– Independent Work Report Spring 2023 –

Using large language models to generate descriptions of the crystal structure of novel materials

Lucas Gen
Adviser: Adji Bousso Dieng

Abstract

In this paper we explore how the power of current open source large language models can be adapted through fine tuning and used for practical tasks that could help researchers reason about new materials better than ever before. Specifically, we fine tune GPT2 to generate descriptions of the crystal structure of materials, evaluate its results, and then explore areas for further potential improvement. Most notably, we modify the original GPT2 tokenizer to process material formulas in a more intuitive manner and evaluate the results of these modifications. While there is still a lot of room for improvement, our process and results are promising, and more work needs to be done to continue to apply the power of natural language and large language models in effective ways to a wide variety of fields and tasks.

1. Introduction

The recent developments in the field of natural language processing have garnered widespread attention across the world and opened the eyes of many people to the power of this technology and the knowledge that is captured in natural language. While there have been many significant advancements in this field in recent years that have led us to this point, the virality of products like ChatGPT have arguably done more than any technological advancement in terms of awakening the masses to the potential of natural language processing to solve some of the world's pressing issues in a variety of fields. Since this boom in NLP activity, we've seen companies race to push competitive products to the market and a huge increase in the serious conversations about artificial general intelligence, or AGI. However, while there is a commercial race towards AGI which has so far been primarily driven by for-profit companies, there are also a lot of pressing real-world

problems that can be helped right now by applying the artificial intelligence that we have today to real world problems. Arguably, the current commercialized state of the artificial intelligence world is far too focused on reaching AGI and is not putting enough effort into solving the current problems that we face today. In the words of Professor Adji Bousso Dieng of Princeton University, "The attention skew in the field of AI causes real societal and socio-economic threats and prevents us from leveraging AI to solve many pressing problems facing humanity—problems whose solutions don't require the development of agents with superhuman intelligence in the first place." [1] Therefore, there needs to be a larger community of people focused on applying the models we already have to solve the issues of today and making a real difference in the world. There are nearly endless avenues to explore when looking to apply the power of pretrained large language models to real world problems, and in this project we explore one route. Specifically, we focus on fine tuning an open source pretrained large language model to generate descriptions of the crystal structure of materials, given only the material's formula.

2. Motivation

Working with materials and creating new ones is known to be an extremely slow and expensive process that often requires a lot of trial and error[2]. However, this is an area of very active research and is absolutely critical for driving innovation in a number of fields. Therefore in this project we explore one way to apply the power of current large language models to assist these very important researchers in their work. We do this by fine tuning a large language model to generate descriptions of the crystal structures of materials given only the formula of the material in question. This information includes predicted bond lengths and angles in between atoms which affect the various properties of the material. By fine tuning this model on all the descriptions of the crystal structures of all the materials we currently know, we hope that the machine learning model will be

[1] Prud'homme, Benjamin; Régis, Catherine; Farnadi, Golnoosh; Dreier, Vanessa; Rubel, Sasha; d'Oultremont, Charline. "Missing Links in AI Governance". UNESCO. 2023. Pg. 53

[2] Edwards, Carl; Lai, Tuan; Ros, Kevin; Honke, Garrett; Cho, Kyunghyun; Ji, Heng. "Translation between Molecules and Natural Language". University of Illinois Urbana-Champaign; X, the Moonshot Factory; New York University; Genetech. November 3, 2022. https://arxiv.org/pdf/2204.11817.pdf

able to "learn" the patterns of how different elements combine to create materials that have certain structures. After this process, we hope that the model will be able to apply it's learned knowledge to new materials that it has never seen before and that we don't have information about yet. Using a model like this, we hope that researchers can easily start to learn more about the structure of these new materials which can help them reason about the properties of materials that don't exist yet.

3. Related Work

While there is not an enormous amount of research that has been done attempting to tackle this exact problem (which is part of why we chose it), there are other natural language processing papers that explore similar problems in the field. One paper in particular by Edwards et al. gave us inspiration[3]. This paper presents a new "self-supervised learning framework for pretraining models on a vast amount of unlabeled natural language text and molecule strings."[4] Similarly to this project, this paper uses natural language to reason about the properties of materials or molecules that don't yet exist. Specifically, they focus on two tasks. First of all, they generate descriptions of molecules, given the molecule's formula. Secondly, they do exactly the inverse by generating a molecule itself based on a text description of the properties of the desired molecule[5]. Whereas our paper focuses mainly on the fine tuning of a large language model to perform well on a specific task, this paper proposes a more broad framework for training a model to perform well on specific tasks, including doing pretraining on unstructured data before any fine tuning occurs. After performing both pretraining and fine tuning to create their model, this paper found that these improvements increased the accuracy of their model in a significant way, although notably not enough to outperform language models with more parameters but weren't fine tuned for the task at hand[6]. In addition to providing a useful framework for this kind of work, this paper also helps to set

[3]Edwards, Carl; Lai, Tuan; Ros, Kevin; Honke, Garrett; Cho, Kyunghyun; Ji, Heng. "Translation between Molecules and Natural Language".

