

Introduction to Large Language Models (LLMs)
Prof. Tanmoy Chakraborty, Prof. Soumen Chakraborti
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi
Lecture 21
Instruction Tuning

Hello everyone, today we will discuss about instruction fine tuning or instruction tuning which is one of the key advancements in recent language modeling research which enables us to easily have a conversation with language models or simply put we can chat with language models. So first, a quick review. So in previous weeks, we have learned about decoder-based language models. So such models are trained on vast amount of text from the internet using the next word prediction task. As a result of this, these models learn to encode a lot of information about the world.



They also have the ability of understanding language to some extent. So these models are very powerful. They're pretty amazing, but we'll see in the upcoming slides that they have some major limitations. One note, these pre-trained language models, they're also known as base models. So I'll be using the term base models throughout the lecture.

So whenever I mention base models, it simply refers to pre-trained language models. For example, let's say we have given the following prompt. What is the national flower of India? We have prompted the language model with this question. Now, what can happen is that the language model can generate the following response, that what is the national animal of India? What is the national bird of India? So this response is nothing but the continuation of the prompt. And this is the result of the next word prediction that is happening after the prompt.

Why do we need instruction tuning?

Prompt	What is the national flower of India?
Response	



So here we see that the response contains questions which are quite common, which we can come across such questions in the web, where we see there's a web page on general knowledge questions about India. Such questions are very common. However, this is not the desired response. Because when we asked this question to the language model, we were expecting the answer. That is, the national flower of India is lotus.

This is what was the desired outcome. However, since the language model is just predicting the next word, so the response to a question could be that it may or may not follow the question. May or may not. Might follow the instruction, might not follow the instruction. Because as I said, it's simply just doing next word prediction at this point.

So the key takeaway from this slide is that next word prediction, which is what is governing this response generation, that does not necessarily ensure that the model understands or follows instructions. So the reason we need instruction tuning is that we want to teach the

language models how to follow and understand instructions. So multitask learning is another very important paradigm in natural language processing literature. So in classical multitask learning, what we do is we combine multiple tasks and we train the model, the language model on multiple tasks with the intention that these models will have a positive influence on each other and thereby the final outcome will be improved across all the tasks. So here, if we take a look at this example, during training, there are two tasks, sentiment analysis and machine translation.

Why do we need instruction tuning?

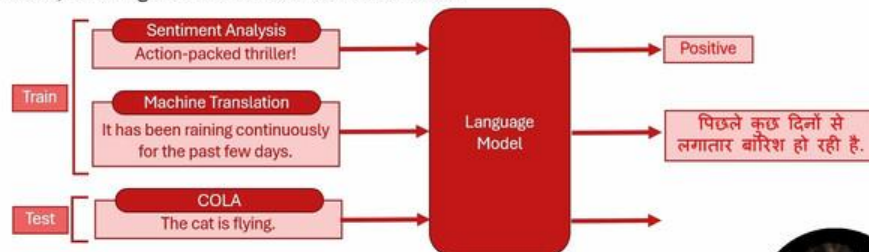
Prompt	What is the national flower of India?	Prompt	What is the national flower of India?
Response	What is the national animal of India? What is the national bird of India? ...	Response	The national flower of India is lotus .

Next word prediction does not necessarily ensure that the model understands and follows instructions.



Multi-task learning

Classical multi-task learning aims to improve performance by combining different training tasks, allowing them to benefit from each other.



These are two tasks with which we are training the model. So when I say I'm training the model, what I'm doing is I'm simply passing the input-output examples corresponding to these tasks to the language model. There's no explicit notion or explicit information that I'm supplying to the language model saying that oh this example corresponds to central analysis or this example corresponds to machine translation no such thing is there, we are just passing the input output examples and that is the way we are training the language model using multi-task learning. Now, suddenly during test time, say, the language model encounters a new task, which was not there during training, which is, for example, here it is COLA, which is the Corpus of Linguistic Acceptability. So what it does is it's actually a task from the glue benchmark.

