ability of language models to identify logical fallacies [15]. Increasingly, commercial prototypes are also demonstrating the potential for language models to assist with human reasoning tasks in a human-in-the-loop manner. A prominent example is *Elicit* [16], an "AI research assistant" powered by OpenAI's GPT-3 language model that includes tools to brainstorm counterarguments, increase the specificity of claims, and suggest antecedents and consequences to expand a partial chain of reasoning.

Without providing a comprehensive review, we note that there are other approaches to automating natural language reasoning that do not rely solely on language models. One prominent example is Project Debator [17, 18], an attempt to build an autonomous agent that can compete in formal debate.

1.2. Contribution

In this project, we systematically explored the performance of the GPT Neo pre-trained language model on 6 argumentative reasoning tasks, under 5 different optimization strategies. The tasks, described in Section 2.2, correspond to tasks commonly performed by an analyst in the course of authoring an argument map. We report both intrinsic evaluation metrics (perplexity), and extrinsic measures of the coherence of model outputs, as judged by a human rater. To our knowledge, this is the first systematic evaluation of a large ($>10^9$ parameter) language model on argumentative reasoning tasks, despite the success of such large models elsewhere in NLP [5, 19]. Our results form a baseline for future work, and provide insight into which optimization

strategies are most successful. In addition, we are releasing the finetuned models, along with code for performing optimization and inference, to aid future research[2].

2. Inputs

In this section we describe the pre-trained foundation model we used, the NLP tasks for which we optimized it, and the datasets used for tuning.

2.1. Foundation model

As our foundation model we used the 2.7 billion parameter version of the open-source GPT Neo language model [2]. GPT Neo has a transformer-based decoder architecture designed to replicate that of GPT-3 [5], albeit with fewer and smaller layers than are implemented in the largest version of GPT-3. For details on the architecture of GPT Neo, we direct the reader to the original papers on the GPT series of models [20, 21, 5]. GPT Neo was pre-trained on 'The Pile', an 800GB corpus of diverse text collated for language modeling [22].

2.2. Tasks

We investigated six argumentative NLP tasks, which are described in Table 1.

Tasks 1-5 could be described as a type of "masked argument modeling" or cloze completion. They take as inputs a small argument map—potentially a subset of a much larger map—with one or more claims missing (strictly one in all cases except `suggest-intermediary-claims`, where arbitrarily many may be missing). The task is then to generate a claim or claims that could coherently fill the gap. The five tasks differ in the type and position of the claim that has been masked (whether it is a reason, co-premise, conclusion, etc.).

All tasks investigated are generative, and intended to aid an analyst as they construct an argument map by (a) improving the efficiency with which they can compose relevant claims, and (b) prompting them to consider counter-arguments or implicit assumptions that they may not otherwise have identified. The optimized models are intended to be integrated with argument mapping tools where a hi-tree data structure [23] can be assumed. This avoids the need to rely on an imperfect argument mining pipeline to extract such a structure from argumentation presented in prose.

Task 6, `suggest-abstraction`, corresponds to a common step in the process of refining an ill-formed argument map. Often a premise will be too specific in relation to the target claim to best characterize the nature of the logical relationship between them. In such circumstances, the analyst should revise the claim to describe a more general or abstract inferential principle, which is the task we try to automate. Consider the following example, taken from [24].

```
Nuclear power is generated by the breakdown of heavy elements.
  ⟹  Nuclear power has very low greenhouse gas emissions.
```

[2]The code, along with details of how to download the models, can be found at https://github.com/Hunt-Laboratory/language-model-optimization.

Table 1
The 6 argumentative NLP tasks we investigated. All tasks are posed in the context of an argument map, assumed to be a set of claims with a hi-tree structure [23]. A *reason* is a claim that supports another claim, an *objection* is a claim that attacks another claim, and *co-premises* are claims that jointly, but not separately, support another claim. A *conclusion* or *target claim* is a claim for which there exist reasons, objections or co-premises.