[4]Edwards, Carl; Lai, Tuan; Ros, Kevin; Honke, Garrett; Cho, Kyunghyun; Ji, Heng. "Translation between Molecules and Natural Language".

[5]Edwards, Carl; Lai, Tuan; Ros, Kevin; Honke, Garrett; Cho, Kyunghyun; Ji, Heng. "Translation between Molecules and Natural Language".

[6]Edwards, Carl; Lai, Tuan; Ros, Kevin; Honke, Garrett; Cho, Kyunghyun; Ji, Heng. "Translation between Molecules and Natural Language".

realistic expectations for the scale of improvement that can be achieved through fine tuning for a similar task.

4. The Model

Given the advancements in natural language processing in recent years, there are a number of high quality open source large language models to choose from for any given fine tuning project. BERT, GPT, and T5 are some of the most popular pretrained large language models available on platforms like Hugging Face, although a number of less well-known but high-quality models have recently come forth which also need to be explored[7]. Specifically, models like LLaMA from Meta are reported to have very impressive performance and there are even some experiments being done that are able to run models like these on much lower powered hardware than most machine learning models[8][9]. All of these models have different architectures and were trained with different objectives which affects the tasks that they perform best at. For example, BERT was trained with a masked language modeling approach, in which predictions for words are made using the full context of the word, on both the left and right sides. Because of how it's trained, BERT performs best at tasks that allow it to know the full context and meaning of the input text. Tasks like question answering, summarization, language translation, are all good tasks for a model with the architecture and training objective of BERT[10]. Since this project focuses on a text-generation prompt, in which the model is given just a formula and asked to generate descriptions of the material at hand, we chose to work with GPT2 for this task. GPT2 is a model that utilizes a decoder-only architecture, a left-to-right language objective during training, and operates autoregressively[11]. GPT2 is designed to excel at text-generation from limited one-sided prompts, and that is exactly what we would like it

[7] Kazi, Suleman. "Top Large Language Models (LLMs): GPT-4, LLaMA, FLAN UL2, BLOOM, and More". Vectara. March 28, 2023. https://vectara.com/top-large-language-models-llms-gpt-4-llama-gato-bloom-and-when-to-choose-one-over-the-other/.

[8] "Introducing LLaMA: A foundational, 65-billion-parameter large language model".

[9] Edwards, Benj. "You can now run a GPT-3-level AI model on your laptop, phone, and Raspberry Pi".

[10] Muller, Britney. "BERT 101 State Of The Art NLP Model Explained". Hugging Face. March 2, 2022. https://huggingface.co/blog/bert-101.

[11] "Fine-tuning GPT-2 from human preferences". OpenAI. September 19, 2019. https://openai.com/research/fine-tuning-gpt-2.

to do. Since GPT2 is one of the most performant open source pretrained large language models whose training objective is similar to the fine tuning task we want to train it for, we chose to use it for this project.

5. Dataset

In order for GPT2 to perform it's best on the particular task we have for it, we fine tuned it on one of Vertaix's internal datasets. This dataset consists of 145825 rows of data, with each row containing descriptions of the crystal structure of each material. Please see Figure 1 for an example of what each row of this dataset looks like. Also note how each line of data starts with the formula of the material in question, followed by the description. This structure allows us to fine tune GPT2 with it's original text-generation goal while still accomplishing our goal of generating a description of the crystal structure of each material only given the formula. It should also be noted that, while the amount of data used for fine tuning a large language model varies a lot by the size of the model, the task at hand, and many other factors, the number of rows of data in our dataset is not uncommonly low. While more data is almost always better when it comes to deep learning, even industry leaders like OpenAI have achieved remarkable results fine tuning with even a much smaller dataset than we have[12]. However, many other finetuning tasks utilize a lot more samples. It's also worth noting at this point that a technique called few-shot learning has become particularly powerful as pretrained large language models get more and more effective. Essentially, this term refers to the act of simply explaining the task at hand to a large language model, giving it a few examples and asking it to perform the rest, without ever training it or changing the weights[13]. However, while this technique may be surprisingly effective for certain tasks, there are many tasks such as the one we propose in this project for which this is not a viable option. Not only is there barely enough context size to feed most of the large language models one example from our dataset, but empirically, even the most state of the art proprietary large language models don't perform well with the task we give it.