So mainly the task is to classify whether the sentence that I'm supplying, whether it's a grammatical sentence or not. So, but to the language model, this is just a random example, and it has no idea what it is supposed to do with this example. So as a result, the language model will have no clue and will not generate a legitimate output in this case. Here a major obstruction is that when I'm training the model, I'm simply passing the examples. I'm not giving the language model any information about the task that I'm solving.

Hence, when it is encountering a new task during test time, it is clueless. Again, the takeaway from this slide is that using this approach, the model can only generalize within the training task. When it encounters a new task during test time, during inference, it struggles with it as it was not encountered during training. What can be a possible solution? Can we describe these tasks by instructions? Again, in this example, we can see that, various NLP tasks can be expressed by natural language instructions. So for sentiment analysis, I can pass the instruction, predict the sentiment of the sentence.

This is the instruction that is defining the task, describing the task, instructing the model that what it has to do with the sentence. action-packed thriller and it predicts the sentiment is positive. Again, for machine translation, I'm explicitly mentioning to the language model that translate the sentence to Hindi. It has been raining continuously for the past few days. So what would be the Hindi translation? So like this, now when I'm training the language model, I'm not only supplying the input and output, I'm also supplying and preventing the instruction along with the input.

And the idea is that we want to teach the language model how to understand instructions, because if it can understand instructions, it can generalize across some arbitrary tasks that it has to solve during test time. And since we'll be describing the task, for example, during inference time, when we are now taking the COLA task, we are now describing the new task by an instruction. We are saying, is this sentence grammatical? We are asking the language model and the language model is able to say that it is not grammatical. So here the change is happening that now when I'm giving a new task during the test time, I'm describing the task via a natural language instruction. And the idea is that during training time, since we had supplied the language model with similar examples, which demonstrated that how it is supposed to respond to such instructions, now it won't fumble when it suddenly sees a new task during test time and it would be able to solve it.

So this is one of the goals of instruction tuning, where the idea is that it should be able to generalize to new tasks seen during inference time. And these tasks will be, and all the tasks will be described by a natural language instructions. So with that, we can discuss what is instruction tuning? So to consolidate what all we have spoken till now, so instruction tuning is a fine tuning technique. We fine tune the large language model on instruction data sets. Instruction data sets have tasks and examples, this type of natural language instructions.

So these data sets contains examples that demonstrate to the language model that how it should respond to an instruction. So this boosts their ability to understand, reason, and follow instructions. And when it comes across a new unseen task, it becomes better at handling it. So in this slide also, we have another example. So this is from the paper, Fine-tuned Language Models as Zero-Shot Learners.

Here we can see in the example that during training, we are instruction tuning the language model using a variety of tasks, common sense reasoning, translation, etc, etc. And for each of these tasks, you see, if we take a look at the translation example, we are first giving the instruction, translate the sentence to Spanish. Then we have the corresponding sentence, which would originally be the input to the language model, the sentence we want to translate. The new office building was built in less than three months. And then the target is the translated Spanish sentence.

So in instruction tuning each example is first we have the instruction the input and output, these three components, input is not always mandatory, sometimes some tasks have input, some do not, but instructions will be there and the target output will be there. One thing is that here the tasks present during training and inference at this joint there's no intersection between them. So during inference time we see how it is performing on a new unseen task which is in this case natural language inference. So we have the premise which says that at my age you will probably have learned one lesson and the hypothesis is it's not certain how many lessons you'll learn by your 30s. And the instructions, so this is originally the input, the premise, and the hypothesis.

Now the instruction to the language model is, does the premise entail the hypothesis? So this instruction, we are describing the task we want the language model to perform. And from this, since the language model now has some understanding of what it is supposed to do, how it is supposed to respond to instructions, it responds by saying it is not possible to tell. So we are not sure whether it's an entailment or not. So again, so the key idea over here is through instruction tuning, we want to teach the language model. We're training the language model to execute tasks based on natural language instructions such that it can adapt efficiently to new, previously unseen tasks.