ID	Task	Description
1	`suggest-reasons`	Given a target claim and (optionally) some reasons, suggest (additional) reasons.
2	`suggest-objections`	Given a claim and (optionally) some objections, suggest (additional) objections.
3	`suggest-conclusion`	Given one or more reasons, suggest a claim that can be inferred from the reasons.
4	`suggest-intermediary-claims`	Given a starting claim (a reason) and an end claim, suggest an expanded sequence of claims containing intermediary inferences between the start claim and the end claim.
5	`suggest-copremise`	Given a claim and one or more co-premises, suggest additional co-premises required for the inference to be valid.
6	`suggest-abstraction`	Given a claim and a reason, suggest a more abstract version of the reason that better represents the logical relationship between the reason and the claim.

This argument is valid in a vague sense, but the premise is overly narrow and does not well represent the core reason it supports the conclusion. A better version would be:

> *The physics of nuclear power generation involves no combustion.*
> $\Longrightarrow$ Nuclear power has very low greenhouse gas emissions.

It is this specific type of revision that the `suggest-abstraction` task is intended to perform, under the assumption that such an abstraction is required.

To formulate all of these tasks so they can be presented to a language model, we format them as a text prompt, the completion of which constitutes a response to the task. For example, the `suggest-reasons` task might be formulated as follows.

```
List reasons why: <TARGET CLAIM>

Reasons:
* <REASON 1>
* <REASON 2>
* <REASON 3>
*
```

The model must then generate `<REASON 4>` to complete the prompt[3].

[3]The prompt templates used can be found at https://github.com/Hunt-Laboratory/language-model-optimization.

2.3. Data

Training, validation, and test datasets for each task, where possible, were generated from a collection of argument maps scraped from the online collaborative argument mapping platform *Kialo*[4]. The scrape was performed in January 2020 by Lenz et al. [25], and the data was provided to us on request. The dataset contains 180,736 claims across 563 argument maps, centered on contentions such as "Traditional bullfighting ... should be banned", "Climate change can be reversed", and "Darwinian evolution is philosophy not science." The maps have the structure of a simple argumentation framework: a tree where vertices are claims and edges denote *pro* or *con* relations. The maps underwent several preprocessing steps, the most noteworthy of which are summarized below.

- Maps were randomly assigned to training (60%), validation (20%), and test (20%) sets.
- Claims falling above a certain distance from the root claim in each map were filtered out because qualitative exploration suggested that the quality and coherence of claims decreased the further you go from the root claim. These claims have likely received less scrutiny on Kialo because they are less salient in the user interface. The depth of truncation differs slightly for each task, and is specified in Table 2.
- All *forks* (a claim with its children) and *branches* (a sequence of supporting claims from a leaf to the root of a tree) were extracted for each map.
- Depending on the task, the forks and branches were randomly or deterministically subsetted further to generate a greater number of training, validation, and test examples. For example, if a fork contained multiple reasons, any subset of those reasons could be included in a prompt for the `suggest-reasons` task, and any one of those reasons could be held out to serve as the "correct" completion for the purposes of supervised training and evaluation. This subsetting process leads to a combinatorial explosion in the number of candidate examples, so random selection was used to limit the size of the dataset, where resource constraints required it. At most, the training sets were limited to 50,000 examples, and the validation and test sets to 10,000 examples.

The final number of examples in each of the dataset splits for each task are provided in Table 2. Note that for both the `suggest-copremise` and `suggest-abstraction` tasks, there is no training or validation data available because Kialo does not support co-premises or tag revisions of claims that could be considered abstractions. For the same reasons, the test set for these tasks is unlabelled, so performance on this task can only be evaluated by human raters.

3. Optimization

In this section we describe the strategies, software and hardware used for optimizing the foundation model.

[4]See https://www.kialo.com/.

Table 2
The sizes of the datasets for each task we investigated. The asterisk on the test set for the final two tasks indicates that it was unlabelled, so could only be used when manually evaluating model outputs.

		Dataset size		
Task	Truncation Depth	*Training*	*Validation*	*Test*
`suggest-reasons`	4	50,000	10,000	10,000
`suggest-objections`	4	50,000	10,000	10,000
`suggest-conclusion`	6	15,250	4,503	4,823
`suggest-intermediary-claims`	4	29,894	6,514	8,766
`suggest-copremise`	4	0	0	1,043*
`suggest-abstraction`	4	0	0	8,766*

3.1. Strategies

There is a rapidly growing literature on strategies for optimizing pre-trained language models to perform specific tasks.

