GWENDOLYN STRIPLING: Hello.
And welcome to Introduction to Generative AI.
My name is Dr. Gwendolyn Stripling.
And I am the artificial intelligence
technical curriculum developer here at Google Cloud.
In this course, you learn to define generative AI,
explain how generative AI works, describe generative AI model
types, and describe generative AI applications.
Generative AI is a type of artificial intelligence
technology that can produce various types of content,
including text, imagery, audio, and synthetic data.
But what is artificial intelligence?
Well, since we are going to explore
generative artificial intelligence,
let's provide a bit of context.
So two very common questions asked
are what is artificial intelligence
and what is the difference between AI and machine
learning.
One way to think about it is that AI is a discipline,
like physics for example.
AI is a branch of computer science
that deals with the creation of intelligence agents, which
are systems that can reason, and learn, and act autonomously.
Essentially, AI has to do with the theory and methods
to build machines that think and act like humans.
In this discipline, we have machine learning,
which is a subfield of AI.
It is a program or system that trains a model from input data.
That trained model can make useful predictions
from new or never before seen data
drawn from the same one used to train the model.
Machine learning gives the computer
the ability to learn without explicit programming.
Two of the most common classes of machine learning models

are unsupervised and supervised ML models.
The key difference between the two
is that, with supervised models, we have labels.
Labeled data is data that comes with a tag like a name, a type,
or a number.
Unlabeled data is data that comes with no tag.
This graph is an example of the problem
that a supervised model might try to solve.
For example, let's say you are the owner of a restaurant.
You have historical data of the bill amount
and how much different people tipped based on order type
and whether it was picked up or delivered.
In supervised learning, the model learns from past examples
to predict future values, in this case tips.
So here the model uses the total bill amount
to predict the future tip amount based on whether an order was
picked up or delivered.
This is an example of the problem
that an unsupervised model might try to solve.
So here you want to look at tenure and income
and then group or cluster employees
to see whether someone is on the fast track.
Unsupervised problems are all about discovery,
about looking at the raw data and seeing if it naturally
falls into groups.
Let's get a little deeper and show this graphically
as understanding these concepts are
the foundation for your understanding of generative AI.
In supervised learning, testing data values or x
are input into the model.
The model outputs a prediction and compares that prediction
to the training data used to train the model.
If the predicted test data values and actual training data
values are far apart, that's called error.
And the model tries to reduce this error

until the predicted and actual values are closer together.
This is a classic optimization problem.
Now that we've explored the difference
between artificial intelligence and machine learning,
and supervised and unsupervised learning,
let's briefly explore where deep learning
fits as a subset of machine learning methods.
While machine learning is a broad field that
encompasses many different techniques,
deep learning is a type of machine learning
that uses artificial neural networks,
allowing them to process more complex patterns than machine learning.
Artificial neural networks are inspired by the human brain.
They are made up of many interconnected nodes or neurons
that can learn to perform tasks by processing data and making predictions.
Deep learning models typically have many layers
of neurons, which allows them to learn
more complex patterns than traditional machine learning models.
And neural networks can use both labeled and unlabeled data.
This is called semi-supervised learning.
In semi-supervised learning, a neural network
is trained on a small amount of labeled data
and a large amount of unlabeled data.
The labeled data helps the neural network
to learn the basic concepts of the task
while the unlabeled data helps the neural network
to generalize to new examples.
Now we finally get to where generative AI
fits into this AI discipline.
Gen AI is a subset of deep learning, which
means it uses artificial neural networks,
can process both labeled and unlabeled data using

supervised, unsupervised, and semi-supervised methods.
Large language models are also a subset of deep learning.
Deep learning models, or machine learning models in general,
can be divided into two types, generative and discriminative.
A discriminative model is a type of model
that is used to classify or predict labels for data points.
Discriminative models are typically
trained on a data set of labeled data points.
And they learn the relationship between the features
of the data points and the labels.
Once a discriminative model is trained,
it can be used to predict the label for new data points.
A generative model generates new data instances
based on a learned probability distribution of existing data.
Thus generative models generate new content.
Take this example here.
The discriminative model learns the conditional probability
distribution or the probability of y,
our output, given x, our input, that this is a dog
and classifies it as a dog and not a cat.
The generative model learns the joint probability distribution
or the probability of x and y and predicts
the conditional probability that this is a dog
and can then generate a picture of a dog.
So to summarize, generative models
can generate new data instances while discriminative models
discriminate between different kinds of data instances.
The top image shows a traditional machine
learning model which attempts to learn
the relationship between the data and the label,
or what you want to predict.
The bottom image shows a generative AI model
which attempts to learn patterns on content so that it
can generate new content.
A good way to distinguish what is gen AI and what is not

is shown in this illustration.
It is not gen AI when the output, or y, or label is
a number or a class, for example spam or not spam,
or a probability.
It is gen AI when the output is natural language, like speech
or text, an image or audio, for example.
Visualizing this mathematically would look like this.
If you haven't seen this for a while,
the y is equal to f of x equation calculates
the dependent output of a process given different inputs.
The y stands for the model output.
The f embodies the function used in the calculation.
And the x represents the input or inputs used for the formula.
So the model output is a function of all the inputs.
If the y is the number, like predicted sales,
it is not gen AI.
If y is a sentence, like define sales,
it is generative as the question would elicit a text response.
The response would be based on all the massive large data
the model was already trained on.
To summarize at a high level, the traditional, classical
supervised and unsupervised learning process
takes training code and label data to build a model.
Depending on the use case or problem,
the model can give you a prediction.
It can classify something or cluster something.
We use this example to show you how much more robust
the gen AI process is.
The gen AI process can take training code, label data,
and unlabeled data of all data types
and build a foundation model.
The foundation model can then generate new content.
For example, text, code, images, audio, video, et cetera.
We've come a long away from traditional programming
to neural networks to generative models.

In traditional programming, we used
to have to hard code the rules for distinguishing a cat--
the type, animal; legs, four; ears, two; fur, yes;
likes yarn and catnip.
In the wave of neural networks, we
could give the network pictures of cats and dogs
and ask is this a cat and it would predict a cat.
In the generative wave, we as users
can generate our own content, whether it
be text, images, audio, video, et cetera, for example
models like PaLM or Pathways Language Model,
or LaMDA, Language Model for Dialogue Applications,
ingest very, very large data from the multiple sources
across the internet and build foundation language
models we can use simply by asking a question,
whether typing it into a prompt or verbally
talking into the prompt itself.
So when you ask it what's a cat, it
can give you everything it has learned about a cat.
Now we come to our formal definition.
What is generative AI?
Gen AI is a type of artificial intelligence
that creates new content based on what it has
learned from existing content.
The process of learning from existing content
is called training and results in the creation
of a statistical model when given a prompt.
AI uses the model to predict what an expected response might
be and this generates new content.
Essentially, it learns the underlying structure
of the data and can then generate
new samples that are similar to the data it was trained on.
As previously mentioned, a generative language model
can take what it has learned from the examples it's
been shown and create something entirely new

based on that information.
Large language models are one type of generative AI
since they generate novel combinations of text
in the form of natural sounding language.
A generative image model takes an image
as input and can output text, another image, or video.
For example, under the output text,
you can get visual question answering
while under output image, an image completion is generated.
And under output video, animation is generated.
A generative language model takes text as input
and can output more text, an image, audio, or decisions.
For example, under the output text,
question answering is generated.
And under output image, a video is generated.
We've stated that generative language models learn
about patterns and language through training data,
then, given some text, they predict what comes next.
Thus generative language models are pattern matching systems.
They learn about patterns based on the data you provide.
Here is an example.
Based on things it's learned from its training data,
it offers predictions of how to complete this sentence,
I'm making a sandwich with peanut butter and jelly.
Here is the same example using Bard,
which is trained on a massive amount of text data
and is able to communicate and generate
humanlike text in response to a wide range of prompts
and questions.
Here is another example.
The meaning of life is--
and Bart gives you a contextual answer
and then shows the highest probability response.
The power of generative AI comes from the use of transformers.
Transformers produced a 2018 revolution

in natural language processing.
At a high level, a transformer model
consists of an encoder and decoder.
The encoder encodes the input sequence
and passes it to the decoder, which
learns how to decode the representation
for a relevant task.
In transformers, hallucinations are words or phrases
that are generated by the model that
are often nonsensical or grammatically incorrect.
Hallucinations can be caused by a number of factors,
including the model is not trained on enough data,
or the model is trained on noisy or dirty data,
or the model is not given enough context,
or the model is not given enough constraints.
Hallucinations can be a problem for transformers
because they can make the output text difficult to understand.
They can also make the model more
likely to generate incorrect or misleading information.
A prompt is a short piece of text
that is given to the large language model as input.
And it can be used to control the output of the model
in a variety of ways.
Prompt design is the process of creating
a prompt that will generate the desired output
from a large language model.
As previously mentioned, gen AI depends a lot
on the training data that you have fed into it.
And it analyzes the patterns and structures of the input data
and thus learns.
But with access to a browser based prompt, you, the user,
can generate your own content.
We've shown illustrations of the types of input based upon data.
Here are the associated model types.
Text-to-text.

Text-to-text models take a natural language input
and produces a text output.
These models are trained to learn the mapping
between a pair of text, e.g.
for example, translation from one language to another.
Text-to-image.
Text-to-image models are trained on a large set of images,
each captioned with a short text description.
Diffusion is one method used to achieve this.
Text-to-video and text-to-3D.
Text-to-video models aim to generate a video representation
from text input.
The input text can be anything from a single sentence
to a full script.
And the output is a video that corresponds to the input text.
Similarly, text-to-3D models generate
three dimensional objects that correspond to a user's text
description.
For example, this can be used in games or other 3D worlds.
Text-to-task.
Text-to-task models are trained to perform a defined task
or action based on text input.
This task can be a wide range of actions
such as answering a question, performing a search,
making a prediction, or taking some sort of action.
For example, a text-to-task model
could be trained to navigate a web UI or make changes to a doc
through the GUI.
A foundation model is a large AI model pre-trained
on a vast quantity of data designed to be adapted or fine
tuned to a wide range of downstream tasks,
such as sentiment analysis, image captioning, and object
recognition.
Foundation models have the potential
to revolutionize many industries, including

health care, finance, and customer service.
They can be used to detect fraud and provide
personalized customer support.
Vertex AI offers a model garden that
includes foundation models.
The language foundation models include
PaLM API for chat and text.
The vision foundation models includes stable diffusion,
which has been shown to be effective at generating
high quality images from text descriptions.
Let's say you have a use case where
you need to gather sentiments about how your customers are
feeling about your product or service.
You can use the classification task sentiment analysis task
model for just that purpose.
And what if you needed to perform occupancy analytics?
There is a task model for your use case.
Shown here are gen AI applications.
Let's look at an example of code generation
shown in the second block under code at the top.
In this example, I've input a code file conversion problem,
converting from Python to JSON.
I use Bard.
And I insert into the prompt box the following.
I have a Pandas DataFrame with two columns, one with the file
name and one with the hour in which it is generated.
I'm trying to convert this into a JSON file
in the format shown onscreen.
Bard returns the steps I need to do this and the code snippet.
And here my output is in a JSON format.
It gets better.
I happen to be using Google's free, browser-based Jupyter
Notebook, known as Colab.
And I simply export the Python code to Google's Colab.
To summarize, Bart code generation

can help you debug your lines of source code,
explain your code to you line by line,
craft SQL queries for your database,
translate code from one language to another,
and generate documentation and tutorials for source code.
Generative AI Studio lets you quickly explore and customize
gen AI models that you can leverage in your applications
on Google Cloud.
Generative AI Studio helps developers create and deploy
Gen AI models by providing a variety of tools and resources
that make it easy to get started.
For example, there's a library of pre-trained models.
There is a tool for fine tuning models.
There is a tool for deploying models to production.
And there is a community forum for developers
to share ideas and collaborate.
Generative AI App Builder lets you
create gen AI apps without having to write any code.
Gen AI App Builder has a drag and drop interface
that makes it easy to design and build apps.
It has a visual editor that makes
it easy to create and edit app content.
It has a built-in search engine that
allows users to search for information within the app.
And it has a conversational AI Engine
that helps users to interact with the app using
natural language.
You can create your own digital assistants, custom search
engines, knowledge bases, training applications,
and much more.
PaLM API lets you test and experiment
with Google's large language models and gen AI tools.
To make prototyping quick and more accessible,
developers can integrate PaLM API with Maker suite
and use it to access the API using a graphical user

interface.
The suite includes a number of different tools such as a model training tool, a model deployment tool, and a model monitoring tool.
The model training tool helps developers train ML models on their data using different algorithms.
The model deployment tool helps developers deploy ML models to production with a number of different deployment options.
The model monitoring tool helps developers monitor the performance of their ML models in production using a dashboard and a number of different metrics.
Thank you for watching our course, Introduction to Generative AI.