One more point that I would like to mention over here is in this week, prompting will also be covered. There's a distinction between prompting and context learning and instruction tuning. In instruction tuning, we are fine-tuning the language model. Parameters of the weights of the language model are getting updated whereas in prompting or in context learning no such thing happens. We are not updating any parameters of the language model.

So let's discuss about training laws in case of instruction fine tuning so say you're visiting Mumbai and you're curious to know what places you can visit over here so you prompt the language model. Can you recommend some places to visit in Mumbai? And the desired target response for the language model is, so here are some great places you can explore in Mumbai and the list of places. So basically what would happen is that during training, what happens is at each point, the prediction happens one word at a time based on whatever the input or the instruction is fed to the language model. Along with it whatever output has

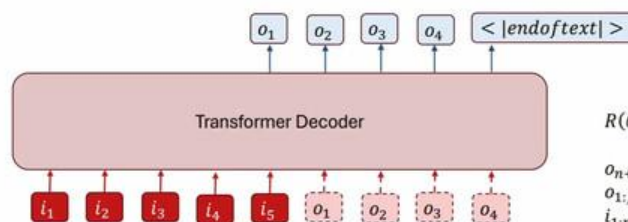
been generated by the language model till this point. For example, initially this is the input that we had passed to the language model, which is an instruction.

For certain tasks, input would also be present. Like the example we saw in the previous slide, the input would also be fed to the language model. Initially, the first step, we are feeding the entire instruction and input to the language model. And what the language model would do is conditioned on this input and instruction, it needs to predict the next word. So it would be predicting the distribution of probability of next word for all the words in the vocabulary.

And we know that this is our target word at this point. So here we can use the cross entropy loss across each of the steps and via that we can train the language model. So this is how it gets updated. Now in the next step, we'll have the instruction input. Along with that, we'll also have this word which was generated by the language model.

And with that, the process goes on until the end of sentence token would be encountered. And with that, the generation is complete. Next is a little bit more explanation about chaining loss for decoder only models and encoder decoder models. So in decoder only models, what happens is we'll be feeding so i_1 to i_5 is the instruction or the input that we are feeding to the language model and o_1 to o_4 are the output that a language model is generating. So here these o_1 o_2 o_3 o_4 which are colored in red.

Training Loss (Decoder-only models)



$$R(\theta) = \sum_{j=0}^n \log p_{\theta}(o_{j+1} | o_{1:j}, i_{1:m})$$

$$o_{n+1} = <|endof\text{text}|>$$

$$o_{1:j} = o_1, \dots, o_j$$

$$i_{1:m} = i_1, \dots, i_m$$

Slide by Gaurav Pandey



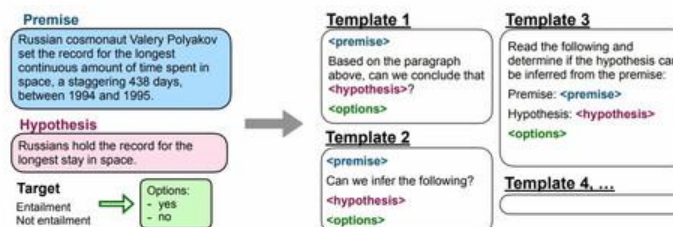
So these are the actual target actual output that we desire. So this is marked in red because here we are doing teacher forcing so we are passing the actual output to the language model and not the one which was generated by the language model. This is a difference. So for decoder models we're feeding the entire thing into the decoder. For encoder decoder models we'll be feeding the instruction and the input to the encoder and the rest of the things and again due to teacher forcing will be again feeding the actual output to the decoder and the output would also be generated from the decoder end.

So this is just there's a bit of there's a difference in architecture between encoder decoder and decoder models where there is only decoder component you'll be feeding everything into the decoder component when there is an encoder and decoder component you feed the input and the instruction to the encoder and you get the output from the decoder. So this is just following the previous lectures. So again, if we just quickly take a look at the formula also, I just described the idea in the previous slide. So here you can see the formulation. So here this is the log likelihood.