Prompt programming refers to one family of methods, in which the pre-trained model weights are taken as fixed but the structure of the text prompt is tweaked to improve the quality of the output [26]. Exploration of different prompt formats can be systematic, but there is an art to crafting better prompts, guided by heuristics. So-called *zero-shot* prompts only contain task-specific instructions and the details of the specific instance of the task being performed. In contrast, *few-shot* prompts contain additional complete examples of the task being performed. In one context, the performance gains afforded by the inclusion of an additional example within a few-shot prompt have been observed to be roughly equivalent to tuning the full model on 100 examples [27]. In other contexts, zero-shot prompts significantly outperform few-shot prompts [8]. When using few-shot prompts, systematic experimentation can help determine which examples to include in the prompt [28], and in what order to list them [29]. Another prompt programming strategy is to formulate tasks in a common template, such as question answering [30] or textual entailment [10].

In a related approach, a short sequence of additional tokens with randomly initialized embeddings (known as a *soft prompt*) is prepended to a minimal input prompt containing the details of the specific instance of the task to be performed. The embeddings for these additional tokens are tuned in a supervised fashion while all other model parameters remain fixed [31, 32, 33]. In this way, the "wording" of the prompt can be continuously optimized using conventional gradient-based optimization methods.

There are a number of proposed strategies for selectively tuning a subset of the parameters in the main body of the model, short of tuning the full model. For example, tuning only the bias parameters can be both computationally efficient and effective [34, 9]. Alternately, *meta-tuning* describes the approach in which the foundation model is first tuned for a general task such as *instruction following* or *question answering*, before being applied to specific tasks of interest [30, 6].

Given our focus on exploring the applicability of pre-trained language models across multiple argumentative reasoning tasks—particularly in data-limited settings—we selected 5 optimization

strategies that we evaluated (data and funding permitting) for each of the 6 tasks. Informed by the above literature, the strategies evaluated were:

- zero-shot prompt, no tuning
- few-shot prompt, no tuning
- soft prompt + zero-shot prompt, only the soft prompt tuned
- zero-shot prompt, only bias parameters tuned
- zero-shot prompt, all parameters tuned

3.2. Software

Model tuning, evaluation, and text generation were performed in Python using PyTorch [35] via the Hugging Face transformers library [36], with some custom class extensions to account for bespoke data loading, logging of evaluation metrics, and the insertion of soft prompts. We used the WarmupLR learning rate scheduler with the AdamW optimizer, a batch size of 32, and continued tuning until the validation loss (evaluated relatively infrequently due to computational cost) was observed to increase. The code for creating, tuning, and generating text in the presence of soft prompts was adapted from Parker [37].

Training large neural networks while avoiding out-of-memory errors can be challenging. To manage parallel and efficient training and evaluation of GPT-Neo on limited hardware we used the Zero Redundancy Optimizer (ZeRO, stage 2), as implemented in the DeepSpeed library [38]. Within this framework, optimizer states and model weights are partitioned across parallel processes such that each process updates only its partition, and retains only the gradients corresponding to its portion of the optimizer states, whilst also offloading optimizer memory and computation to the CPU. This avoided out-of-memory errors and allowed training to be performed.

3.3. Hardware

Training and evaluation were performed remotely on a commercial virtual machine with four NVIDIA Quadro RTX 6000 GPUs, twenty-four AMD EPYC 7502P (2.50 GHz) virtual CPUs, and 2.78TB of storage. In total, the virtual machine had 96GB of virtual RAM (24GB per GPU), and 184 GB of conventional RAM. Total cloud compute costs were about USD 1250, and the tuning for all tasks and strategies took place over 228 hours.

4. Evaluation

In this section we describe the methods used to evaluate each optimization strategy on each task, along with our results.

4.1. Methods

Where possible, we evaluated each application of an optimization strategy to a task using both automated (intrinsic) and manual (extrinsic) methods. The inclusion of manual evaluation was to provide more interpretable insight into the quality of the model outputs, and to allow us to evaluate model outputs on unsupervised tasks where only prompts—not responses—were available (namely, `suggest-copremise` and `suggest-abstraction`).