Welcome to my Primer on Generative AI. I'm Jules White.
I'm a professor in computer Science at Vanderbilt University, and I'm also the Director of the Vanderbilt Initiative on the Future of Learning and Generative AI.
This is our initiative at Vanderbilt to help make sure that Vanderbilt is a world leader and innovator in the use of generative AI, and so I'm going to talk to you about how we approach generative AI within Vanderbilt and how you should be thinking about generative AI and what you should know about it, what you should think about in terms of training your workforce and also in terms of the capabilities and why it's so important.

Now this is a technology that in
October of 2022, if you had asked me,
is something like ChatGPT
going to exist in your lifetime,
I would have told you flat out,
no way, never going to happen, don't believe it.
Never, and boy, was I wrong,
and what I want to show you is the depth of
capability that's there and why this is exciting to me.
Now as a computer scientist.
I often feel I'm looking up into a telescope saying,
I promise in five years
we'll make it to that planet over there,
and then one day I was looking at my telescope and I
looked over my shoulder and a UFO had landed behind me,
and that's what ChatGPT is.
It's like this incredible new alien technology
that now as we're kicking it,
it's beaming trees around the Earth and solving
all incredible problems that I
never thought a computer would be able to solve,
and so what I'm excited to do is to show
you and help you explore all of
these capabilities and understand it
in a different way from what
you may have seen in the news.
Now one of the things I want to take
a moment to mention is I teach
a course on Prompt Engineering for ChatGPT on Coursera,
and there's over 150,000 people
that at the time of this filming have taken the course.
A lot of what you're going to see is their ideas,
their innovation that has filtered back to me,
and I'm going to be talking about what I'm seeing them
doing and how they're innovating out in the workplace.

I want to give a special thank you to everybody
who's taken that class and
all of the feedback that I've received.
Now whenever we start talking about generative AI,
we need to start putting
a mental model in place of what it really is.
We hear AI used all over the place,
generative AI is a type of AI.
Really a lot of the excitement that
we're seeing is about generative AI.
I want to encourage you when you start talking about AI,
instead talk about generative AI because
generative AI is one of
the most exciting things that I've ever seen,
and that's what's really
creating a lot of the excitement.
Now when we start talking about generative I,
a lot of people really don't have a
mental model of what it is,
how it works, what to do with it.
I want to give you that mental model.
Imagine text messaging. We all know how it works.
You go and type in a message,
and you send it off,
and another human being, probably your friend,
is going to type in a response to you and send it back,
and generative AI works really similar to
text messaging or at least
a lot of the interfaces that we use for it.
You go in and you type in a message,
you send it off, but
generative AI rather than your friend,
texts you back or you
send a message off and
generative AI rather than your friend,

creates a photo and sends it back to you.
Or creates a video or creates a recording.
It's like text messaging,
except that we're not
talking to another human being anymore.
We're talking to generative AI.
Now, whenever we start talking about generative AI,
it's important to have some basic terminology,
and one of the really important terms
that you need to know is a prompt.
A prompt is the message that
you type into that generative AI,
text messaging or chat application.
It's the message that you're
sending off to the AI that it's
going to respond to and
when it sends you a message back,
that message that it's sending you back is the output.
Now we see a bunch of
examples of potential prompts on the screen here,
like write a poem about the French Revolution,
and if you've got the right type of friend,
they could write a point about the French Revolution.
Now, these prompts aren't particularly good prompts,
and we'll talk about that more later.
But these are all things that you could
go and type in and send off.
Now, whenever I start looking in the news about
generative AI recently and since it's come out,
I feel really disheartened
because it's really missing the point,
and I want to encourage you to rethink
what is possible with generative AI
and how it's going to impact us.
Now, when I look at the news,

what I see is,
I asked ChatGPT this question
that was potentially bad or worded poorly,
and I got this bad answer out, and therefore,
ChatGPT is useless,
and that's completely the wrong way
of thinking about generative AI and these tools.
You want to think about iterating and
refining and creating content through a conversation,
and you want to think about
not going and asking one question,
getting an answer and throwing
it away if it's not perfect.
Instead you want to think about getting
it back and then providing
feedback on what could be
improved and continuing to refine it.
Just like if you had an assistant
and you brought them to
work, and on their first day of work,
they handed in some work to you, and it wasn't perfect,
you're not going to fire them and tell them it's terrible
and say that person's an idiot and get rid of them.
No, you're going to work with them and
train them and help explain more
about what your task was because you
could have communicated with them wrong too.
Let's take a look at what it means to
refine content through a conversation.
I'm going to give ChatGPT
a task that I don't think I could find somebody
here in Nashville and maybe in
the United States that could do it as well or as fast.
I'm going to start off with,
let's create a meal plan. We all have to eat.

Please create a meal plan for
my family that is based on a fusion
of food from Ethiopia and Uzbekistan.
I want to eat keto and up to 2,000 calories per day,
pick dishes where the ingredients are
easy to get from an average US grocery store.
That's a really hard task.
I don't know who I would find to do that task for me.
It's got a lot of complexities to it,
like average US grocery store ingredients,
what does that mean?
Who has the knowledge of that?
Trying to find somebody to accomplish
this task would be really hard.
What does ChatGPT say?
Says, here's a sample meal plan that
combines flavors from Ethiopia and Uzbekistan,
also being keto friendly,
and within a 2,000 calorie per day limit,
and that's nearly instantaneous.
Then it says, breakfast,
scrambled egg with sauteed onions,
tomatoes, and Ethiopian berbere spice mix.
Then notice what it does.
In parentheses, it says made with chili powder,
paprika, garlic powder,
ginger, cumin, and coriander.
It's signaling to us these are ingredients
you can get from an average US grocery store.
This isn't something that you can't make.
It's something that is makeable with
average US grocery store ingredients and these all are.
Now this is a draft.
It's creating draft input,
draft output force that we can

go and work with and refine.
Now, if this was in the news,
the next thing I would see would be,
this will never replace a nutritionist.
It's a terrible tool.
You said 2,000 calories per day,
it didn't even give you serving sizes.
It's completely useless.
That's completely the wrong way
of thinking about these tools.
Now the analogy I give for this is
imagine if somebody handed Michael Angelo a new hammer,
and Michael Angelo walks up to the piece of stone
and he whacks the stone and then
he looks at the stone and he says it doesn't look like
the Pita or some other beautiful sculpture
and he throws the hammer down in disgust.
He says, it's a terrible tool,
I'll never use it again.
That's what's happening every day when we go and
look at the news and they have these ridiculous articles.
No. What does Michael Angelo do?
He takes the hammer and he
hits the stone and he hits it again and he
hits it again and he continues to refine and iterate on
the stone with the tool until he gets a beautiful output.
He also understands that
his skill with the tool is equally,
if not, more important.
He has to know how to use the tool,
he has to refine his skill with the tool,
and he has to educate himself.
He doesn't just go and say, well,
the tool didn't get it right on the first time,
so the tool is stupid and I'm never going to use it.

No, that's not what he does.
He iterates and refines, and works with it,
and he also understands that
he's an equal participant in this.
Let's say we really care about the serving sizes.
Well, what do we do? We just follow up.
We ask in the same conversation,
can you give me an approximate serving size for myself
for each dish that is within
my 2,000 calorie per day limit?
What does ChatGPT say? It says sure.
Here are approximate serving
sizes for each dish based
on a 2,000 calorie per day limit.
Now it says breakfast, two eggs,
1/2 cup of onions,
1/2 a cup of tomatoes, one teaspoon of spice mix.
Now, I know if it was in the news,
the next thing would be, you said keto,
and that thing says 1/2 a cup of onions.
The macronutrients are all wrong on that,
way too many carbs.
If we really care about
carbohydrates or macronutrients or whatever it is,
we can go and keep refining and shaping the content with
follow-up questions to try
to explore it and make it better.
But I'm going to take it in a different direction
to help you understand
the capability and depth of possibilities with this tool.
That is because macronutrients aren't
my real problem with this meal plan.
I'm going to have macronutrient variation
all the time just because I don't have enough willpower.
My real challenge with this meal plan is that

I have a nine-year-old and he's an adventurous eater,
but the real challenge is getting him to
try the dish in the first place.
I thought about it. Can I use ChatGPT
to help me solve this problem?
I said my son is nine,
sometimes he won't try new dishes.
To make this culinary adventure more fun for him,
can you create a short Pokemon
battle story to go with each dish?
I'll read the stories with him before
dinner to get him excited about trying the new food.
Make sure the story ends with
a cliff-hanger that will motivate him to try the food.
That's a really complex task.
Good luck finding somebody that
can go and generate that meal plan
and then write convincing
Pokemon battle stories with
cliff-hangers for a nine year old.
What does ChatGPT say?
Certainly, story.
Pikachu and his friends were exploring the wilds of
Ethiopia when they were suddenly
ambushed by a group of sneaky Pokemon thieves.
As they raced through the dense jungle,
they stumbled upon a hidden cave filled with treasure.
Then it goes on to say, in the end,
they emerged victorious but not before the dragon left
a fiery spice and imbued
their breakfast eggs with a flavorful kick.
Can you help Pikachu and
his friends defeat the Berbere Dragon and
enjoy this spicy breakfast? That's incredible.
Now, before November of last year,

we didn't have any computing capabilities
that were generally available that somebody
could go in and just dump in and
create Pokemon battle stories to motivate a
nine-year-old to eat a bunch
of dishes that we just dreamed up.
Nothing that we had could even come close to that.
If I had had to go as a software
engineer and tried to program it,
I couldn't have done it.
In a paragraph of text,
we've gone from this meal plan,
we refined it, we've added serving sizes,
and now we've got battle stories to go with it.
What else can we do with ChatGPT and generative AI?
I thought about what we do
with my son at the dinner table,
and we love to talk with him
about what he's learning at school.
He loves math and he
loves to play math games on the iPad.
I thought, wouldn't it be cool if I could get
ChatGPT to play a math game with him?
In that way, when we're sitting at the table,
he can read the Pokemon battle story,
and then he can play this cool math game and it'll
make this really exciting experience for him.
I said, I'd like you to play a math game with my
nine-year-old to teach him about
division with fractions and nutrition.
Ask him questions one at a time involving
Pokemon and these topics and make it into a game.
When he wins the Pokemon win.
Ask the first question.
This is beginning to use a technique called prompt

engineering to make all of this work seamlessly.
Now the key thing of this
is I'm just having a conversation and this is
just a follow up message in the conversation that's
refining it and taking it in a different direction.
What does ChatGPT say?
Great, let's play a fun math game
involving Pokemon and nutrition.
Question 1: Pikachu and his friends
are planning a meal that includes
half a cup of cooked rice,
but they need to divide it evenly among three Pokemon.
How much rice will each Pokemon get?
What does it do? What does ChatGPT do?
It waits.
It waits for him to answer because it knows that it's
playing a game and that it needs
to wait for him to provide input.
Now we normally think about asking
questions to generative AI or telling it what to do.
But it can also ask us
questions and solicit information from
us and wait for us to give
that information back to it and then respond.
Also understands the concept of a game and turn
taking in the rules of the game
and what it needs to do at this point in time.
That's really deep reasoning.
Then what happens when he goes in and
types in the correct answer, which is 1/6?
It says, good job, that's correct.
It works out the math from for him.
Then it says, Question 2,
Ash wants to make a smoothie
and then goes on to the next question.