[12] "Fine-tuning GPT-2 from human preferences".

[13] Oppermann, Artem. "What is Few-Shot Learning?". builtin. April 6, 2023. https://builtin.com/machine-learning/few-shot-learning.

For example, please see Figure 2 for GPT3.5's output for this specific task using few-shot learning. By comparing Figure 2 to Figure 1, we can see that clearly few-shot learning is not a viable option for this particular task.

> Li2NbV3O8 is Spinel-derived structured and crystallizes in the trigonal R-3m space group. Li(1) is bonded to one O(1) and three equivalent O(2) atoms to form LiO4 tetrahedra that share corners with three equivalent Nb(1)O6 octahedra and corners with nine equivalent V(1)O6 octahedra. The corner-sharing octahedral tilt angles range from 58,Äì60¬. The Li(1),ÄìO(1) bond length is 2.06 Ö. All Li(1),ÄìO(2) bond lengths are 2.02 Ö. Nb(1) is bonded to six equivalent O(2) atoms to form NbO6 octahedra that share corners with six equivalent Li(1)O4 tetrahedra and edges with six equivalent V(1)O6 octahedra. All Nb(1),ÄìO(2) bond lengths are 2.03 Ö. V(1) is bonded to two equivalent O(1) and four equivalent O(2) atoms to form VO6 octahedra that share corners with six equivalent Li(1)O4 tetrahedra, edges with two equivalent Nb(1)O6 octahedra, and edges with four equivalent V(1)O6 octahedra. Both V(1),ÄìO(1) bond lengths are 2.03 Ö. All V(1),ÄìO(2) bond lengths are 2.08 Ö. There are two inequivalent O sites. In the first O site, O(1) is bonded to one Li(1) and three equivalent V(1) atoms to form distorted OLiV3 tetrahedra that share corners with three equivalent O(1)LiV3 tetrahedra, corners with nine equivalent O(2)LiNbV2 trigonal pyramids, and edges with three equivalent O(2)LiNbV2 trigonal pyramids. In the second O site, O(2) is bonded to one Li(1), one Nb(1), and two equivalent V(1) atoms to form distorted OLiNbV2 trigonal pyramids that share corners with three equivalent O(1)LiV3 tetrahedra, corners with nine equivalent O(2)LiNbV2 trigonal pyramids, an edgeedge with one O(1)LiV3 tetrahedra, and edges with two equivalent O(2)LiNbV2 trigonal pyramids.

Figure 1: This figure shows one of the rows from the dataset used to fine-tune GPT2 for our task. Notice how the line starts with the material's formula and then proceeds to describe its crystal structure.

> Li2NbV3O8 is a lithium niobium vanadium oxide compound. Its crystal structure is described as a layered perovskite-type structure.
> The layers consist of NbO6 octahedra and VO6 octahedra that are connected through corners, and Li+ cations that are located between the layers to balance the charge. The layers are stacked on top of each other along the c-axis of the crystal structure.
> The crystal symmetry of Li2NbV3O8 is orthorhombic with space group Pnma (No. 62). The lattice parameters are a = 11.464 Å, b = 7.677 Å, and c = 5.377 Å.

Figure 2: This figure shows the prediction that a larger and more advanced large language model (GPT3.5) predicts for this specific task when using zero-shot learning. Comparing this to Figure 1, it's clear that few-shot learning for many tasks is not a viable option.

6. Tokenizer

When looking to fine tune GPT2 for this specific task, one of the main areas that we looked into in order to improve the performance of the overal model is the tokenizer used. While GPT2 is great at working with normal day-to-day text and uses a popular method of tokenization known as subword tokenization that is remarkably effective in a wide variety of use cases, the tokenizer is not specifically optimized for the kind of text that we want it to work for on this task. To be more specific, when looking through the dataset from Section 5, we can see that the text contains words that most likely aren't very common in the datasets that GPT2 was originally trained on. Even the element symbols that make up the material formulas will not have been seen very much in the original pretraining. Furthermore, there are a lot of scientific words and number combinations that might be very rare in the original pretraining dataset but very common in the fine tuning one. For example, our fine tuning dataset has a lot of information about bond lengths and angles which are very important to the overall structure of the materials but appear in ways that aren't. For this reason, one of our theories going into this project was that the performance of our final model could be significantly improved by modifying GPT2's original tokenizer to better tokenize the words and formulas that our dataset contains.