4.1.1. Automated

For each task that included responses in the test set, we calculated the perplexity of each approach across all examples in the test set. Perplexity is a standard evaluation measure for language models in the supervised setting and is equal to the exponential of the average cross-entropy across tokens in the evaluation text, relative to the model. The lower the perplexity of a sequence of tokens, the greater the likelihood the model assigns to that sequence. Perplexity is not a measure of reasoning quality specifically, but in this context captures how likely the model was to generate the "correct", human-written argumentative claims that form the gold standard responses to the prompts.

We calculated perplexity in two ways: (a) using the mean cross-entropy across all tokens (in both the prompts and responses) and (b) using the mean cross-entropy across only the response tokens (though the prompts were still fed through the model to allow it to condition on them). Perplexity across all the tokens is substantially lower because they contain repetitive boilerplate (which could be memorized during tuning) and, further, this measure corresponds to the loss function on which the models were tuned, where such tuning occurred. That said, the second perplexity measure (considering only the response tokens) is perhaps more meaningful, given that we ultimately care about generating unknown outputs and not regenerating the input prompts.

4.1.2. Manual

Manual evaluation was performed using a bespoke method and rubric. We randomly sampled 100 examples from the test set for each task. Then, under all optimization strategies for each task, we generated sample responses of length 150 tokens for each of these 100 examples, conditioned on the appropriately formatted prompt. The temperature used for generation was 0.9. These generated outputs were cleaned according to the following rules.

- Rounded parentheses and their contents were removed, along with asterisks, underscores, backticks, any leading numbered list indices (e.g. "`1. `"), trailing whitespace, and all characters before the first letter, number, or quotation mark.
- The text was split into lines, and lines into sentences. Only the first line was retained and, unless the task was `suggest-intermediary-claims`, only the first sentence on the first line. The first letter was capitalized.
- If the task was `suggest-intermediary-claims`, the string was split on the claim delimiter (either "` ~ `" or "` => `"), and the first and last claims were removed on the assumption that

Table 3
The rating rubric used by raters to evaluate coherence of suggestions generated by the model.

1	Incoherent −	Suggestion (as written) is not relevant or coherent, and there is no insight to be gained from it.
2	Incoherent +	Suggestion (as written) is not relevant or coherent, but the suggestion prompts the user to think of adjacent ideas or suggestions that are relevant and coherent.
3	Coherent −	Suggestion (as written) is relevant and coherent, but some editing is required to be usable.
4	Coherent +	Suggestion (as written) is relevant and coherent, and would be usable as written.

they had been correctly reproduced from the input prompt.

The generated outputs were pooled with the human-generated claims from Kialo (to provide a human benchmark) and sorted according to randomly assigned IDs for each test example, such that tasks and strategies appeared in a random order, but outputs for each example task appeared consecutively. The example tasks and responses were presented to raters in this order (to reduce cognitive switching costs), and formatted as small argument maps using a custom interface[5], in which the claims to be rated were highlighted. Raters were blind to the source of the highlighted claims.

Each generated output was rated for coherence, using the rubric in Table 3. In this context, a claim is understood to be coherent (either "Coherent−" or "Coherent+") if it is (a) able to be understood, and (b) is logically consistent with neighboring claims, in the manner implied by its position in the argument map. Note, claims can be coherent without being true[6]. We chose to evaluate coherence because it arguably represents the minimum requirement for generated model outputs to be useful to a human analyst, and is a dimension of quality against which all six tasks can be evaluated.

The rubric and the rating interface were developed over two pilot rating rounds. Each model output was rated once by one of two raters (the authors of this study), both of whom are familiar with argument mapping conventions.

4.2. Results

In all training runs, learning was successful with an initial rapid decrease in loss, followed by a plateauing of the loss functions on both the training and validation sets. Severe overfitting (e.g. a U-shaped validation loss curve) was not seen within the training durations observed. As more parameters were made available for tuning, fewer examples were needed to reach the point of (mild) overfitting. For example, when tuning all parameters in the model, the model started to overfit less than 10% of the way through the first epoch. In contrast, the soft prompt and bias-only tuning strategies took at least one full epoch to reach the point of overfitting.

[5]A variant of the interface at https://luke-thorburn.github.io/argument-processor/.

[6]We originally included truth as a second dimension in the rating rubric. However, in practice the truth status of most model suggestions was either not able to be assessed (because they were normative or nonsensical), or was not verifiable in the time available.