It would keep going and asking
him questions over and over and over
again on this topic
and create a whole game
that he could go and interact with.
Notice, all we did to create this was we
basically gave it some paragraphs
of text in a conversation.
We continued to refine
this idea and follow up on it and evolve it.
But I thought as a computer scientist,
it's great that he could play a game in ChatGPT,
but what if I want to take that out and create
some real working software that he could play on an iPad?
Can I do that? I tell it,
let's create code for this in Python.
Then I give it some of my domain knowledge as
a programmer on what I want for it.
ChatGPT goes and it generates the code.
Here's this excerpt of it.
Then this is the software running in a browser
that he could play on an iPad
or he could play on a laptop.
Now notice what I did.
I went and I basically took content and
I iteratively refined it through a conversation.
All I'm going and doing is putting in paragraphs of
text and very quickly I'm
evolving and creating different drafts
of content which as
a human I can go and use a refine and edit,
but I'm driving this whole process.
It's my creativity, it's
my thought that's driving this process,
but it's accelerating my productivity

and my capabilities and what I can do.
If you think back to our computing
capabilities last year,
there is really nothing on the planet that
could even touch what we just saw,
or certainly nothing on the planet
that was generally available.
Generative AI is absolutely going to
transform what we can do in the workplace,
and it's going to transform what humans can do.
I'm going to talk about that throughout this primer.
: Added to Selection. Press [CTRL + S] to save as a note
Required
English

Help Us Translate

Now we've seen ChatGPT generate a bunch of
Pokemon stories and meal plans
and it's generated softwares,
and that's pretty exciting.
But I want to take a step back and
show you the larger trend that's taking place
that's really going to impact
the workplace and impact every single organization.
When people ask me, well, which
domains are going to be most impacted,
my answer is any domain that
involves a computer. Now, why is that?
That's because our fundamental interface to
computing is about to change
and is in the process of changing right now.
Now we've spent years working with
the Gooey where you can go and all
you can do is whatever buttons
your programmer has created in that,
and you're limited by what has been pre-programmed.

Or if you were like me, you were
a software engineer and you could
go source code and program anything you wanted,
but at really high cost in terms of time and effort.
Then you had to maintain it, and it took a lot of work.
There was nothing that sat in between that
gave you the flexibility of programming
with the ease of use of Gooey and
Generative AI is about to change that.
It's going to supplant the Gooey and programming as
our dominant interface to computing and it's going to
revolutionize what people are capable of.
Let me show you some examples of this.
Let's take a look at
something that we deal with at Vanderbilt all the time,
and probably you deal with it in
your organization too, expense reporting.
Now, this is the Vanderbilt
travel business expense policy.
It's about 20 pages long,
and I have to admit, I don't think I've ever read it,
and I bet a lot of the people in your organization,
and maybe you have never read
your travel and business expense policy either.
It's not exactly what you want to sit down and read.
How can we use Generative AI to change that?
Well, it turns out I actually have read
the Vanderbilt travel and
business expense policy, but not directly.
I've read it through ChatGPT and through a tool called
ChatGPT Advanced Data Analysis. How do we do that?
Will we upload the policy
to ChatGPT Advanced Data Analysis?
We say please read the provided document
so you can answer questions about it.

I give it the PDF.
Now notice what I'm not doing.
I'm not programming anything.
I'm not even bother taking this out of
the PDF and turning it into text of some kind.
I'm not doing an Internet search,
I'm just giving it the PDF of the policy.
Now I'm going to ask a question that looks like
the questions that I actually ask in the real world.
I'm going to say, let's assume I'm in a foreign country,
traveling in a taxi,
and it breaks down on the way to an important meeting.
Can I get reimbursed if I pay someone to drive me
on the back of their motorcycle to
the meeting and assume they're a private citizen?
That actually looks like the types of questions I asked
my business unit manager
and I'm going to show you what ChatGPT says.
It says based on
the information provided in the document,
and I just want to call your attention
to that because we hear
all the time ChatGPT hallucinates.
It wasn't trained after 2021.
It doesn't have access to my information.
Well, this document came after 2021.
This is the latest version from when I filmed this of
the Vanderbilt Travel and Business Expense Policy
and ChatGPT didn't have this thing.
I've actually gone and uploaded it.
It's answering it based on my data or Vanderbilt's data.
It says, the university requires travelers to select
the most reasonable and economical form
of transportation.
That is true.

That's a direct statement out of that document.
Non-conventional transportation options are
not explicitly mentioned in this policy.
It's also telling me what's not in the policy,
which can be really valuable.
Vanderbilt does not have a
separate policy on ride sharing,
and that is a direct statement inside of the policy.
They say that, and it says finally, given these points,
it seems plausible that in an emergency situation where
no other reasonable and economical form
of transportation is available,
the cost of transportation by
a private citizens motorcycle
might be considered an allowable expense.
However, the university's policy
does not explicitly state this,
and reimbursement may be subject to review.
That's absolutely
a Vanderbilt careful answer to my question of,
can I get reimbursed for
this crazy thing that I just did?
It's a very carefully thought through answer.
It's pulled out appropriate portions
of the text from the policy,
and then it's put together a very reasoned response.
Nothing that we had in computing could do
that effectively before November of last year.
If you wanted to do something like this,
that would take an immense amount of effort to try to
create some tool that could answer that question.
But let's take it further and start to look
at how this is going to transform computing.
What if I have a receipt and I want
to know if it complies with the policy?

I'm going to upload a rental car receipt in PDF form.
I'm going to say, please analyze
the attached receipt and let me know if it complies.
Now, this is a totally different thing.
Can you understand the policy and actionably apply it?
This is different from summarizing
the information related to the question.
This is take the policy,
turn it into actionable information
or knowledge and apply it to a receipt.
What does ChatGPT Advanced Data Analysis do?
It says collision damage waiver.
For domestic travel, the CDW should be decline as
rental vehicles are fully insured
through Vanderbilt's insurance portfolio.
The receipt shows a charge for CDW,
which may not comply if this is
domestic travel, that's transformative.
Now, if I had had to do that
before ChatGPT Advanced Data Analysis,
how would I have done it?
Before generative AI,
how would I have solved this problem?
I would have gone and programmed that.
I would have programmed some huge rules engine.
It wouldn't have been just been me.
It would've been a team of people
working in programming and
trying to create that application
that applies all the different rules,
and you probably have such things in
your organization that try to
go through and automate some of this,
and it's hard and difficult to do.
How did I create that application?

I didn't program it.
I just gave it the document
and I gave it the right prompt.
Then I gave it a second document
and a second prompt and now I've got
an application that's analyzing
a receipt based on the travel policy.
That's fundamentally transformative capability.
Let me give you another example
of a transformative capability in
terms of how we access computing and how
accessible computing is going to be to everyone else.
Now, we've always had programmers
over here who could go do whatever they wanted.
They just had to program it and put
in the time and effort,
but everybody else didn't have access to
that same flexibility in computing and capability.
This is the Vanderbilt instructional
and enrollment report
about who's taking classes in
the different schools within Vanderbilt, and it's a PDF.
If I was doing research or work with
this PDF before these tools came out,
before these generative AI tools,
I probably would have had to go and cut and paste
from this PDF into something like Excel,
and then I would have to go and try to calculate or draw
diagrams or something to analyze whatever it is,
but it would have been a ton of manual labor and effort.
As a programmer, maybe I could have automated
this progress or process
by going and writing some custom software,
or maybe I could have found some one
off Unicorn tool that could take

that PDF and these complex details
and produce it into text or Excel sheets,
but probably at great expense
and it would only be limited in the things it could do.
What can we do with generative AI?
Well, I'm going to upload
that instructional and enrollment report
to ChatGPT Advanced Data Analysis.
I'm going to say please extract each page of
this PDF as a separate text file.
It goes through, extracts
all the pages as individual text files.
Now I can go and click on each link and download it.
Now, before to do that,
I would have had to have some specialized tool
to go and extract all the pages and convert them.
I would have had to click a bunch of
buttons and done it manually.
That's neat, it's useful.
What else can I do with it?
I can take it and I can say turn
a particular page of it into an Excel file,
and then here's the link that I can go and click on it to
download all the data that was on
that page converted into Excel.
That's a transformative capability.
If I had had to go and do that manually,
it would have taken me forever.
If I had to go and program something to do
that based on arbitrary tables,
I would have probably highly
customized and figure out how to do it.
Here it is. It's just targeted into Excel for me.
Now I could take it and I can do a bunch of
calculation or visualization in Excel with it,

but why would I do that when I'm using
this new interface to computing that gives me
all radical new capabilities.
Without even leaving the tool
just in the same conversation,
I just follow up and I say
display four interesting visualizations of the data.
Now notice what I'm not telling it.
I'm not telling it what to visualize, what axes,
what colors, what columns
from the data I want it to pull in.
I'm just telling it, give me four interesting things
because I want to go and explore this dataset.
Because as a human being, I want to spend
more time thinking about the data.
I want to spend more time
analyzing and trying to figure out how to
communicate my message and
less time copying and pasting out of PDF
into an Excel file or doing some other task like that.
I want to really accelerate my thinking.
What does ChatGPT Advanced Data Analysis do?
Generates four visualizations for me and here they are.
Now I can go and look at these visualizations,
decide what I want to do,
decide what this means for me,
and that's really powerful.
I didn't have to program to do this,
I didn't have to cut and paste them to Excel.
I didn't have to go and know how to navigate with
an Excel or some other tool
in order to create these visualizations.
I just tell it what I want, and it generates it.
But I thought about it and I was
like if I was really generating these visualizations,

it's not just to generate a bunch of visualizations.
It's probably because I'm going to give
a presentation and wouldn't it be
great if it could turn
those visualizations into PowerPoint slides for me.
I'm going to go back into ChatGPT,
Advanced Data Analysis and I'm just going to
upload a PowerPoint slide template,
and I'm going to tell it save
each visualization as an image,
then insert each of
these images as separate slides
into the attached PowerPoint.
I give it some more information,
some more prompt engineering to get
those images in the right place on the slide and then
also to add a text box next to the image
that where it describes what's in the visualization.
It goes through and it generates my PowerPoint file
and then I can click the download link
and get my PowerPoint file,
and here's the PowerPoint file.
Notice this is the same PowerPoint template
that I'm using for this presentation.
It's gone through and it's inserted each of
those visualizations as a separate slide into
my PowerPoint presentation and it's also put
that text box next to
the visualization describing what it shows.
Now I can spend my time thinking about my presentation,
my message, what I'm analyzing,
what's important in less time on these mundane tasks.
But that's a really transformative capability
and that's going to have huge impact on
the workplace and it's going to go way

beyond PowerPoint files and Excel.
It's going to transform
all kinds of different computing tasks.
Anything we do in computing is going to be transform.
Now why is that?
It's because what you just saw is way
more transformative than you realize,
and I want to talk about why.
Now when I went through
and I started working with this tool,
it was doing something radically new.
Now imagine you're in
your backyard and you've got a shovel,
and you're about to dig a hole.
Right before you go to dig your shovel,
looks down and sees that there's a rock in the way of
where you're going to dig and
your shovel jumps out of your hands,
builds itself a pickaxe,
breaks up the rock,
and then jumps back into your hands,
and you keep digging.
That's exactly what this tool just did.
It went and built itself another tool to solve
a problem that it needed to solve for you and then it
jumped back into your hands and
created that PowerPoint presentation for you.
This is a tool,
generative AI can go and
create other tools to help it accomplish tasks.
When I said go and create the PowerPoint presentation and
put each of these images into it as a separate slide,
this little gray box that we see
right here popped up that says finished working,
that is generative AI building itself a tool.