So this is the conditional probability of generating this token, given all the previously generated tokens and the input that we have given to the model. And you take a sum of it ideally since this is log likelihood you will be maximizing this. Similar formulation holds for the encoder decoder modules as well. So another interesting thing in instruction tuning is that there's a lot of diversity in how we can phrase the instruction, given a single task. For example, here we have the task of natural language inference.

Multiple instruction templates for a task

Rephrasing the instructions for a task helps the model learn and generalize more effectively.



Paper: Wei, J., Bosma, M., Zhao, V.Y., Guu, K., Yu, A.W., Lester, B., Du, N., Dai, A.M. and Le, Q.V., 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.



For this task, there are several different ways in which we can describe the task, in which we can provide the instruction to the language model. And all these different ways are helpful for instruction tuning. So say for this task, there are four different templates or 10 different templates that I can generate. For all of those templates, if I train the language model on that by rephrasing the instructions, we'll see that the model can learn and generalize more effectively. So if I discuss the example in more details, so here we have the premise, we have the hypothesis.

The target is whether it's an entailment or not. So the options would be yes or no. So in this template, we have a placeholder for premise and hypothesis and the options. Other than that, the text in red is the instruction describing the task in various different ways. So one way to talk about it is based on the paragraph above, which is the premise, can we conclude that, which is the hypothesis.

So this is one way to describe the task. Another way is we give the premise and we ask, can we infer the following? And then we provide the hypothesis. Another way is read the following and determine if the hypothesis can be inferred from the premise. This is very explicitly we are giving and explicitly providing the premise and hypothesis along with the options. So there are so many ways to describe the same task.

And we train the model when we generate the instruction in the fine tuning data set. We use all these ideas and train the model on various versions of the instructions for the same task. So here, if I talk about the data set which is present in this slide, it is from the Scaling Instruction Fine-Tuned Language Models paper, which is a very interesting paper. If anybody is interested, please feel free to read the paper. Here, what they have done is they have created this dataset consisting of 1,836 tasks, which is a huge number of tasks, and they have further fine-tuned a language model on these tasks.

Diverse tasks



- The fine-tuning dataset consists of 473 datasets, 146 task categories, and 1,836 total tasks

Paper: Chung, H.W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S. and Webson, A., 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70), pp.1-53.

Introduction to LLMs

NPTL



LCS

Poulami Ghosh

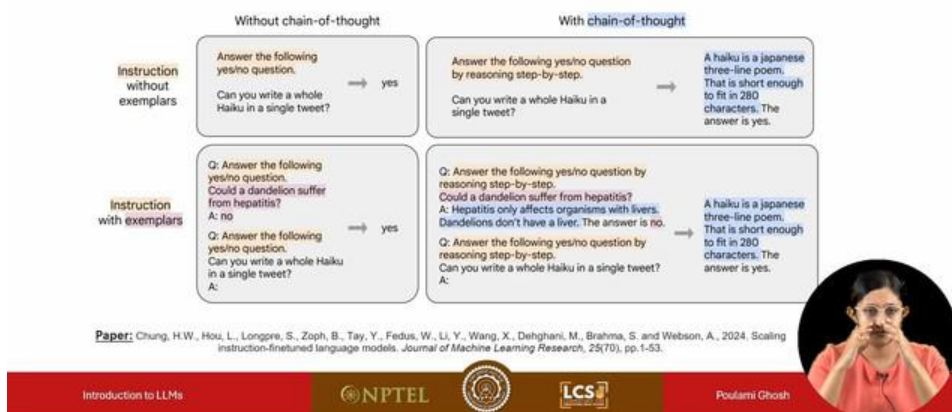


So here's one interesting idea that they have used is, so from a single data set, there are different varieties of tasks that can be generated. For example, if we take the SQuAD data set, SQuAD is originally it's an extractive question answering data set. We have a passage, a paragraph, and there's a question, and there's a specific span in the paragraph which is the answer to that question. So it's an extract of question answering data set. So now other tasks can also be generated from this.