4.2.1. Automated

The results of the automated evaluation are shown in Table 4. Across all strategy/task combinations studied, full text perplexity ranges between about 5 and 25. For reference, GPT-Neo-2.7B achieves a perplexity of 5.646 on its test set from The Pile [2]. The zero-shot, no tuning strategy consistently performed the most poorly, whilst the soft prompt strategy produced the lowest perplexities observed across all tasks.

When considering perplexity calculated only using the response tokens, the picture changes. In general, perplexity values are much higher here due to the exclusion of repetitive prompt boilerplate. The soft prompt strategy still performs competitively, but it is the few-shot strategy with no tuning that produced the lowest perplexity across all tasks. In contrast, the strategies with greater numbers of parameters tuned often performed more poorly, especially so in the case of the bias-only tuning strategy. This may be because they overfit to the prompt boilerplate at an earlier checkpoint, but this was not noticed in the training metrics because they only included the full text (both prompt and response).

4.2.2. Manual

The results of the manual evaluation are shown in Table 5. In general, the picture emerging from the manual evaluations is less clear than that of the automated evaluations, with different optimization strategies performing best on different tasks. This may in part be due to the relatively small sample of 100 examples rated, as well as noise due to the imperfect reliability of the rating scale.

With a few exceptions, the rate at which models produced coherent responses ranged roughly between 15% and 50%, which may be acceptable in a human-in-the-loop setting where multiple suggestions can be generated concurrently and those that are incoherent quickly discarded. Notably, the relatively more subtle tasks `suggest-abstraction` and `suggest-copremise` achieved coherence rates of 18% and 25% using merely a few-shot prompt and no parameter tuning.

We also rated the original human outputs from Kialo, where available, to provide a benchmark. Whilst better than all the language model outputs, the human benchmark is relatively low, not rising above 82% coherence. This reinforces the difficulty of the tasks (at least when posed to crowdsourced teams) and also raises questions about the quality of Kialo as a source of training data.

Figure 1 includes two examples of generated model outputs for the `suggest-objections` task. The complete set of generated examples along with their ratings can be found in the project GitHub repository[7].

From one perspective, coherence is a low bar. A suggestion can be coherent without being new, true, important, or eloquent. On the other hand, coherence is a significant milestone, revealing an ability to abide by conventional rules of logical argumentation. The observed coherence rates were achieved in a model that is at least two orders of magnitude smaller than commercial models that are state-of-the-art [39], with limited exploration of the space of optimization regimes. It is foreseeable that with larger language models and more dedicated

[7] Available at https://github.com/Hunt-Laboratory/language-model-optimization.

Claim:

`Cultural appropriation is wrong.`

Objections:

- `Integrating different cultures is one of the main way for cultures to develop themselves.`
- **`People who learn other cultures will become more tolerant, open minded and open to new experiences.`**

(a) Example of model output that was rated "Coherent +".

Claim:

`Private schools preserve traditions that are absent, or otherwise impractical to maintain, in the state system.`

Objections:

- `Given this has never been tried, and no examples are given, there is no reason to assume these traditions cannot be moved to a public system.`
- `Not all of these traditions are good, and many can perpetuate socio-economic divides far beyond the school system, for example by creating 'old boys clubs'.`
- **`This is a 'we must preserve these traditions for our daughters', rather than a 'it is to our children' argument.`**

(b) Example of model output that was rated "Incoherent −".

Figure 1: Two randomly selected examples of model-generated outputs for the `suggest-objections` task. We present the generated model output (in bold) and the context on which the model was conditioned in a form abstracted from the particular prompt templates used. Example (a) was generated by GPT Neo with an optimized soft prompt, example (b) was generated by unfinetuned GPT Neo with a zero-shot prompt.

effort, an automated approach to argumentative reasoning tasks could reach coherence on par with that of Kialo users.

5. Conclusion

Recently, large language models have come to dominate the field of natural language processing, but arguably remain underexplored in the computational argumentation literature. In this paper, we systematically evaluated the performance of a 2.7 billion parameter pretrained language model across 6 argumentative reasoning tasks, using 5 different optimization strategies. With a few exceptions, the rate at which the models produced coherent responses ranged from 15-50%, compared to human performance of 65-82%. We share our finetuned models and code.