It's building itself a tool on the fly to help you
solve your problem and then it's using its own tool,
solving a part of the problem and
then coming back and continuing on in the process.
It's anticipating that rock in the way of your shovel.
It's building itself a pickaxe.
It's breaking up that rock and then it's getting
back in your hands to help you.
The tool that it built is software.
What we see right here is what happens when you
expand that little finished work box.
What we see is a Python program that
on the fly it has written that allows it to
generate a PowerPoint presentation and
generate the images for those data visualizations.
It's creating software on
the fly on your behalf to help you solve a problem.
Now we think about these tools and we think about guis as
the way that we
interact with computing but it's about to change.
We're going to be able to go beyond programming,
go beyond the gui,
to something that makes a lot more sense,
that makes computing much more accessible and also
radically speeds up how
quickly we can innovate and do things.
Because we can create a tool that when
we tell it here's the problem I want to solve,
it can go and start building
all the pieces of software that are needed to solve
that problem without us having
to know how to program and it can assemble
them all and solve our problem for us and that's going to
radically transform what we do with computing.
The real innovation that's happening is

that generative AI is in the process of replacing
the gui and is in the process of replacing
programming as our dominant interface to computing.
We're still going to have tons of programmers.
We're going to have computer scientists
and huge demand for them.
We're still going to have guis but
we're going to have a new interface,
which is generative AI that's going to
rapidly take over a lot of the task and
dramatically expand our capabilities
and what we can do with computing,
and what an average user can
do without having to have
specialized knowledge in computer science.
Required
English

Help Us Translate

So what do we need to learn and what does our workforce need to learn in order to be
effective with this new interface to computing?
How do we use generative AI appropriately and effectively?
It's all just a bunch of text, right?
Well, unfortunately, our mental model of a text box is an internet search and
that's the wrong model for thinking about generative AI.
Now, why is it the wrong model?
Well, in internet search, you go and you type in what you want and
it gives you a set of facts back and that's not what generative AI is for.
That's not the right use case for this technology.
And people are still stuck thinking about it as something that generates facts and
then pointing out all the flaws in its facts.
No, that's not how it's going to transform our world and the workplace.
Now, the other problem is, is that we think when we see this text box that we
can type in anything we want, and if the answer that comes back isn't great,
it's not our fault, it's generative AI's fault.
That somehow we should be able to express anything we want in there as poorly as we
want and it should be perfect, whatever it produces, and that's wrong too.
Neither of those models is right.
We have to learn how to go and use it effectively.
And so let's take a look at what we need to know in order to
be effective users of generative AI.

Play video starting at :1:24 and follow transcript1:24
Now to start off, I want to give you some intuition for
how these large language models were trained, these models like ChatGPT.
So when you hear large language model, what they're talking about is these models
where you go and can essentially prompt them to get some output,
ChatGPT being a large language model.
Now, there are lots of nuances beyond what I'm about to show you,
but this is sort of some fundamental intuition
you can have that'll help you in understanding this whole space.
So the way that these models were trained is they were taught to predict
the next word in a sentence.
So they would have gone and they would have shown it something like Mary had
a little and then they would have tried to get it predict the lamb because they're
trying to get it to predict the nursery rhyme,
Mary had a little lamb its fleece as white as snow.
And so they would have gone and shown it the first few words or large amount of
words and then they would have tried to get it predict the next word.
Similarly they would have said Mary had a little lamb and
they would have tried to get it to predict its, they would have then said
Mary had a little lamb its and they would have tried to get it to predict fleece and
they did this over and over in huge amounts of text from the internet.
And over time these models learn patterns in human language and
how they predict what word should come next.
Now, if you think about what we've talked about, basically what we're seeing on
the left hand side is the prompt, and on the right hand side is the output.
So it's basically learning how patterns in the language and
the prompt impact what comes next.
And part of prompt engineering is the discipline of understanding how do you
word things in order to get the output on the other side that you want.
And how to solve different problems by wording your request differently, how to
structure your request effectively, how to overcome issues that you're dealing with.
But it's also broader than that, it's also about how do you take these things and
validate them, how do you make sure that they stay correct and accurate over time,
how do you version these things, how do you build architectures of it all?
All these things are part of prompt engineering, it's much broader than just
the wording, although the wording is part of it as well.
But prompt engineering is like software engineering,
it encompasses a lot of different things, from architecture to best practices,
to the individual wording of different prompts.
And so it's a large field and it's going to be evolving and
really important in business over time.
Now, what we study at Vanderbilt and we're talking about is prompt patterns.
And that's basically documenting patterns in
the language that can be used to solve different problems.
And then providing these patterns as building blocks to people so
that they can learn them and then build off of them and
go and innovate within their individual departments or disciplines.
Now, when I say prompt pattern, this sounds like this crazy thing,
like some really out there thing.
But if you think about it,

we know patterns in our language that we use all the time.
If you go to write a formal letter, you'll say dear so and so comma,
and that dear, is to indicate that this is a formal thing and
the person is indicated so you know who it's addressed to and
the comma to say, okay, now this is what comes next.
And that's a pattern that we use in our language.
And there are things like that that are easy to learn,
that they can help people become very powerful with these tools.
So I'm going to start off with an example of a pattern that we documented that you see in use all over the internet.
And it's a really powerful first pattern to know, and hopefully you'll go and experiment with ChatGPT or Claude or
one of these other generative AI tools afterwards,
because this is one that's really valuable and it's called the persona pattern.
So the idea behind this is, in the real world, if you have a problem and
you need solved, you know who to go to to get the answer or to get the solution or you know what organization to go to or
what department to go to in order to get the solution or organization.
But you don't know what they're going to say to you or
how they're going to arrive at their answer.
You just know, if I have this problem solved, this is where I go for
the solution.
Now, the persona pattern is all about doing this within a large language model.
It's about replicating that real world experience.
And what we do is we go and we tell the large language model, act as some persona,
act as a nutritionist, act as a chief information security officer,
act as a computer science professor, act as the IRS.
And then whenever you produce an output,
produce the outputs that persona would produce.
Basically what we're telling it as, go and act like that persona and
do what they would do.
Now, this seems kind of wild and out there, how is this going to actually work, what can it really do?
Well, let me give you an example.
My wife is a speech language pathologist, she works for the school system and she tests kids, typically three or four before they go to kindergarten,
determine if they need special educational supports.
Now, this is a persona that I can fact check, but
I at the same time I don't know anything about it.
So I go and I say, act as a speech language pathologist.
I'm going to tell you what a three year old said and
you'll write an assessment for me.
The child said, I need way woy.
And I tried to think about what my son sounded like when he was three and
try to capture that in words.
My wife promptly told me that this isn't exactly how they sound
typically at that age, but bear with me.
Now, what does ChatGPT say?
In response to this, it says assessment.

Based on the provided speech sample,
it appears that the child may be experiencing some phonological and
articulation errors, which are common in children around this age.
Consonant errors, it seems that the child is having difficulty with the production
of some consonant sounds, specifically the n and l sounds.
The child said meed instead of need and woy instead of toy,
assuming the intended words were need and toy.
This indicates the child may be experiencing difficulty with
the production and differentiation of these consonant sounds.
Play video starting at :7:35 and follow transcript7:35
Syllable structure, the child's production of the word way may indicate some
difficulty with the correct syllable structure.
It is possible the child intended to say play but produced way instead,
which suggests a substitution of the initial consonant
cluster pl with a single consonant w.
This could be an example of a simplification process called cluster
reduction.
That's wild, notice what I didn't ask it for.
I didn't ask it for phonological errors or articulation errors or
syllable structure or consonant cluster reduction,
because I wouldn't even known how to ask for those things.
So it's doing and allowing us to access computing in a completely different way.
Normally, if I had to program that, I'd have to go and
tell it in explicit detail every single thing it needs to do and exactly how.
But in this case I'm just telling it who to act like or what organization to act
like or what department or inanimate object to act like.
And then it's going and reasoning and
providing outputs that look like they came from that.
Now, I fact checked this with my wife and
also with some other speech therapists and the feedback I got was, hey look,
we're not going to diagnose a child based on one speech sample, but
what it's saying is accurate relative to that speech sample you gave it.
That's wild that it can go and reason at that level of
capability simply by saying act as some persona.
These prompt patterns are building blocks that you can teach
people in your organization to help them go and innovate.
They can then take these patterns and
adapt them and use them to be able to do things that are really powerful,
like thinking about a particular issue from a different perspective.
Act as a Chief Information Security Officer and tell me why I shouldn't
do what I'm about to do and it'd probably give you some insightful
feedback on why you maybe shouldn't click that link in the phishing email.
But it's a really transformative new way of
doing computing that we really have to begin teaching people.
We have to teach the workforce, and we have to do it in a structured way that's
not just, look at a thousand different prompts people came up with, but
instead we start giving them building blocks,
like the beginning of a formal letter should look like this.
We can teach them prompt patterns like the persona pattern that they can then take
and imply and all of their individual disciplines and departments and

they can lead the innovation by adapting the pattern to their context and where they're working.

Now, I've emphasized that this is going to change the way that we interact with computing, that generative AI is going to make us able to control computing using a better interface, in my opinion, than the GUI or programming.

That it's going to give us something unique, new, much more flexible, and much more intuitive in many cases.

But I want to give you a sense of how it's also going to change the types of things, things that we can do.

We think of programming as something that involves going and putting in source code,

like creating source code and programming in Python or some programming language.

And even in the examples that I've shown you, I've shown the tool going and generating code, but

we actually don't even have to have code anymore to begin programming.

And I want to show you what I mean by this.

Everyone can program with prompts.

And I want to show you what this means.

So I'm going to go and I'm going to create a basic program.

And here's my program, I'm going to say, whenever you generate output, turn it into a comma-separated value list.

So a comma-separated value list is basically a format for describing tabular data like you would put into Excel, it's just basically a series of columns separated by commas.

So like Name, Role, whatever you want, Jules White, teacher, that type of thing.

So I'm going to say, write a program which is basically whenever you generate output,

turn it into a comma-separated value list.

Now notice this is just text.

Now I'm going to go and run my program.