For example, query generation. We already have the paragraph. Maybe we can generate a query from that or context generation that we have the question, can we generate the context for that? So just from this, just from the squad data set, we can have multiple different tasks that can be created from it. And using this idea, they were able to generate this huge number of tasks. Again, like like we saw in the previous slides here, there's a distinction between which tasks appear in the training data set versus the ones in the held out task.

So they are disjoint. There's no common tasks which appear both in training as well as held out set. So the authors of this paper they also tried various data formats when fine tuning. So one way is your instruction tuning, so one way is when your instruction tuning you provide no examples, just you ask that you have a question which is can you write a whole haiku in a single tweet and you simply ask the question answer the following yes or no question. So that's it.

Fine-tuning data formats



All you can do is you can give examples you have another question that you provide as an example to the language model and then you ask the intended question. Another thing that you can try is using chain of thought prompting. So you can ask the language model to reason step by step. And here also there are two variations. You can give examples.

You can simply ask the question without any examples. So there are like four different like variations in the data format. So this is also like can be useful. And this technique was also discussed in the paper. If you look at the results, First, let's start from the topmost results.

Here we can see that if we increase the model size and as well as if we increase the number of fine-tuning tasks, we see that the performance of the language model increases, improves. So if we look in the first graph, here each of the lines indicate a fixed number of tasks. And we see as the number of tasks are increasing, the lines are also ascending in the graph. So here, this is also indicative that with a higher number of tasks, the performance increases. Also now, along the x-axis, we see the number of parameters of the model.

So here also, if we increase the parameters of the model, we also see that there's this increasing trend. So the performance is increasing with a higher number of model parameters, higher model size, better performance. In the second graph, again, here we see along the x-axis now we have the number of fine-tuning tasks. And here we have three models. We have an 8 billion model, we have 62 billion, and we have 540 billion parameter models.

And we see that if we increase the model size, again, we see that the graphs are above the previous graphs. So there's an increase in performance. And also as we're increasing the number of tasks, the performance is increasing. So overall, the key takeaway is length. When you increase the model size, when you increase the number of fine tuning parameters, the performance increases.

So till now we are only focusing on the improvement in performance. Does this come at a cost? Do we have to bear a lot of compute or not? So here we compare the amount of compute that was required for pre-training versus that which was required for fine-tuning, we see that it's simply like 0.2%, 0.6%, 1.6%, which is very, very less compared to, so the compute for fine-tuning is very, very less compared to what was required for the pre-training.

And so here, the authors have also tried to experiment across different varieties of tasks. So tasks with different architectures, encoder-decoder model, decoder-only model, and also with different pre-training objectives. So we have span corruption, which is the pre-training objective for T5 models. For PARM models, which is a decoder-only model, their causal language modeling is a pre-training objective. So like that across different varieties of language models, the authors have tried to see if the results are generalizable.

So if you look at this big table over here, now, if you look at the results, the difference in results between the base model and the fine-tuned version of the model, we see that there is an increment in performance, which is written in this blue numbers we can see. So all these blue numbers indicate that after fine-tuning the language model, we observe that the performance of the language model has become better. So the previous approach that we spoke about, there the instruction tuning data set was mainly generated by, mainly consisted of human written instructions. So that becomes very tedious when there's a human in the loop, like creating and writing instructions, it becomes very tedious and it becomes difficult to scale it beyond a certain point. So for that, there has been research which has explored automated ways of generating instruction tuning data sets.

And one important work in this area is that of self-instruct. So in this slide, we have the overview of the technique that they follow. So they start with a number of C tasks. So there are 175 C tasks. They have one instruction and one instance per task.

This is human written. Initiate. Then there's a task pool where they sample a number of instructions and they pass it to their language model, which in this case is GPT-3. And from that, they generate instructions. And there's a few details that we'll cover in the next few slides.