To our knowledge the language model studied is larger than those previously considered in the argumentation literature, but it has at least two orders of magnitude fewer parameters than those that are state of the art on other NLP tasks, and the labeled data used for finetuning was of dubious quality. Natural next steps would be to evaluate the performance of much larger pretrained language models on the same argumentative reasoning tasks, and to invest in the development of larger, high-quality labeled datasets of natural language argumentation to use for finetuning.

That said, language models fundamentally model statistical—rather than logical—relationships between words, and it is not clear whether bigger models and better data alone will be sufficient to produce reliably coherent results. Thus, it would be valuable to explore how language models could be combined with symbolic argumentation methods to improve the coherence of generated arguments.

Table 4
Perplexity scores on the test set, calculated separately for the full text (prompt and repsonse), and the response only. Lower perplexity scores are preferred, indicating that the test was assigned a higher likelihood by the language model. Note that `suggest-copremise` and `suggest-abstraction` were not evaluated because there was no human data to reference against, and `suggest-intermediary-claims` was not evaluated for the zero-shot optimization method because the output required particular formatting conventions which could not be plausibly conveyed in a zero-shot prompt (i.e. without concrete examples).

	GPT-Neo (zero-shot)	GPT-Neo (few-shot)	GPT-Neo (soft prompt tuned)	GPT-Neo (bias parameters tuned)	GPT-Neo (all parameters tuned)
	Full Text				
`suggest-reasons`	20.62	10.03	**9.99**	14.01	14.59
`suggest-objections`	20.10	10.39	**9.88**	12.91	13.42
`suggest-conclusion`	23.31	10.51	**8.67**	13.50	13.89
`suggest-intermediary-claims`		8.15	**4.69**	5.89	5.96
`suggest-copremise`					
`suggest-abstraction`					
Average	21.35	9.77	**8.31**	11.58	11.96
	Response Only				
`suggest-reasons`	203.37	**146.91**	378.24	754.77	298.97
`suggest-objections`	363.26	**68.28**	260.71	1375.35	757.49
`suggest-conclusion`	355.20	**223.70**	223.81	936.33	867.84
`suggest-intermediary-claims`		82.43	**63.65**	5260.00	635.38
`suggest-copremise`					
`suggest-abstraction`					
Average	307.28	**130.33**	231.60	2081.61	639.92

Table 5
Manual coherence ratings for samples of 100 examples from each test set (rubric in Table 3). The same explanation of the empty cells that is given in the caption of Table 4 holds, with the additional note that we did not manually evaluate the "few-shot, no tuning" strategy for the `suggest-reasons`, `suggest-objections`, and `suggest-conclusion` tasks. This was a strategic decision due to limited funds available for rating, and based on prior research indicating that well-formed zero-shot prompts had outperformed few-shot prompts in similar contexts. Given that the perplexity scores were lower for few-shot prompts than for zero-shot prompts across the response tokens, it would be interesting to perform this manual evaluation in future.

	GPT-Neo (zero-shot)	GPT-Neo (few-shot)	GPT-Neo (soft prompt tuned)	GPT-Neo (bias parameters tuned)	GPT-Neo (all parameters tuned)	Human
	Coherence (%)					
`suggest-reasons`	47.0		41.0	**53.0**	39.0	80.0
`suggest-objections`	37.0		33.0	**46.0**	33.0	82.0
`suggest-conclusion`	18.0		21.0	26.0	**32.0**	65.0
`suggest-intermediary-claims`		16.0	3.0	15.0	**24.0**	75.0
`suggest-copremise`		**25.0**				
`suggest-abstraction`		**18.0**				
Average	34.0	19.7	24.5	**35.0**	32.0	75.5

	Coherence (mean)					
`suggest-reasons`	2.26		2.17	**2.46**	2.13	3.36
`suggest-objections`	2.03		1.88	**2.24**	1.93	3.31
`suggest-conclusion`	1.51		1.75	1.75	**1.90**	2.82
`suggest-intermediary-claims`		1.54	1.15	1.56	**1.94**	3.17
`suggest-copremise`		**1.71**				
`suggest-abstraction`		**1.54**				
Average	1.93	1.60	1.74	**2.00**	1.98	3.17

Acknowledgments

This research was funded by the Australian Department of Defence and the Office of National Intelligence under the AI for Decision Making Program, delivered in partnership with the Defence Science Institute in Victoria. The authors would like to thank other members of the Hunt Lab, particularly Tim van Gelder and Ashley Barnett for helpful discussions, and anonymous reviewers for their constructive feedback.