I follow up in the next prompt in the conversation, and I say, my name is Jules White and I'm teaching a course on prompt engineering.

And that's the input to my program.

And what does generative AI do?

In this case, chat GPT.

It says, great, here's a CSV list with that information, Name,
Course, Jules white, Prompt Engineering, that is CSV.
I've basically written a program that converts whatever I put in the next prompt into a CSV that I could copy and paste into a file save and import into Excel.
Now, notice what I didn't do, I didn't write a bunch of programming language instructions, I didn't go into great detail on how to do this.
And in fact, if I had a program,
this would probably take a lot of work to do the natural language processing, and it figured out automatically how to map into the columns.
But notice, I didn't even tell it what the columns were, I didn't tell it name and course, I mentioned them when we were talking, but
I didn't put it into my prompt.
It just figured it out that it needed to generate something that worked.
And this is one of the ways that generative AI is really different with computing, is it tries to come up with a solution that works it doesn't require every last detail to be specified.
It's constantly trying to generate something that will work and so
it makes it more accessible and user friendly.
So it generated Name, Course, it then appropriately semantically mapped.
And notice that capability is a really complex sophisticated capability
that would have been really hard to do any other way before these tools came out.
Now, let's say I don't want just name and role, let's say I want more than that.
I can simply go and modify my program and I follow up in the conversation and I say,
from now on the columns of the comma-separated value list should be NAME, COURSE, ROLE.
So I'm modifying my program just by telling it, hey, here's the new rule, and
then I go back and I give it exactly the same input in the next prompt.
My name is Jules White and I'm teaching a course on prompt engineering, and
it says here is your information format as a CSV list, and
now it says NAME, COURSE, ROLE, Jules White, Prompt Engineering, Teacher.
And notice what it's done, it's modified my program and
then it's run the new program that I specified on the same input.
This is exactly the same input I gave last time, but
now I'm getting a different output.

I have a different program that's running, and it's giving me different output.
So we can actually go and start programming through prompts and
having complex logic just in prompts themselves.
Play video starting at :4:4 and follow transcript4:04
So let's say for example, I want to modify it further, I just follow up and
I say in addition to whatever I type in, generate additional examples that fit
the format of the CSV that I've asked you to produce.
Now notice that's a really, really hard task, and
before these tools came out that would be extremely difficult to generate additional
fate data that fit the format that we've just specified.
So not only now are we programming, but we're also tapping into some of
the most sophisticated computing capabilities that we've ever had,
and we're doing it on the fly with some text.
So all I've done is modified my program by giving it this new instruction,
and then I run it again on my original input, which is my name is Jules White and
I'm teaching a course on prompt engineering, and
now it produces new CSV that follows the new program, which is NAME, COURSE, ROLE,
Jules White, Prompt Engineering, Teacher, and then a bunch of synthetic data.
And that is completely transformative in terms of capability.
Now let's make it more concrete.
Now, CSV, don't get me wrong, like,
extracting structured CSV from all kinds of data is a hugely important problem that
will have really important impacts in all kinds of domains.
Everything from taking focus groups and turning the conversations and focus groups
into CSV that can then be actionable and have concrete product insights,
to taking what people are saying and how they're reviewing products and
turning it into concrete information about which product are they reviewing and
what's their feeling, and what are the features they mentioned,
all kinds of different capabilities.
Or free text, you imagine giving people surveys and they enter in all this free
text information and now you got to laboriously look at it.
Well, you can actually turn that free text directly into CSV or some other structured
format like JSON for your business to then automate and do things with it.
But I'm going to give you something more close to home, probably for all of us.

And that's that weekend problem we face when we decide to just check out and not answer our email over the weekend.
And I don't know if you're like me, but what happens is I come in on Monday morning and I start looking at my email and there is some gigantic email thread, where everybody else didn't check out over the weekend and
they all had a very long conversation that I now need to read and
try to figure out who said what and when and what am I supposed to do,
what are my action items, what are the questions for me, what do I do about this?
So I'm going to use prompt engineering and
I'm going to write my own program to solve this problem.
So here's what I say, act as my personal assistant, whenever I give you a long email thread, you will create a bulleted list of questions and action items for me.
The bullet should be numbered, if I type in a number of a bullet, you will summarize the conversation from the email chain around that bullet in one paragraph.
If I type in a bullet and a number, you will draft a reply for
me to read with that many paragraphs.
So I'm creating a whole little application for myself, and it's even got a menu system, I can type in an Identifier of a bullet and it'll summarize it.
And if I type in the Identifier of a bullet and
I give it a number of paragraphs, it'll just draft a reply for me.
This is like my email accelerator, right?
And all I did to program it was give it a block
of text explaining clearly what I wanted.
Now, I'm going to go and cut and
paste a very long email chain when we were scheduling my generative AI class that I'm co teaching with four other faculty members.
So there's five of us on this email chain, plus a number of other people in the university, and we were trying to figure out what room and time and
what all the information for this course.
Very long email thread, lots and lots of discussions,
lots of people weighing in on different aspects, how do I go and decipher this?
Well, I'm going to copy and paste that long email thread in right after and inputting those instructions, and note, this is one of those important things everybody tells you, it was trained on 2021, it can't do anything after that.

Of course it can, you just have to give it the information in a prompt.
So I'm giving it my email chain in a prompt, and then what does it do?
It says, here is a bulleted list of questions and
action items based on the email thread.
Questions and action items.
What are the suggested title and transcript for the course?
And then it has the title, possible examples of what I'm looking at.
Can we start working on a draft syllabus?
And then look at this 1.7 at the end, it says, have the room and
schedule been finalized?
T/Th 9.30 in ESB 048.
And that's actually real information that it's dissected from the conversation.
And that would have taken me a long time to go and find all of that,
follow up on it, potentially, so I'm going to follow up on that bullet.
I'm going to find out and have it summarize, what happened.
So I'm using my menu system that I wrote into it, I type 1.7,
and I told it if I type in the Identifier of a bullet,
you summarize the conversation related to that bullet.
And it comes back and it says in the email thread, the team discusses the room and
schedule for the course, Charreau suggests the T/Th 09:30,
a.m time slot with the only available room at that time is Engineering and
Science Building room 048, with a capacity of 60.
Jules approves of this room choice, mentioning is a good size for
the first run of the class.
The conversation does not provide any further details on the room and schedule,
so it seems the team has agreed.
That's wild, right?
Because not only is it telling who said what, but it's also saying only two
people weighed in on what room and time and nothing else was said.
So it seems that everybody's agreed.
So it even is intuiting and understanding that because nobody else said no, nobody
else said yes, it's probably the right room, but it's also pointing that out.
That's a really sophisticated, complex task, and how did I get there?
All I did was I gave it some text,
a block of text explaining my program, I've programmed in a prompt,

so there's going to be two huge revolutions in computing.
One is that Generative AI has all these unique computational capabilities,
like when it looked and it intuited and understood that, hey,
nothing else in this email chain talks about or
agrees to that room, so it seems that the team has agreed.
That's a really complex capability or
when we took that unstructured text where we said, my name is Jules White,
I'm teaching a course on prompt engineering, and it turned it into CSV and
it correctly semantically mapped it to the right columns.
Those are unique new capabilities.
And the second important thing is it's going to be a new interface to computing,
because I can go and
control computing with natural language in completely new ways.
I can go and generate source code for programs and run it, but I can also,
just within a prompt, write an entire application and
begin doing things that are useful and valuable to me, like summarizing my email.
And we're doing all of this off of very simple tools right now, and
the tools are going to radically explode in the coming year and months ahead.
Required
English

Help Us Translate

I'm passionate and
excited about a particular vision of where we're going with generative AI.
And it's not artificial intelligence that tries to replace and supplant people, but
it's about augmented intelligence, where we amplify and augment human creativity
and problem solving skills, where we basically give people new capabilities.
It's like an exoskeleton for the mind that can amplify by that human spark within
you, enable you to create and do more interesting things than ever before.
But not something that's going to take away who you are.
But allow us to spend more time thinking critically, creating new and
exciting things that we couldn't create before or
have more time to interact with each other and be human.
That's what I think the real potential of this is and not going and
replacing people.
And I think it's really important that we approach generative AI from that
standpoint.
So, I'm going to give you a framework to think about how to
appropriately use generative AI.
And the framework that I use is called ACHIEVE.
So this is a framework that I created to help think about what is the right way to
use it to achieve this vision of augmented intelligence,

as opposed to something that's going to take away from us by
trying to replace us with this artificial substitute.
And we've seen how artificial things of all kinds don't really substitute for
the real thing.
So, ACHIEVE is about a couple of things, it's about how we use and
think about ways of leveraging these new capabilities.
One, to aid in human coordination, and
I'm going to show what each of these things means in a minute.
To cut out tedious tasks so we can spend more time thinking, reasoning and
interacting with other human beings and less time on these things that honestly,
probably people don't really enjoy.
To help provide a safety net so that it can help us catch mistakes.
It could be as simple as catching a mistake in spelling to
something more complex like we'll see in a minute.
And then the IEV, I give it three letters because it's so important, and
this is the one that I think really, really matters.
When you're using these tools and approaching them,
it's all about inspiring better problem solving and creativity within the human.
You want the human thinking when they're using their tools,
you want them getting excited and you want it sparking their imagination.
You want them being an integral part of this thing,
not something that's like sitting back and letting it do all the work.
And then finally, we want to enable great ideas to scale faster by doing all kinds
of things, everything from the Pokémon meal plan that I then turn into software
and games like on the fly in a conversation.
We want to enable scaling up and doing crazy things like that, and these tools
will allow us to do all of these things, scale up great ideas really quickly.
So I'm going to give you an example, and I want you to imagine we're going to
design
a workshop and then we're going to host this workshop.
And it's going to be a faculty workshop at a university where we're bringing in
a bunch of participants and we have to solve and deal with a bunch of different
issues from how do we communicate with the participants?
To how do we plan the workshop to dealing with unexpected issues and
like childcare to, how do we follow up with them and
communicate with afterwards and be engaging?
So we're going to see a lot of real world problems that we see and
we're going to use generative AI within the ACIEVE framework to go and solve and
make an impact.
So let's start with aiding human coordination.
Now we don't think of these tools as something that can help human beings, but
they can be really helpful in helping us coordinate with each other.
So, I've gone and I've recorded the first meeting where we were planning this
workshop and I've captured notes from it.
Maybe this is a virtual meeting where you just recorded the raw transcript or
maybe you took notes and typed them in.
And now I'm going to tell it, act as my personal assistant, read the following
meeting transcript and provide me a summary of the key points of discussion.
It goes through and
reads it and that summary could be really valuable in itself.