6.1. Formulas

For example examine how the original GPT2 tokenizer tokenizes the following material formula in Figure 3: "Li2NbV3O8". Notice how the tokenizer doesn't natively recognize each element

['Li', '2', 'N', 'b', 'V', '3', 'O', '8']

Figure 3: How the original GPT2 tokenizer tokenizes a random formula. Note that it's not processing the formulas element-by-element and number-by-number, but rather in whatever subword it defaults to.

and therefore the subword tokenization method that GPT2 uses natively breaks down some of the elements, such as separating the "N" and "b" of "Nb" into two separate tokens. Ideally we really want our model to intuitively understand that these elements are atomic units and they build on each

other together to create materials that reflect the properties of these atomic units. Therefore a better tokenizer for this particular fine tuning task might instead break the same material down as seen in Figure 4.

['Li', '2', 'Nb', 'V', '3', 'O', '8']

Figure 4: How our modified tokenizer tokenizes the same material. Notice how it now processes each element as atomic units.

Notice the small but important difference in that the tokenizer separates the material into the correct atomic units, with each token of this sequence referring either to an element specifically, or the number of atoms of each element. Logically, this makes more sense since we can think about Li2NbV3O8 as two atoms of Li, one atom of Nb, three atoms of V, and 8 atoms of O, all put together.

6.2. Numbers

Another area in which we look for performance improvements is in how GPT2 handles numbers. When generating the descriptions of the crystal structure of materials, the bond lengths and angles are extremely important. Therefore, we want the model to have the tools to reason about these numbers in the best way possible. However, when we examined how GPT2 natively handles numbers, we again noticed room for improvement. For example, when tokenizing the following sequences of numbers, we notice (as shown in Figure 5) that there is some unexpected behavior in how GPT2 tokenizes them. Why is "5324" broken up into two tokens? Ideally we really want this

"12 1 5324" $\rightarrow$ ["Ġ12", "Ġ1", "Ġ53", "24"]

Figure 5: A figure showing how GPT2 doesn't natively tokenize numbers digit-by-digit. Empirically, the behavior seems to be a little unpredictable with numbers often being grouped together like here, and sometimes split by digits.

tokenizer to behave very predictably on bond lengths and angles, which is why we again attempt to improve the performance of our model by limiting the model to only tokenize numbers one digit at a time. Please note that this idea was inspired and advised by Andre Niyongabo Rubungo who developed similar number modifications for the T5 tokenizer with promising results.

6.3. Implementation

In order to accomplish this, we create a custom GPT2 tokenizer that wraps around the open source GPT2 tokenizer. The code of this custom tokenizer can be found below in Figure 6. This custom tokenizer implements a very simple fix to unpredictable behavior of GPT2 when tokenizing numbers. Essentially, it looks through the text it's tokenizing and if it finds numbers, it tokenizes it digit by digit automatically. Otherwise it just passes everything else on to the original GPT2 tokenizer.

```
class CustomTokenizerGPT2(GPT2Tokenizer):
    def tokenize(self, text, **kwargs):

        words = text.split()
        tokens = []

        def is_number(s):
            pattern = "^␣*\(?-?\d+\.?\d*\(?\)?$"

            return bool(re.match(pattern, s))

        for index, word in enumerate(words):
            if index != 0:
                word = "␣" + word
            if is_number(word):
                tokens += super().tokenize(word[:2], **kwargs)
                tokens += list(word[2:])
            else:
                tokens += super().tokenize(word, **kwargs)
        return tokens
```

Figure 6: One simple way of wrapping the GPT2 tokenizer to create a custom class that forces it to process certain number formats that are common in our dataset in a more consistent and predictable way. Note that the "is_number" function can be modified and adapted to fit the task at hand. Check this model's GitHub repo for the most up-to-date regular expression being used for this task.

Now we show how to force the GPT2 tokenizer to tokenize each material's formula in the most ideal way. To do this is even simpler than dealing with the numbers since all we need to do is obtain a list of all the elements we want tokenized, and then add them to the tokenizer's dictionary and the model's embedding size. The code of how we did so is shown in Figure 7.

```
tokenizer = CustomTokenizerGPT2.from_pretrained(
    model_checkpoint)
symbol_list = ['H', 'He', 'Li', 'Be', 'B', 'C', 'N', 'O', 'F'
    , 'Ne', 'Na', 'Mg', 'Al', 'Si'...]
tokenizer.add_tokens(symbol_list)
model.resize_token_embeddings(len(tokenizer))
```

Figure 7: This figure shows how we add each element symbol the tokenizer's vocabulary so that it can tokenize formulas in a potentially better way for this task than its default behavior.