But they clear the instruction. That is one important step. Once you have the instruction, So the LLM creates the instruction, and once you have the instruction, you make the LLM itself generate instances like input output pairs, and then there's a filtering step involved, and if it's a valid, reasonable instruction input output example that was generated, then it is added to the task pool, and like that, the cycle goes on. So one of the steps that we saw was instruction generation. So for instruction generation, what is done is...

Prompt Templates

Come up with a series of tasks:

Task 1: {instruction for existing task 1}

Task 2: {instruction for existing task 2}

Task 3: {instruction for existing task 3}

Task 4: {instruction for existing task 4}

Task 5: {instruction for existing task 5}

Task 6: {instruction for existing task 6}

Task 7: {instruction for existing task 7}

Task 8: {instruction for existing task 8}

Task 9:

New instruction generation

Paper: Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khazanchi, and Hannaneh Hajishirz. 2023. [Self-Instruct: Aligning Language Models with Self-Generated Instructions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Can the following task be regarded as a classification task with finite output labels?

Task: Given my personality and the job, tell me if I would be suitable.
Is it classification? Yes

Task: Give me an example of a time when you had to use your sense of humor.
Is it classification? No

Task: Replace the placeholders in the given text with appropriate named entities.
Is it classification? No

Task: Fact checking - tell me if the statement is true, false, or unknown, based on your knowledge and common sense.
Is it classification? Yes

Task: Return the IMDb number for the person.
Is it classification? No

Task: Detect if the Reddit thread contains hate speech.
Is it classification? Yes

Task: Analyse the sentences below to identify biases.
Is it classification? No


...

Task: To make the pairs have the same analogy, write the fourth word.
Is it classification? No

Task: Given a set of numbers, find all possible subsets that sum to a given number.
Is it classification? No