Meeting summaries are great, but
I'm going to do something more important with it for human coordination.
I'm going to tell it, list any ambiguities in the plan that the team should address.
How often as human beings do we build plans but then they're ambiguous and
we don't coordinate well because one person interprets something a different
way than another person and then that leads to issues.
That's a really important thing to be able to do, and you can imagine sitting at
the end of a meeting just putting in the meeting notes and saying, hey,
what's ambiguous about this?
Let's talk about it, or taking in a specification or plan that was written in
advance and getting feedback on what might be ambiguous.
Now let's say it's wrong, well, what is the harm in that?
Nothing, in fact, it's good because as a team we will then talk about here's what
it said is ambiguous, do we think it is or not and why?
And then we've basically gotten more insight onto what we're doing and
we can make sure that we are not missing something.
We're getting a second pair of eyes on our human coordination.
It says, for example, the exact number of registrations for
the July workshop is not provided in the transcript,
this information is critical for planning and logistics.
And then it says, there is no clear deadline for when John should provide
his draft for review, which could lead to delays or miscommunication.
So it's going out and actually reading the notes and the plan and
then coming up with really concrete things that aren't clear from the plan.
Play video starting at :5:31 and follow transcript5:31
Now, what else can we do with it?
Well, we can think about who should follow up on it.
So we can say, for example,
who should follow up on each of the listed ambiguities and
if we have a transcript where it's actually recorded who said what.
It can tell us, for example, the number of July workshop registrations,
which it said was an ambiguity.
Sam Grey, who was directly asked about this,
should provide the information to Daniel Brown.
Or if we look at the deadline for the draft review, Reema Mistry,
who initially requested the update, should set a clear deadline for John Burke.
And so it's beginning to help us as a team coordinate better and
think through ambiguities and way of solving the problem.
Play video starting at :6:11 and follow transcript6:11
Let's talk about the C, the cutting out tedious tasks.
Play video starting at :6:16 and follow transcript6:16
We spend so much time every day on things that aren't maybe the best use of our
critical reasoning and creativity skills as human beings.
And honestly, a lot of times they're what the tasks that make jobs hard to
deal with and make people want to leave.
So how do we use it to cut out tedious tasks?
Well, in the planning of this workshop, we've created a survey and
we've allowed people to in free text write how are they using ChatGBT today and
anything they want.
Now, as the person who's going to teach this workshop,

I could go and manually try to cut and paste that information into different segments of the audience and try to figure out who's using it different ways and do all that analysis, but that's a lot of work.
So I'm going to upload the survey responses, I'm going to say,
this is the list of people that registered for our workshop on prompt engineering and ChatGBT, describe the data in this file.
And then I'm going to follow up and I'm going to say the registration had,
how are you using ChatGBT, Barred, Bing, etc.,, as a question?
Can you turn this into a list of usage and
then group the responses underneath your types?
And so I'm asking it to look at that free text response and categorize or group people together and group their responses under these different categories.
And so it comes back with categories like not using it yet, not using it at all, and it groups all the different responses under different categories automatically.
Now I could have copied and pasted and come up with all these categories and done it manually, but I want to spend my time thinking about my audience and thinking about how best to communicate with them.
And then I can go and I can think about, well, let me think of other ways
of grouping and I can quickly have it regroup everything so
that I can look at it in different ways.
And this is one of the real powers of the tool,
is I can quickly go and look at different groupings.
And before it would have taken so much time, so much tedious cutting and pasting and reorganizing, that I wouldn't have explored many different ways
of segmenting my audience and understanding them.
But now I can just say, give me three more possible groupings, and it says, okay, well, here's a grouping by specific task.
Not using it, writing and communication, coding and development, research and teaching.
And I could then follow up and look at those and
look at the individual responses and have it shape all of that.
But the key is it gives me more time as a human to think about how I'm going to interact with these other human beings.
It's taking away that tedious task, so I have more time for
the thing that really matters.
Let's talk about help providing a safety net.
This is something really powerful that we can do,
is we have to worry about as humans making mistakes all the time, and
we'd like to try to make it less likely that we would or that we can catch them.
So, one of the things that I'm thinking about is I'm going to go and
present this material and I've presented it to a different audience previously,
maybe I presented it to a bunch of computer science professors.
And so they're going to have different terms that they're going to
be familiar with.
And I don't want to start my presentation and
realize I'm using all these terms that I haven't defined and
make the mistake of losing my audience in the first ten slides.
So I upload my 120 slides to ChatGBT Advanced Data Analysis,
and I say read my first ten slides and
tell me any technical terms that I used that I haven't defined and

an audience of manufacturing business executives might not be familiar with.
And it comes back with for that particular audience, here are possible terms,
it's providing me a safety net in terms of my presentation.
So now maybe I won't lose my audience because it's giving me second sight on
what is in there that maybe I haven't defined, that maybe I need to talk about
and explain in more detail because my audience has changed.
Play video starting at :9:59 and follow transcript9:59
Or let's look at something that we can do in terms of coordination and a safety net.
And this is a huge problem when we're building software, but
it's also a problem in other domains.
And that is we're working on a problem, we have two teams that go and
work independently on different parts of the problem.
But when the teams come together,
they realize they've made conflicting decisions.
And in software, the MIS means that we can't integrate our two pieces of
the problem, our two pieces of the software don't work together.
And now we've just wasted all this time and effort and
money where in the case of this workshop,
maybe we have two teams working on one on logistics and one on the overall planning.
And they've made conflicting decisions, and that could be a problem.
So let's try to solve that problem.
So I say, I'm going to upload the second team's meeting notes and
I say a second team working on logistics for the workshop just met.
These are their meeting notes, please point out any potential conflicts and
decisions made by this team and the other team.
And it comes back and it says venue, size and layout.
The logistics team suggests using the university's main auditorium for
the sessions, which conflicts with the initial plan to conduct the sessions in
a smaller setting, presumably in conference Room B.
Or IT requirements, the logistics team list of necessary equipment includes
a high resolution projector, multiple wireless microphones, a good sound system, and several laptops.
This contradicts the initial team's IT requirements,
which were likely based on a smaller room setup.
So now it's helping us catch not only mistakes in our individual things like
presentations, but also coordination, potential mistakes that could be
really impactful and so helping us do a better job as humans.
And I love it when my mistakes are caught before they become a problem, like before
I'm giving the presentation, or before we realize we've scheduled the wrong room.
Play video starting at :11:50 and follow transcript11:50
Now let's talk about the IEV.
This is the most important piece from my perspective, and
it's all about inspiring better problem solving and creativity with humans.
Now we hear all the time about how AI is biased, but
we as humans are incredibly biased, and
we actually produced the biased data that led to the training of AI.
But what's interesting is it can help us overcome some of our own biases if we use it correctly.

So we have biases, for example, in problem solving, like we know one way of solving
a problem, so we automatically try to solve every problem that way.
We have bias in confirmation, we have confirmation bias on data,
we're going to look at it, we're going to try to see what we want to see in it.
And so, one of the uses of generative AI is to help us think outside the box,
to be more creative and inspire new ways and new solutions.
So, we've got an interesting problem with our workshop, and
that's that we need to provide childcare.
And we've looked at using our audience segmentation from our free text forms,
and we've realized it's all fourth graders, and
it turns out they're all interested in Pokémon.
Well, can we build some interesting content that goes along with what their
parents are learning, but is appropriately targeted to this younger audience?
So we say, please provide three different 90 minutes lesson plans for
teaching large language model concepts to fourth graders using Pokémon examples and
hands on exercises.
And it comes back with different lesson plans,
like what is a large language model and how it works?
Imagine a game where you describe a Pokémon and
I have to guess which one you're thinking of,
the better your description, the easier it is for me to guess.
Highlight how a large language model is similar,
it tries to guess the next word based on the words that is already seen.
So what it's doing is it's then giving us ideas that we can go and inspire.
We're not going to just take them, we're going to use them as draft output that we
as humans respond to, we edit, we improve on it.
And in fact that's one of the most important things that you
as a user of this need to know is that you have to engage with it,
you have to take it as draft and you have to improve and iterate and work on it.
You have to inject your thinking and creativity into the process, otherwise,
you're going to be automated out realistically.
If all you're doing is copying and pasting quiz questions in and
getting the answers to cheat, you're not going to be employable long term.
But if you're somebody that goes in and engages with the tool in creative and
interesting ways, if you view the output as draft and you're going to edit and
refine it and improve on it,
that is the philosophy that you need to have with the tool.
And that's the philosophy we need to encourage with people is don't take it for
face value, take it and really look at it, really understand if there are flaws,
really do your work to improve it and then get really creative and insightful and
use it to solve bigger and better problems.
Play video starting at :14:39 and follow transcript14:39
Now, let me give you another example of this., I've got these 120 slides and
I'm going to upload them and I'm just going to ask it to summarize them.
But then I'm going to something that's really valuable.
After this summary comes in, I'm going to go and say act as a skeptic of everything
I say in this presentation, find flaws in my assumptions, assertions and
other key points and 10 generate ten hard questions for me, and what does it do?

It comes back with here's the assumptions you're making in your slides,
you have the assumption that ChatGBT and
similar tools will bring incredible increases in productivity.
Question, isn't it possible that these tools might also introduce new
challenges and inefficiencies, such as the need for additional oversight or
the risk of misuse?
And that's a great question and I should address it.
And by going in and using the tool it's inspiring me to think more critically,
to get outside the box of my perspective on this particular content and
start improving in thinking.
And it's not replacing me, it's making me think, and then I respond and I think
about how to address that question, that's a fundamentally new capability.
Now let's look at the E, enabling great ideas to scale faster.
We've seen a lot of examples of wild new capabilities and things we can do,
but I want to talk to you about this one.
So I'm imagining at the end of this workshop, I want to go and
I want to send personalized emails to every single person to get them excited
about ChatGBT or generative AI or Claude or whatever it is that I want.
And I need some way of engaging that audience.
So I come up with this idea, it would be awesome if in every one of my emails
there was a custom prompt that was specific to that person's department and
what they do that showcased something that you could potentially do with ChatGBT.
But for me to go and try to scale that idea up would be really hard, right,
to think of prompts for every single department and have the domain knowledge
to make them even close to relevant is not going to be easy.
And how do I do that as an individual person?
Well, using a tool like this, I can scale that idea up.
So I say, I'm going to create customized post workshop emails for
every attendee and I'm just following up in the conversation.
I want you to create a list of interesting ChatGBT prompts and
ways of combining large language models with each discipline or department.
And basically I'm telling it generate prompts that people could then take out of
those emails, copy and paste into ChatGBT and see it doing something interesting and
relevant to them, to help get them excited to go and engage.
But also notice what's happening, we're asking ChatGBT to come up with ideas for
how other people could talk to itself.
How would people go and talk to ChatGBT?
It's proposing ways that you could talk to it,
that's a wild new capability that's transformative.
And it comes back and it has a whole bunch of different ideas and
I'm going to zoom in on two of them.
So here's the marketing prompt that somebody from marketing could cut
and paste.
Design a comprehensive marketing campaign for launching an eco-friendly product
incorporating digital and traditional channels.
And now they could copy and paste that out of the email and
see it doing things that are relevant to their domain.
Or alumni relations, outline a year long engagement strategy to reconnect with
alumni, including events, communications and collaboration opportunities.

They could copy and paste that out of the email.
Now if I had to go and think of all those ideas from scratch and
write them all down, I couldn't scale this idea up.
Now, the important thing was the idea, the template that I had in my idea of
how I would do this, and then I'm using that template to scale up and
it's all about that ability to go and scale things up.
It's all about augmented intelligence, not about replacing people,
not about this artificial substitute.
It's about that exoskeleton for the mind and using it through the ACHIEVE
framework to augment and amplify our creativity and problem solving skills,
so we can put that exoskeleton on and do amazing, amazing things.
So, what's coming next is the tools.
Now we've done all of this so far with basically cutting and
pasting in and out of a chat interface.
But the future is going to be in these tools that integrate generative AI
into the tools we use in the workplace every day.
And they're going to transform what we can create, how fast we can create,
how big of problems we can solve, how creative we can be,
it's going to give a totally new medium for expression.
And how we access computing is going to fundamentally change.
And the future of computing is generative AI as our interface to it, of being
able to radically change what we can do and how we can control these things.
So when you're thinking about how do we engage with this,
you have to think about it from this augmented intelligence perspective.
Think about the ACHIEVE framework and how can you go and apply that framework in
your discipline, in your domain, to do things that benefit people, that give them
that exoskeleton for the mind, that not that replace them in some artificial way.