7. Results

One of the purposes of this project is to explore how effective fine tuning can be to a somewhat niche problem in a field where getting quality data is often difficult, slow and expensive. Therefore, this section we will explore the initial results of our fine tuning models, compare it against a more advanced model without fine tuning, and then explore the effectiveness of different hyperparameters that affect our model's performance at this task.

First, let's examine how our fine tuned model performs with different sized models. On Hugging Face, there are a number of different sized GPT2 models available, including GPT2 (124M parameters), GPT2-Medium (355M parameters), GPT-Large (774M parameters), and GPT2-XL (1.5 B parameters). In the scope of this paper, we focus both on the GPT2 model and the GPT2-Large model. We hope that these two model sizes will give us a good understanding of the potential that GPT2 has to perform well at this task while also demonstrating the improvements we can expect to see when scaling up our model to obtain even better results.

For example, after fine-tuning our model for three epochs across the whole training set and evaluating it on the testing dataset, we find that the GPT2 model achieves a final perplexity of 1.42, while the GPT2-Large model acheives a final perplexity of 1.37. As we can see, there is a fairly significant improvement in the quality of results from the 124M parameter model to the 774M parameter model. This seems to be a consistent conclusion across a number of research papers, and intuitively so. However, this is also one of the problems with the state of the AI world right now. As things currently stand, it seems as if most of the innovations with these models are coming from for-profit companies who have the money and infrastructure to build and run these very large

language models. As a result, they are the only ones who have the most performant models, and they make a lot of money by selling their services and then use that money to keep getting bigger and better models. This cycles ensures that these companies keep creating bigger and better models, and the rest of the world just has to rely on for-profit companies to make good decisions with the world's most powerful and potentially dangerous technology.

However, one positive result that comes from this is that even these relatively small models are able to perform the task at a higher level than the example in Figure 2 which uses a larger and more recent model. Of course if OpenAI wanted to create a model fine tuned for this task with their much larger models they would certainly acheive more impressive results. However, the ability to take a smaller open source model and apply it to a practical task is a good sign that we don't always need to rely on these large companies to apply machine learning to our practical day-to-day problems. In fact, we can outperform larger models fairly quickly with a bit of fine tuning. Please see Figure 8 for an example of the different outputs from GPT2-Large (modified tokenizer), and GPT2-Large (default tokenizer), compared with the ground truth text.

From this figure however, we can also see that there is a lot of room left for improvement. While the generated descriptions tend to look very passable at first glance, much of the information is not yet accurate and could lead to misinformation (as is always a threat with machine learning models). This problem of "hallucinations" is a well-documented problem with popular AI language models, and more work needs to be done on this model to keep it from making up information as much as possible. One potential route we explored in order to increase the model's "understanding" of the task we've given it is to adjust the tokenizer to better fit the text that the model sees during fine tuning. We already explained the adjustments we made to the GPT2 tokenizer in Section 6, so now we will discuss the results.

First of all, the perplexity for both GPT2 and GPT2-Large with the modified tokenizer vs native tokenizer was remarkably similar. However, while perplexity is a very common metric for NLP tasks like this, it is far from the only one. In this case, we look at other evaluation metrics to see if there is a difference between the different tokenizations. As it turns out, we do find some modest

improvements when evaluating the results of our model with the different tokenizers using a variety of metrics. For example, when taking the average BLEU score across the test set for both the modified and unmodified tokenizer models, the GPT2-Large (modified tokenizer) model results in an average BLEU score of 0.10, compared to 0.63 for the GPT2 (modified tokenizer) model and 0.79 for the GPT2-Large (unmodified tokenizer). While none of these scores are outstanding results by any means, it's extremely valuable to learn how we can keep pushing these models to perform better on specific tasks so that we can create useful fine tuned models that are effective in solving practical problems that we face without waiting for AGI or relying on corporate structures to let us use their biggest and most powerful models. By looking at Figure 9, we can also see the effect that our custom tokenizer has on the various Rouge scores. Notice how our modified tokenizer systematically performs slightly better across these tests. This seems to signal that this is a promising area to continue to develop in.