References

[1] D. C. Engelbart, Augmenting Human Intellect: A Conceptual Framework, Summary Report Project #3578, Stanford Research Institute, 1962. URL: https://apps.dtic.mil/sti/pdfs/AD0289565.pdf.

[2] S. Black, L. Gao, P. Wang, C. Leahy, S. Biderman, GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, Zenodo, 2021. doi:10.5281/zenodo.5297715.

[3] G. Branwen, GPT-3 Creative Fiction, 2020. URL: https://www.gwern.net/GPT-3.

[4] G. Branwen, GPT-3 Nonfiction, 2020. URL: https://www.gwern.net/GPT-3-nonfiction.

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language Models are Few-Shot Learners, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[6] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, Q. V. Le, Finetuned Language Models Are Zero-Shot Learners, CoRR (2021). doi:10.48550/arXiv.2109.01652.

[7] E. Perez, D. Kiela, K. Cho, True Few-Shot Learning with Language Models, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, J. W. Vaughan (Eds.), Advances in Neural Information Processing Systems, volume 34, Curran Associates, Inc., 2021, pp. 11054–11070. URL: https://proceedings.neurips.cc/paper/2021/file/5c04925674920eb58467fb52ce4ef728-Paper.pdf.

[8] T. Schick, H. Schütze, True Few-Shot Learning with Prompts—A Real-World Perspective, Transactions of the Association for Computational Linguistics 10 (2022) 716–731. doi:10.1162/tacl_a_00485.

[9] R. L. Logan, I. Balazevic, E. Wallace, F. Petroni, S. Singh, S. Riedel, Cutting Down on Prompts and Parameters: Simple Few-Shot Learning with Language Models, CoRR (2021). doi:10.48550/arXiv.2106.13353.

[10] S. Wang, H. Fang, M. Khabsa, H. Mao, H. Ma, Entailment as Few-Shot Learner, CoRR (2021). doi:10.48550/arXiv.2104.14690.

[11] R. El Baff, H. Wachsmuth, K. Al Khatib, M. Stede, B. Stein, Computational Argumentation Synthesis as a Language Modeling Task, in: Proceedings of the 12th International Conference on Natural Language Generation, Association for Computational Linguistics, Tokyo, Japan, 2019, pp. 54–64. doi:10.18653/v1/W19-8607.

[12] P. Clark, O. Tafjord, K. Richardson, Transformers as Soft Reasoners over Language, CoRR (2020). doi:10.48550/arXiv.2002.05867.

[13] T. Gurcke, M. Alshomary, H. Wachsmuth, Assessing the Sufficiency of Arguments through Conclusion Generation, CoRR (2021). doi:10.48550/arXiv.2110.13495.

[14] G. Skitalinskaya, J. Klaff, H. Wachsmuth, Learning From Revisions: Quality Assessment of Claims in Argumentation at Scale, in: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Association for Computational Linguistics, Online, 2021, pp. 1718–1729. doi:10.18653/v1/2021.eacl-main.147.

[15] Z. Jin, A. Lalwani, T. Vaidhya, X. Shen, Y. Ding, Z. Lyu, M. Sachan, R. Mihalcea, B. Schölkopf, Logical Fallacy Detection, 2022. doi:10.48550/arXiv.2202.13758.

[16] Ought Inc., Elicit, 2022. URL: https://elicit.org/, accessed: 2022-04-29.

[17] N. Slonim, Project Debator, in: Computational Models of Argument: Proceedings of COMMA 2018, 2018, p. 4. doi:10.3233/978-1-61499-906-5-4.

[18] N. Slonim, Y. Bilu, C. Alzate, R. Bar-Haim, B. Bogin, F. Bonin, L. Choshen, E. Cohen-Karlik, L. Dankin, L. Edelstein, L. Ein-Dor, R. Friedman-Melamed, A. Gavron, A. Gera, M. Gleize, S. Gretz, D. Gutfreund, A. Halfon, D. Hershcovich, R. Hoory, Y. Hou, S. Hummel, M. Jacovi, C. Jochim, Y. Kantor, Y. Katz, D. Konopnicki, Z. Kons, L. Kotlerman, D. Krieger, D. Lahav, T. Lavee, R. Levy, N. Liberman, Y. Mass, A. Menczel, S. Mirkin, G. Moshkowich, S. Ofek-Koifman, M. Orbach, E. Rabinovich, R. Rinott, S. Shechtman, D. Sheinwald, E. Shnarch, I. Shnayderman, A. Soffer, A. Spector, B. Sznajder, A. Toledo, O. Toledo-Ronen, E. Venezian, R. Aharonov, An Autonomous Debating System, Nature 591 (2021) 379–384. doi:10.1038/s41586-021-03215-w.