Task: {instruction for the target task}

Classify a task instruction is a classification



Introduction to LLMs

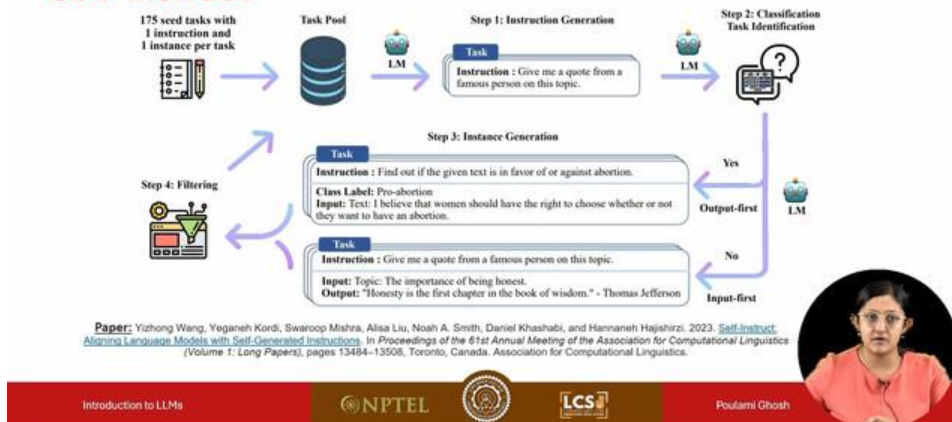
NPTEL

LCS

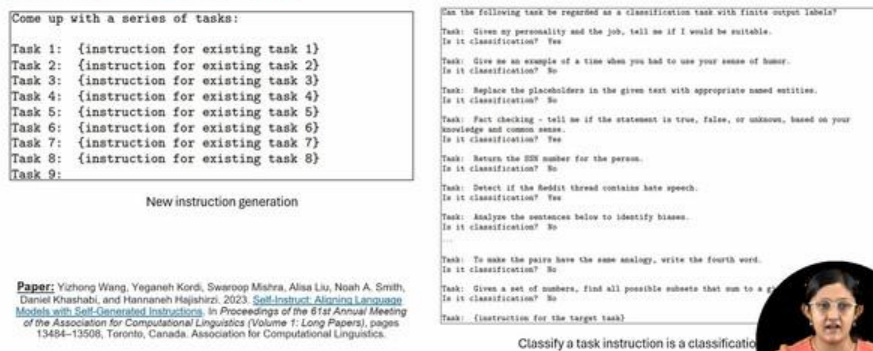
Poulami Ghosh

What is done is from the initial pool, so we have 175 seed tasks. From the seed tasks, eight tasks are sampled. And in this age task, six tasks are human written and two are generated by the language model. And like that, we pass instructions for those existing tasks. So these would serve as examples.

Self-Instruct



Prompt Templates



And based on this, we want the language model to generate new instructions. So these are the eight tasks which were randomly sampled from the pool of seed tasks. And with this, new instructions will be generated. Now once an instruction is generated, next you want to identify whether the task that corresponds to the instruction, is that a classification task or non-classification task? I'll mention the reason in some time for this distinction. So how do we do that? We prompt the language model again.

So here we are doing few-shot prompting. We prompt the language model with examples that given this is the task instruction, we need to, is this a classification task? Yes or no. So

for example, you can describe a task with the instruction, given my personality and the job, tell me if I would be suitable. So the answer of this is yes, you can be suitable, you cannot be suitable. So there's a finite output space and this is indeed a classification task.

Another example is, give me an example of a time when you had to use your sense of humor. So this is not a classification task. This is a quite open-ended task. So this is not a classification task. So we are giving the language model examples, which illustrate that given an instruction, the language model needs to predict whether the task is classification or not.

And then finally, given all these examples, we pass to the language model the instruction for the target task. It predicts whether it is or not. Till now, in step one, we created the instructions. Step two, we found out whether the task is a classification task or not. Now, the next step is for classification tasks, there's a separate way in which we'll be generating the input and output.

Prompt Templates

```

Come up with examples for the following tasks. Try to generate multiple examples when possible.
If the task doesn't require additional input, you can generate the output directly.

Task: Which exercises are best for reducing belly fat at home?
Output:
- Lying Leg Raises
- Leg In And Out
- Plank
- Side Plank
- Sit-ups

Task: Converting 85 F to Celsius.
Output: 85°F = 29.44°C

Task: Sort the given list accordingly.
Example 1
List: [10, 92, 2, 5, -4, 92, 8, 101]
Output: [-4, 2, 5, 8, 10, 92, 99, 101]
Example 2
Input 2 - List: [9, 99, 10, -5, -1000, 567, 999]
Output: [-1000, -5, 9, 99, 10, 999, 567]
...

Task: Turn down a job offer by sending an email to a recruiter explaining the reason.
Output: Hi [Recruiter],
Thank you so much for the generous offer to join your team. As we discussed, I've admired the
company for a number of years, and as a proud admirer of its products. However, after further
consideration of where I currently am in my career, I've decided to accept an offer at another
company.
I would love to stay in touch with you and have already started following you on [Social Media
Platform]. Again, thank you so much for your time and consideration.
Thanks again,
[Your Name]

Task: (Instruction for the target task)

```

Input-first approach of instance generation

```

Given the classification task definition and the class labels, generate an input that
corresponds to each of the class labels. If the task doesn't require input, just generate the
correct class label.

Task: Classify the sentiment of the sentence into positive, negative, or mixed.
Class label: mixed
Sentence: I enjoy the flavor of the restaurant but their service is too slow.
Class label: Positive
Sentence: I had a great day today. The weather was beautiful and I spent time with friends.
Class label: Negative
Sentence: I was really disappointed by the latest superhero movie. I would not recommend it.
Task: Tell me if the following email is a promotion email or not.
Class label: Promotion
Email: Check out our amazing new sale! We've got discounts on all of your favorite products.
Class label: Not Promotion
Email: We hope you are doing well. Let us know if you need my help.
Task: Detect if the Reddit thread contains hate speech.
Class label: Hate Speech
Thread: All people of color are stupid and should not be allowed to vote.
Class label: Not Hate Speech
Thread: The best way to cook a steak on the grill.
...

Task: Which of the following is not an input type? (a) number (b) date (c) phone number (d)
email address (e) all of these are valid inputs.
Class label: (a)

Task: (Instruction for the target task)