How can you go learn how to be

a master of all of this new generative AI capability?

Well, the way you do it is through

learning prompt engineering.

I'm going to give you

some resources that you can follow up

with to become a master of prompt engineering.

First, I have a course on prompt engineering on Coursera.

It has about over 150,000

people that have taken it to date.

There's also a great community

there that's coming up with

phenomenal ideas and innovations in all disciplines,

from finance to banking,

to law, to marketing, to art.

You imagine that people are
out there doing it and innovating with it.
Second, is if you've
seen a lot of the interesting things
that I've done with documents and automation,
there's a course that I have on
ChatGPT advanced data analysis.
Although this tool sounds like
it's something that doesn't apply to most people,
it applies to everybody.
Because really what this tool is a tool for automation.
It's a tool for how you take
your documents and begin automating different types of
analysis like the creation of
that PowerPoint slide or PowerPoint deck,
or the creation of those visualizations,
or how I began asking questions about the travel policy.
All of those things are done through
ChatGPT advanced data analysis.
It's not just a advanced data analysis tool,
it's really automation for everybody.
Then if you're a teacher,
or you're somebody who's delivering
educational experiences and you want to see
more concrete examples of how you go and
innovate with teaching and
creation of learning materials,
I have a course on innovative teaching with ChatGPT,
and this is all about doing things like lesson planning,
creation of examples, creation of assessments,
all of these types of things that can be really useful.
Then finally, if you want to read
the prompt patterns that I've talked about and
a bunch of patterns that are
really helpful regardless of what domain you're in,

where you can go in and learn how to take
these building blocks and then
go and innovate with them within your discipline,
I'd encourage you to read our paper on
a prompt pattern cataloged
enhanced prompt engineering with ChatGPT.
Right now, if you go to Google Scholar and you search,
prompt engineering will typically
be one of the number 1 result.
But over time that may change and here is
the citation to the paper
in case you want to go and find it.
It's freely available on archive.
Thank you so much
for participating in this primer on Generative AI.
I hope you'll be as excited about this space as I'm.
It's going to fundamentally change
how we interact with computing,
but also this vision of
augmented intelligence is what gets me so excited.
I want to put on that exoskeleton for the mind.
I want to go and do crazy new creative things and
scale up my ideas using the achieved framework.
I want to do it in a way that
amplifies and augments me as a human.
I want to go and explore things that help us to amplify
those individual human sparks
in each individual person to spend
more time interacting with the people that
we care about and the other human beings around us.
I don't want to be replaced with
artificial intelligence and I
don't want to replace other people.
I want to think about how do we augment who we
are and we take advantage of

those uniquely human capabilities,
those unique creative capabilities,
and we really build this into a vision
that is based on augmentation of intelligence,
not supplanting our intelligence
with something that's an artificial substitute.
Required
English

Help Us Translate
Hello, and welcome to “Introduction to Large Language Models”.
My name is John Ewald and I am a Training Developer here at Google Cloud.
In this course, you learn to:Define Large Language Models (LLMs)
Describe LLM Use Cases Explain Prompt Tuning
and describe Google’s Gen AI Development tools
Play video starting at ::20 and follow transcript0:20
Large Language Models (or LLMs) are a subset of Deep Learning.
To find out more about Deep Learning, see our Introduction to Generative AI course video.
LLMs and Generative AI intersect and they are both a part of Deep Learning.
Another area of AI you may be hearing a lot about is generative AI.
This is a type of artificial intelligence that can produce new content –including
text images, audio, and synthetic data.
So, what are large language models?
Large language models refer to large, general-purpose language models that can be pre-trained and
then fine-tuned for specific purposes.
What do pre-trained and fine-tuned mean?
Imagine training a dog.
Often you train your dog basic commands such as sit, come, down, and stay.
These commands are normally sufficient for everyday life and help your dog become a good
canine citizen.
However, if you need a special-service dog such as a police dog, a guide dog, or a hunting
dog, you add special trainings.

This similar idea applies to large language models.
These models are trained for general purposes to solve common language problems such as
text classification, question answering, document summarization, and text generation across
industries.
The models can then be tailored to solve specific problems in different fields such as retail,
finance, and entertainment, using a relatively small size of field datasets.
Let’s further break down the concept into three major features of large language models.
Large indicates two meanings, First is the enormous size of the training dataset, sometimes
at the petabyte scale.
Second it refers to the parameter count (in ML, parameters are often called hyperparameters).
Parameters are basically the memories and the knowledge that the machine learned from
the model training.
Parameters define the skill of a model in solving a problem, such as predicting text.
General-purpose means that the models are sufficient to solve common problems.
Two reasons lead to this idea: First is the commonality of a human language regardless
of the specific tasks, and second is the resource restriction.
Only certain organizations have the capability to train such large language models with huge
datasets and a tremendous number of parameters.
How about letting them create fundamental language models for others to use?
This leads to the last point, pre-trained and fine-tuned, meaning to pre-train a large
language model for a general purpose with a large dataset and then fine-tune it for
specific aims with a much smaller dataset.
The benefits of using large language models are straightforward: First, a single model
can be used for different tasks.
This is a dream come true.

These large language models that are trained with petabytes of data and generate billions
of parameters are smart enough to solve different tasks including language translation, sentence
completion, text classification, question answering, and more.
Second, large language models require minimal field training data when you tailor them to
solve your specific problem.
Large language models obtain decent performance even with little domain training data.
In other words, they can be used for few-shot or even zero-shot scenarios.
In machine learning, “few-shot” refers to training a model with minimal data and
“zero-shot” implies that a model can recognize things that have not explicitly been taught
in the training before.
Third, the performance of large language models is continuously growing when you add more
data and parameters.
Let’s take PaLM as an example.
In April 2022, Google released PaLM(short for Pathways Language Model), a 540 billion-parameter
model that achieves a state-of-the-art performance across multiple language tasks.
PaLM is a dense decoder-only Transformer model.
It has 540 billion parameters.
It leverages the new Pathways system, which enabled Google to efficiently train a single
model across multiple TPU v4 Pods.
Pathway is a new AI architecture that will handle many tasks at once, learn new tasks
quickly, and reflect a better understanding of the world.
The system enables PaLM to orchestrate distributed computation for accelerators.
We previously mentioned that PaLM is a transformer model.
A Transformer model consists of encoder and decoder.
The encoder encodes the input sequence and passes it to the decoder, which learns how

to decode the representations for a relevant task.
Play video starting at :5:21 and follow transcript5:21
We’ve come a long way from traditional programming, to neural networks, to generative models!
In traditional programming, we used to have to hard code the rules for distinguishing a cat - type: animal,
legs: 4, ears: 2,
fur: yes, likes: yarn, catnip.
Play video starting at :5:40 and follow transcript5:40
In the wave of neural networks, we could give the network pictures of cats and dogs and
ask - “Is this a cat” - and it would predict a cat.
In the generative wave, we - as users - can generate our own content - whether it be text,
images, audio, video, etc.
For example, models like PaLM (or Pathways Language Model or LaMDA (or Language Model
for Dialogue Applications) ingest very, very large data from multiple sources across the
Internet and build foundation language models we can use simply by asking a question - whether
typing it into a prompt or verbally talking into the prompt.
We - as users - can use these language models to generate text or answer questions or summarize
data, among other things.
So, when you ask it “what’s a cat”, it can give you everything it has learned
about a cat.
Let’s compare LLM Development using pre-trained models with traditional ML development.
First, with LLM Development, you don’t need to be an expert.
You don’t need training examples, and there is no need to train a model.
All you need to do is think about prompt design, which is the process of creating a prompt
that is clear, concise, and informative.
It is an important part of natural language processing (NLP).

In traditional machine learning, you need expertise, training examples, train a model and compute time and hardware.
Let’s take a look at an example of a Text Generation use case.
Question answering (QA) is a subfield of natural language processing that deals with the task
of automatically answering questions posed in natural language.
QA systems are typically trained on a large amount of text and code, and they are able
to answer a wide range of questions, including factual, definitional, and opinion-based questions.
The key here is that you needed domain knowledge to develop these
Question Answering models.
For example, domain knowledge is required to develop a question answering model for
customer IT Support, or Healthcare or Supply Chain.
Using Generative QA, the model generates free text directly based on the context.
There is no need for domain knowledge.
Let’s look at three questions given to Bard, a large language model chatbot developed by
Google AI.
Question 1 - This year’s sales are 100,000 dollars.
Expenses are 60,000 dollars.
How much is net profit?
Bard first shares how net profit is calculated then performs the calculation.
Then, Bard provides the definition of net profit.
Here is another question: Inventory on hand is 6,000 units.
New order requires 8,000 units.
How many units do I need to fill to complete the order?
Again, Bard answers the question by performing the calculation.
And our last example, We have 1,000 sensors in ten geographic regions.
How many sensors do we have on average in each region?
Bard answers the question with an example on how to solve the problem and some additional
context.
In each of my questions, a desired response was obtained.

This is due to Prompt Design.
Prompt design and prompt engineering are two closely related concepts in natural language
processing.
Both involve the process of creating a prompt that is clear, concise, and informative.
However, there are some key differences between the two.
Prompt design is the process of creating a prompt that is tailored to the specific task that the system is being asked to perform.
For example, if the system is being asked to translate a text from English to French, the prompt should be written in English and should specify that the translation should
be in French.
Prompt engineering is the process of creating a prompt that is designed to improve performance.
This may involve using domain-specific knowledge, providing examples of the desired output,
or using keywords that are known to be effective for the specific system.
In general, prompt design is a more general concept, while prompt engineering is a more
specialized concept.
Prompt design is essential for, while prompt engineering is only necessary for systems
that require a high degree of accuracy or performance.
There are three kinds of Large Language Models - Generic Language Models, Instruction Tuned,
and Dialog Tuned.
Each needs prompting in a different way.
Generic Language Models predict predict the next word (technically token)based on the
language in the training data.
This is an example of a generic language model - The next word is a token based on the language
in the training data.
In this example, the cat sat on —- the next word should be “the” and you can see that

“the” is the most likely next word.
Think of this type as an “auto-complete” in search.
In Instruction tuned, the model is trained to predict a response to the instructions given in the input.
For example, summarized a text of “x”, generate a poem in the style of ‘x”, give me a list of keywords based on semantic similarity for “x”.
And in this example, classify the text into neutral, negative or positive.
In Dialog tuned, the model is trained to have a dialog by the next response.
Dialog-tuned models are a special case of instruction tuned where requests are typically
framed as questions to a chat bot.
Dialog tuning is expected to be in the context of a longer back and forth conversation, and
typically works better with natural question-like phrasings.
Chain of thought reasoning is the observation that models are better at getting the right
answer when they first output text that explains the reason for the answer.
Let’s look at the question: Roger has 5 tennis balls.
He buys 2 more cans of tennis balls.
Each can has 3 tennis balls.
How many tennis balls does he have now?
This question is posed initially with no response.
The model is less likely to get the correct answer directly.
However, by the time the second question is asked, the output is more likely to end with
the correct answer.
Play video starting at :11:53 and follow transcript11:53
A model that can do everything has practical limitations.
Task-specific tuning can make LLMs more reliable.
Vertex AI provides task specific foundation models.
Let’s say you have a use case where you need to gather sentiments (or how your customers
are feeling about your product or service), you can use the classification task sentiment
analysis task model.