Looking to the future, there are a number of parameters that could prove to be useful in improving this model's performance and that should be explored.

8. Future Work

While our results so far are promising and educational, they are more a sign that more work needs to be done rather than any sort of conclusive evidence or finished product. One potential area of improvement is with the tokenizer we use for this task. We already made some modifications to the tokenizer in this project, and witnessed the marginal benefit it had on the model's performance, but this should only really act as encouragement that this path is promising and this should inspire us to keep working to improve it's performance. For example, instead of only adding new tokens to the tokenizer, perhaps we could also forget certain tokens that aren't likely at all to occur in the new dataset which could help improve the model's efficiency by reducing the model's embedding size. Another interesting area to pursue is the effect that pretraining GPT2 on scientific texts might have, even before exposing it to the fine tuning dataset. This is a fairly popular idea in the natural

language processing world (and one explored in Edwards et al.[14]), although the effects of it are mixed and not always conclusively positive or negative. However, when presenting a large language model with a task whose language is very different from the original one it was trained with, it's a very intuitive idea to get the model more used to this kind of language even before performing fine tuning and therefore is definitely worth an experiment at least to see if it improves this model at all.

Additionally, there are some other hyperparameters that should be experimented with to see if they lead to promising progress. For example, for this project we left the weight decay variable at a sensible default that is used by other projects utilizing GPT2. However, this parameter is used to reduce the weight sizes as much as possible in order to prevent overfitting and the task that we are fine tuning it for might be slightly more formulaic than most, resulting in a great tolerance larger, weights.

Finally, if people were to attempt to use this model to do actual research they would want to scale up the model to the largest size that they can run on their servers, increase the training time, and perhaps even train the model on every row of the dataset instead of splitting some into a testing dataset so they can get the best use out of all the data we have available.

9. Conclusion

In this project, we examine how one can fine-tune an open-source large language model to generate descriptions of the crystal structure of various materials when only given the materials formula. We witness the strong effect that fine tuning has, and it's ability for even smaller, less advanced models to perform better at specific tasks than larger, more advanced models without fine tuning. We then explore different areas that have the potential to improve this models performace even without scaling up the model or using more (often scarce) data. Most notably, we modify GPT2's default tokenizer to treat element symbols as atomic units as well as to tokenize numbers digit by digit rather than it's somewhat unpredictable default behaviour. Both of these modifications intuitively should help GPT2 process our dataset in a more effective way, and we examine the results of these

[14]Edwards, Carl; Lai, Tuan; Ros, Kevin; Honke, Garrett; Cho, Kyunghyun; Ji, Heng. "Translation between Molecules and Natural Language".

modifications for any improvements. While the improvements by the modified tokenizer to the model's performance were modest, it is definitely a promising avenue and one that should be looked into to continue to improve this model's performance.

References

Edwards, Benj. "You can now run a GPT-3-level AI model on your laptop, phone, and Raspberry Pi". arsTECHNICA. March 13, 2023. https://arstechnica.com/information-technology/2023/03/you-can-now-run-a-gpt-3-level-ai-model-on-your-laptop-phone-and-raspberry-pi/.

Edwards, Carl; Lai, Tuan; Ros, Kevin; Honke, Garrett; Cho, Kyunghyun; Ji, Heng. "Translation between Molecules and Natural Language". University of Illinois Urbana-Champaign; X, the Moon-shot Factory; New York University; Genetech. November 3, 2022. https://arxiv.org/pdf/2204.11817.pdf

"Fine-tuning GPT-2 from human preferences". OpenAI. September 19, 2019. https://openai.com/research/fine-tuning-gpt-2.

"Introducing LLaMA: A foundational, 65-billion-parameter large language model". MetaAI. February 24, 2023. https://ai.facebook.com/blog/large-language-model-llama-meta-ai/.

Kazi, Suleman. "Top Large Language Models (LLMs): GPT-4, LLaMA, FLAN UL2, BLOOM, and More". Vectara. March 28, 2023. https://vectara.com/top-large-language-models-llms-gpt-4-llama-gato-bloom-and-when-to-choose-one-over-the-other/.

Muller, Britney. "BERT 101 State Of The Art NLP Model Explained". Hugging Face. March 2, 2022. https://huggingface.co/blog/bert-101.

Oppermann, Artem. "What is Few-Shot Learning?". builtin. April 6, 2023. https://builtin.com/machine-learning/few-shot-learning.