[19] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, N. Fiedel, PaLM: Scaling Language Modeling with Pathways, 2022. doi:10.48550/arXiv.2204.02311.

[20] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving Language Understanding by Generative Pre-Training, Technical Report, OpenAI, 2018.

[21] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language Models Are Unsupervised Multitask Learners, Technical Report, OpenAI, 2019. URL: http://www.persagen.com/files/misc/radford2019language.pdf.

[22] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, C. Leahy, The Pile: An 800GB Dataset of Diverse Text for Language Modeling, CoRR (2021). doi:10.48550/arXiv.2101.00027.

[23] K. Marriott, P. Sbarski, T. van Gelder, D. Prager, A. Bulka, Hi-Trees and Their Layout, IEEE Transactions on Visualization and Computer Graphics 17 (2011) 290–304. doi:10.1109/TVCG.2010.45.

[24] T. van Gelder, P. Monk, Argument Mapping Short Course, 2017.
[25] M. Lenz, P. Sahitaj, S. Kallenberg, C. Coors, L. Dumani, R. Schenkel, R. Bergmann, Towards an Argument Mining Pipeline Transforming Texts to Argument Graphs, in: Computational Models of Argument: Proceedings of COMMA 2020, 2020, pp. 263–270. doi:10.3233/FAIA200510.
[26] T. Gao, Prompting: Better Ways of Using Language Models for NLP Tasks, The Gradient (2021). URL: https://thegradient.pub/prompting/.
[27] T. L. Scao, A. M. Rush, How Many Data Points is a Prompt Worth?, CoRR (2021). doi:10.48550/arXiv.2103.08493.
[28] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, W. Chen, What Makes Good In-Context Examples for GPT-3?, CoRR (2021). doi:10.48550/arXiv.2101.06804.
[29] Y. Lu, M. Bartolo, A. Moore, S. Riedel, P. Stenetorp, Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity, CoRR (2021). doi:10.48550/arXiv.2104.08786.
[30] R. Zhong, K. Lee, Z. Zhang, D. Klein, Meta-tuning Language Models to Answer Prompts Better, CoRR (2021). doi:10.48550/arXiv.2104.04670.
[31] B. Lester, R. Al-Rfou, N. Constant, The Power of Scale for Parameter-Efficient Prompt Tuning, CoRR (2021). doi:10.48550/arXiv.2104.08691.
[32] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, J. Tang, GPT Understands, Too, CoRR (2021). doi:10.48550/arXiv.2103.10385.
[33] G. Qin, J. Eisner, Learning How to Ask: Querying LMs with Mixtures of Soft Prompts, CoRR (2021). doi:10.48550/arXiv.2104.06599.
[34] E. B. Zaken, S. Ravfogel, Y. Goldberg, BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models, CoRR (2021). doi:10.48550/arXiv.2106.10199.
[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An Imperative Style, High-Performance Deep Learning Library, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.
[36] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, A. M. Rush, Transformers: State-of-the-art Natural Language Processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 38–45.
[37] K. Parker, soft-prompt-tuning, 2021. URL: https://github.com/kipgparker/soft-prompt-tuning, accessed: 2021-11-20.
[38] J. Rasley, S. Rajbhandari, O. Ruwase, Y. He, DeepSpeed: System Optimizations Enable Training Deep Learning Models with Over 100 Billion Parameters, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Association for Computing Machinery, New York, NY, USA, 2020, pp. 3505–3506.
[39] W. Fedus, B. Zoph, N. Shazeer, Switch Transformers: Scaling to Trillion Parameter Models

with Simple and Efficient Sparsity, Journal of Machine Learning Research 23 (2022) 1–39. URL: http://jmlr.org/papers/v23/21-0998.html.