```

Output-first approach of instance generation

Paper: Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khoshnabi, and Hannaneh Hajishirzi. 2023. [Self-Instruct: Aligning Large Language Models with Human Instructions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13458–13470. Association for Computational Linguistics.

Introduction to LLMs

NPTEL



LCS

Poulami Ghosh



And for non-classification tasks, we'll be doing it in a separate way. This is because the authors of the paper saw that for classification tasks, since there is a finite output space, the language model gives importance to the more popular output labels. As a result, in the final dataset, the representation of examples across all the labels is not even. So we want like more or less equal number of examples across all the class labels that are present in the output space for the classification task. And for that, what the authors do is they propose

this approach called output first approach for instance generation in case of classification task.

Here they give the classification task definition, they give the class labels, and based on that, they ask the language model to generate an input that corresponds to the class level. For example, they're giving the task label, which is the output label, and then they're asking the language model to generate the input or the sentence. Like that, finally, for the target task, the language model needs to perform this similar response, the language model needs to generate. These are all examples of few-shot prompting where we are showing the language model. This is the behavior I want the language model to follow.

We're first showing the language model via the examples. Again, now if you talk about non-classification tasks, there they're following the much more intuitive way that you give the task, then you ask the language model to first generate the input and then the output. So this is simple. And then there's also filtering mechanism where we see that the new instruction that is generated should be different from the instructions that are already present in the task. So they have used Rouge-L similarity score and they've seen that Rouge-L similarity score between the new instruction and the existing instruction should be less than 0.7,

Filtering

- A new instruction is included in the task pool only if its ROUGE-L similarity score with any existing instruction is less than 0.7.
- Exclude instructions that have certain keywords such as image, picture, graph
- Remove duplicate instances or ones with the same input but different outputs.
- Filtering based on heuristics (e.g., length of the instruction).



only then the instruction will be included in the task. Second is keywords such as image, picture, graph. Instructions which contain such keywords will be excluded because the

language model do not have the capacity to handle such inputs. Another is, now we have generated an instruction at the time of instance generation, say, we come across duplicate instances. We remove them.

Or for example, we come across instances where the input is same and the output is different. These are also removed. And finally there's a heuristic based filtering where say so there's a constraint on the length of the instruction so the instruction should not be too long or too short and based on this also filtering happens. So these are some examples of like data that was generated using this mechanism. So you can look at the last example, so here the instruction that was generated by the language model is write a story with three characters a person an animal and an object.

Then again the input was also generated by the language model. So here we have a person named john who's a programmer, an animal, a dog, an object, a laptop and based on that a story was again generated by the language model. So this is also like quite reasonable valid data instance that was generated by the language model in an automatic way without any human in the loop So there are other examples. You can pause the video and take a look at them. Nice survey paper, Instruction Tuning for Large Language Models.

I would encourage the students to read this paper if they're interested. And here in this slide, all the lists of various instruction tuning data sets are present. So in today's lecture, we discussed about instruction tuning, which is a technique via which we teach the language models to follow instructions. There are other techniques such as reinforcement learning from human feedback which is our nature which further improves upon instruction tuning and it helps the language model to better align with human objectives and preferences which is so this topic would also be taken up in the upcoming weeks along with that another point is when we spoke about instruction fine tuning we were mainly talking about that we are fully fine tuning the language model. So all the entire model is getting updated.

So this is a compute intensive, takes up a lot of memory. So there are much more efficient strategies present that is best parameter efficient fine tuning. So those strategies will also

be covered in the upcoming lectures. So that's it from my end for today. Thank you for listening and happy learning.