Prud'homme, Benjamin; Régis, Catherine; Farnadi, Golnoosh; Dreier, Vanessa; Rubel, Sasha; d'Oultremont, Charline. Missing Links in AI Governance. UNESCO. 2023. Pg. 53

Radford, Alec; Wu, Jeffrey; Amodei, Dario; Amodei, Daniella; Clark, Jack; Brundage, Miles; Sutskever, Ilya. "Better language models and their implications". OpenAI. February 14, 2019. https://openai.com/research/better-language-models.

Ground Truth
Li5SbS4 crystallizes in the orthorhombic Pbca space group. There are five inequivalent Li sites. In the first Li site, Li(1) is bonded to one S(1), one S(2), one S(3), and one S(4) atom to form a mixture of edge and corner-sharing LiS4 tetrahedra. The Li(1),ÄìS(1) bond length is 2.43 Ö. The Li(1),ÄìS(2) bond length is 2.45 Ö. The Li(1),ÄìS(3) bond length is 2.54 Ö. The Li(1),ÄìS(4) bond length is 2.48 Ö. In the second Li site, Li(2) is bonded to one S(1), one S(2), one S(3), and one S(4) atom to form a mixture of distorted edge and corner-sharing LiS4 tetrahedra. The Li(2),ÄìS(1) bond length is 2.41 Ö. The Li(2),ÄìS(2) bond length is 2.50 Ö. The Li(2),ÄìS(3) bond length is 2.57 Ö. The Li(2),ÄìS(4) bond length is 2.55 Ö. In the third Li site, Li(3) is bonded to one S(1), one S(2), one S(3), and one S(4) atom to form a mixture of edge and corner-sharing LiS4 tetrahedra. The Li(3),ÄìS(1) bond length is 2.38 Ö. The Li(3),ÄìS(2) bond length is 2.46 Ö. The Li(3),ÄìS(3) bond length is 2.58 Ö. The Li(3),ÄìS(4) bond length is 2.57 Ö. In the fourth Li site, Li(4) is bonded to one S(1), one S(2), one S(3), and one S(4) atom to form a mixture of distorted edge and corner-sharing LiS4 tetrahedra. The Li(4),ÄìS(1) bond length is 2.43 Ö. The Li(4),ÄìS(2) bond length is 2.47 Ö. The Li(4),ÄìS(3) bond length is 2.58 Ö. The Li(4),ÄìS(4) bond length is 2.49 Ö. In the fifth Li site, Li(5) is bonded to one S(1), one S(2), one S(3), and one S(4) atom to form a mixture of distorted edge and corner-sharing LiS4 tetrahedra. The Li(5),ÄìS(1) bond length is 2.39 Ö. The Li(5),ÄìS(2) bond length is 2.56 Ö. The Li(5),ÄìS(3) bond length is 2.61 Ö. The Li(5),ÄìS(4) bond length is 2.50 Ö. Sb(1) is bonded in a 4-coordinate geometry to one S(1), one S(2), one S(3), and one S(4) atom. The Sb(1),ÄìS(1) bond length is 2.83 Ö. The Sb(1),ÄìS(2) bond length is 2.64 Ö. ...

GPT-Large with Default Tokenizer
Li5 Sb S4 crystallizes in the orthorhombic Cmcm space group. There are five inequivalent Li sites. In the first Li site, Li(1) is bonded to one S(1) and three equivalent S(2) atoms to form distorted LiS4 trigonal pyramids that share corners with two equivalent Li(2)S4 tetrahedra, corners with nine equivalent Li(1)S4 trigonal pyramids, and edges with three equivalent Li(2)S4 tetrahedra. The Li(1),ÄìS(1) bond length is 2.36 Ö. There are two shorter (2.40 Ö) and one longer (2.41 Ö) Li(1),ÄìS(2) bond length. In the second Li site, Li(2) is bonded to four equivalent S(2) atoms to form distorted LiS4 tetrahedra that share corners with two equivalent Li(1)S4 trigonal pyramids, corners with four equivalent Li(4)S4 tetrahedra, corners with eight equivalent Li(3)S5 trigonal bipyramids, an edgeedge with one Li(2)S4 tetrahedra, edges with two equivalent Li(3)S5 trigonal bipyramids, and a faceface with one Li(1)S4 trigonal pyramid. There are two shorter (2.47 Ö) and two longer (2.58 Ö) Li(2),ÄìS(2) bond lengths. In the third Li site, Li(3) is bonded to one S(1) and four equivalent S(1) atoms to form distorted LiS5 trigonal bipyramids that share corners with four equivalent Li(4)S4 tetrahedra, corners with four equivalent Li(1)S4 trigonal pyramids, an edgeedge with one Li(4)S4 tetrahedra, an edgeedge with one Li(5)S4 trigonal pyramid, and edges with four equivalent Li(1)S4 trigonal pyramids. The Li(3),ÄìS(1) bond length is 2.30 Ö. There are two shorter (2.47 Ö) and two longer (2.75 Ö) Li(3),ÄìS(1) bond lengths. In the fourth Li site, Li(4) is bonded to one S(3) and four equivalent S(2) atoms to form distorted LiS4 tetrahedra that share corners with two equivalent Li(2)S4 tetrahedra, an edgeedge with one Li(3)S4 tetrahedra, and edges with two equivalent Li(5)S4 trigonal pyramids. The Li(4),ÄìS(3) bond length is 2.53 Ö. There are two shorter (2.47 Ö) and two longer (2.50 Ö) Li(4),ÄìS(2) bond lengths. In the fifth Li site, Li(5) is bonded to one S(3) and four equivalent S(1) atoms to form distorted LiS5 trigonal bipyramids that share an edgeedge ...

GPT-Large with Modified Tokenizer
Li5 Sb S4 crystallizes in the orthorhombic Pb ca space group. The structure consists of a Li5 Sb S4 framework. There are five inequivalent Li sites. In the first Li site, Li(1) is bonded to one S(1), one S(2), one S(4), and one S(5) atom to form a mixture of distorted corner and edge-sharing LiS4 tetrahedra. The Li(1),ÄìS(1) bond length is 2.50 Ö. The Li(1),ÄìS(2) bond length is 2.49 Ö. The Li(1),ÄìS(4) bond length is 2.54 Ö. The Li(1),ÄìS(5) bond length is 2.41 Ö. In the second Li site, Li(2) is bonded to one S(1), one S(2), one S(3), and one S(4) atom to form distorted corner-sharing LiS4 tetrahedra. The Li(2),ÄìS(1) bond length is 2.51 Ö. The Li(2),ÄìS(2) bond length is 2.52 Ö. The Li(2),ÄìS(3) bond length is 2.48 Ö. The Li(2),ÄìS(4) bond length is 2.48 Ö. In the third Li site, Li(3) is bonded in a distorted trigonal planar geometry to one S(1), one S(2), and one S(4) atom. The Li(3),ÄìS(1) bond length is 2.48 Ö. The Li(3),ÄìS(2) bond length is 2.62 Ö. The Li(3),ÄìS(4) bond length is 2.50 Ö. In the fourth Li site, Li(4) is bonded in a trigonal planar geometry to one S(2), one S(3), and one S(5) atom. The Li(4),ÄìS(2) bond length is 2.57 Ö. The Li(4),ÄìS(3) bond length is 2.47 Ö. The Li(4),ÄìS(5) bond length is 2.50 Ö. In the fifth Li site, Li(5) is bonded in a 2-coordinate geometry to one S(1), one S(3), and two equivalent S(4) atoms. The Li(5),ÄìS(1) bond length is 2.66 Ö. The Li(5),ÄìS(3) bond length is 2.49 Ö. There is one shorter (2.54 Ö) and one longer (2.58 Ö) Li(5),ÄìS(4) bond length. Sn(1) is bonded in a 4-coordinate geometry to one S(1), one S(3), one S(4), and one S(5) atom. The Sn(1),ÄìS(1) bond length is 2.67 Ö. The Sn(1),ÄìS(3) bond length is 2.70 Ö. The Sn(1),ÄìS(4) bond length is 2.63 Ö. The Sn(1),ÄìS(5) bond length is 2.59 Ö. There are two inequivalent Sb sites. In the first Sb site, Sb (1) is bonded in a 2-coordinate ...

Figure 8: This figure shows the different outputs of our fine-tuned GPT2-Large models compared to the ground truth

Metric (avg across test set)	GPT2-Large (Default Tokenizer) Score	GPT2-Large (Modified Tokenizer) Score
Rouge 1 precision	0.315420681	0.347801657
Rouge 1 recall	0.687149927	0.786873484
Rouge 1 fmeasure	0.379542979	0.429164236
Rouge L precision	0.237186465	0.264559763
Rouge L recall	0.532699943	0.615768789
Rouge L fmeasure	0.286885274	0.32914991
Bleu	0.079248104	0.100031353

Figure 9: This figure shows the scores for various metrics commonly used to evaluate natural language processing tasks. Notice how the scores for the GPT2 model with our modified tokenizer are slightly higher.