Same for vision tasks - if you need to perform occupancy analytics, there is a task specific
model for your use case.
Tuning a model enables you to customize the model response based on examples of the task
that you want the model to perform.
It is essentially the process of adapting a model to a new domain or set of custom use
cases by training the model on new data.
For example, we may collect training data and “tune” the model specifically for
the legal or medical domain.
You can also further tuned the model by “fine-tuning”, where you bring your own dataset and retrain
the model by tuning every weight in the LLM.
This requires a big training job and hosting your own fine-tuned model.
Here is an example of a medical foundation model trained on Healthcare data.
The tasks include question answering, image analysis, finding similar patients, etc.
Fine-tuning is expensive and not realistic in many cases.
So, are there more efficient methods of tuning?
Yes.
Parameter-Efficient Tuning Methods are methods for tuning a large language model on your
own custom data without duplicating the model.
The base model itself is not altered.
Instead, a small number of add-on layers are tuned, which can be swapped in and out at
inference time.
Generative AI Studio lets you quickly explore and customize generative AI models that you
can leverage in your applications on Google Cloud.
Generative AI Studio helps developers create and deploy generative AI models by providing
a variety of tools and resources that make it easy to get started.
For example, there is a:
Library of pre-trained models Tool for fine-tuning models

Tool for deploying models to production Community forum for developers to share ideas
and collaborate.
Build generative AI search and conversations for customers and employees
with Vertex AI Search and Conversation (formerly Gen App Builder).
Build with little or no coding and no prior machine learning experience.
You can create your own: Chatbots, Digital assistants, Custom search engines, Knowledge bases,
Training applications, and more.
PaLM API let's you test and experiment
with Google's Large Language Models and Gen AI tools.
To make prototyping quick and more accessible, developers can integrate PaLM API with MakerSuite
and use it to access the API using a graphical user interface.
The suite includes a number of different tools, such as a model training tool, a model deployment
tool, and a model monitoring tool.
The model training tool helps developers train ML models on their data using different algorithms.
The model deployment tool helps developers deploy ML models to production with a number
of different deployment options.
The model monitoring tool helps developers monitor the performance of their ML models
in production using a dashboard and a number of different metrics.
That's all for now.
Thanks for watching this course, Introduction to Large Lanugage Models.
Required
English

Welcome to generative AI for everyone.
Since the release of ChatGPT, AI specifically,
generative AI has caught the attention of
many individuals, corporations, and governments.
It is a very disruptive technology that's

already changing how many people learn and work.
Many developers think that generative AI will
empower many people and lead to productivity gains,
and also make a significant contribution
to global economic growth.
But there could be downsides as well,
since it's job loss or
worse that some people worry about.
In this course, you learn what generative AI is.
What it can and cannot do,
and also how to use it in your own work or business.
Because generative AI is so new,
there is still a lot of misinformation out there,
and so in this course,
I hope to convey an accurate non-technical understanding
of what it really is.
It also work of you to think through
how you can best make use of this technology.
This course does not assume
any technical or AI background and
is designed to be useful to anyone in business,
science, engineering, the humanities, arts,
or other sectors so that, let's dive in.
Generative AI had caught the mainstream attention
starting around November 2022 when
OpenAI released ChatGPT and
its momentum has continued
unabated according to McKinsey,
it could add $2.6 to $4.4
trillion annually to the economy.
Goldman Sachs estimates it could raise
global GDP by 7% in the next decade,
and a study by OpenAI and
UPenn estimates that it could impact
10% of the work or tasks carried

out daily by over 80% of workers in the United States.
The same study also estimates
that there is a 20% of workers
whose work or task is
more than 50% impacted by generative AI,
and so studies like these lead
to hope for a tremendous productivity gain,
as well as worries about job loss through automation.
What is generative AI?
This term refers to AI,
or artificial intelligence systems
that can produce high-quality content,
specifically text, images and audio.
One of the best-known generative AI or GenAI systems
is OpenAI's ChatGPT,
which can follow instructions
to carry out tasks like write
three captions for a social media posts about
our new line of sunglasses for robots,
and have a generate a variety of creative outputs.
Many users are familiar with websites or
direct consumer applications that
can generate texts like this.
Other examples include Bard,
which is offered by Google,
as well as Bing Chat offered by Microsoft.
But there are now many companies that
are offering user interfaces that let
you type in some texts called
a prompt and will generate a response.
But beyond these consumer applications,
there's one other application of generative AI that I
think may turn out to be
even more impactful in the long term,
which is the use of generative AI as a developer tool.

AI is already pervasive in
our lives and many of us use it
dozens of times a day or
more without even thinking about it.
Every time you do a web search
on Google or Bing, that's AI.
Every time you use your credit card,
there's probably an AI checking if
it really is you using your credit cards or every
time you go to a website like Amazon or
Netflix and it recommends products
of movies to you, that's AI.
But many AI systems have
been complex and expensive to build,
and generative AI is making
many AI applications much easier to build.
This means that the number and variety
of AI product offerings is blossoming
because it's becoming much cheaper to build
certain AI applications compared to before.
In this course, one of the things
we'll touch on a few times as
well will be how generative AI may
allow your business to much more
inexpensively build very valuable AI applications,
and you learn best practices for identifying and
exploring whether or not such applications
might be useful for a given business.
I've described generative AI as generating text,
images and audio,
and of these three types of contents,
the biggest impact so far has been on text generation.
But it can also generate images where given instructions,
a different type of prompt.
It can generate beautiful images like this one,

or even a photo-realistic image like this.
Generative AI can also generate audio.
For example, here is a voice clone of me.
Hi, I'm an AI-generated voice clone of Andrew.
Andrew never actually said these words.
Isn't that cool? You can also put audio together with
image or even video generation
to create a video clone of me like this.
Hi, I'm a video clone of Andrew,
it's nice to meet you.
There's a lot going on in generative AI,
and this is an exciting and disruptive technology
that I'm confident you will find useful in your work.
In this course, during the first week,
we'll go through how generative AI technology actually
works and specifically what it can and cannot do,
and you also see a variety of
common use cases that I hope will help
spark your creativity to how you may use
it to create value in your life on your work.
In the second week,
we'll discuss generative AI projects.
Specifically how to identify
useful generative AI applications,
as well as best practices
for how to go about building them,
and we'll take a deeper dive into the range of
technology options for building
a variety of valuable projects.
In the final week, we will zoom out
beyond a single project to look at
how generative AI will impact
businesses as well as society at large.
We'll look at some best practices for
how teams where a company can take advantage of

generative AI and also take a look at AI risks and how to
make sure that what we do with
AI is responsible and benefits people.
I'm excited to dive into this material with you.
Until that let's go on to
the next video where we'll go through
a non-technical description of
how generative AI technology
actually works. I'll see you in the next video.
Required
English

Help Us Translate

The ability of systems like ChatGPT and
Bard to generate text seems almost magical.
They do represent a big step forward for AI technology.
But how does text generation actually work?
In this video, we'll take a look at what
actually underlies the generative AI technology,
and this will hopefully,
help you understand what you can use it
for and also when you might not want to count on it.
Let's take a look. Let's start by looking
at where generative AI fits within the AI landscape.
There's a lot of buzz and
excitement and also hype about AI,
and I think a useful way to think of AI
is as a collection or as a set of tools.
One of the most important tools in
AI is supervised learning,
which turns out to be really good at labeling things.
Don't worry if you don't know what this means,
we'll talk more about it on the next slide.
Second to it that started to work really well
only fairly recently is generative AI.
If you study AI,
you may recognize that there are other tools as well,
such as things called
unsupervised learning and reinforcement learning.
But for the purposes of this course,
I'm going to touch briefly on what is
supervised learning and then
spend most of our time talking about generative AI.
These two, supervised learning and generative AI,
are the two most important tools in AI today.

For most business use cases,
you should be fine if you just not worry
about the other tools than these for now.
Before describing how generative AI works,
let me briefly describe
what is supervised learning because it
turns out generative AI is
built using supervised learning.
Supervised learning is the technology that has
made computers very good when given an input,
which I'm going to call A,
to generate a corresponding output,
which I'm going to call B.
Look at a few examples.
Given an email,
supervised learning can decide
if that email is spam or not.
The input A is an email and
the output B is either zero or one,
where zero is not spam and one is spam.
This is how spam filters work today.
As a second example,
probably the most lucrative application,
not the most inspiring,
but lucrative for some companies that
I've ever worked on was online advertising,
where given an ad and some information about a user,
an AI system can generate an output B
corresponding to whether or
not you're likely to click on that ad.
By showing slightly more relevant ads,
this drives significant revenue
for the online ad platforms.
In self-driving cars and in driver assistance systems,
supervised learning is used to take as
input a picture of what's in front of
your car and radar info and
label that with the position of other cars.
Give it a medical x-ray,
it can try to label that with a medical diagnosis.
I've also done a lot of work in
manufacturing defect inspection where you can
have a system take a picture of a phone as it rolls off
the assembly line and check if
the phone has any scratches or other defects,
or in speech recognition,
the input A would be a piece of audio
and we would label that with the text transcript,
or as a final example,
if you run a restaurant or some other business where

occasionally you have reviews
written about your business or your products,
supervised learning can read those reviews and
label each one as having
either a positive or a negative sentiment.
This is useful for reputation monitoring of the business.
It turns out the decade of around
2010-2020 was a decade
of large-scale supervised learning.
I want to touch on this briefly
because it turns out this laid
the foundation for modern generative AI.
But what we found starting around
2010 was that for a lot of applications,
we had a lot of data,
but even as we fed it more data,
its performance wasn't getting that much
better if we were training small AI models.
This means, for example,
if you were building a speech recognition system,
even as your AI listened to
tens of thousands or
hundreds of thousands of hours of data,
that's a lot of data,
it didn't get that much more accurate compared to
a system that listened to
only a smaller amount of audio data.
But what more and more
researchers started to realize through
this period is if you were
to train a very large AI model,
meaning an AI model on very fast,
very powerful computers with a lot of memory,
then its performance as you fed it
more and more data will just
keep on getting better and better.
In fact, years ago when I
started and led the Google Brain team,
the primary mission that I set for
the Google Brain team in the early days was I said,
let's just build really, really large
AI models and feed them a lot of data.
And fortunately, that recipe worked and ended
up driving a lot of AI progress at Google.
Large-scale supervised learning remains important today,
but this idea of very large models for
labeling things is how we got to generative AI today.
Let's look at how generative AI generates
text using a technology called large language models.
Here's one way that large language models,

which I will abbreviate LLM, can generate text.
Given an input like,
I love eating, this is called a prompt,
an LLM can then complete
this sentence with maybe bagels with cream cheese,
or if you run it a second time, it might say,
my mother's meatloaf,
or if you run it a third time,
maybe it'll say out with friends.
How does an LLM,
a large language model, generate this output?
It turns out that LLMs are
built by using supervised learning.
That's a technology to input A and output a label B.
It uses supervised learning to
repeatedly predict what is the next word.
For example, if an AI system
has read on the Internet a sentence like,
my favorite food is a bagel with cream cheese,
then this one sentence will be turned into a lot of
data points for it to try
to learn to predict the next word.
Specifically, given this sentence,
we now have one data point that says,
given the phrase, my favorite food is a,
what do you think is the next word?
In this case, the right answer is bagel.
Also, given my favorite food is a bagel,
what do you think is the next word?
It's with, and so on.
This one sentence is turned into
multiple inputs A and outputs B for it to try to learn
from where the LLM is learning
given a few words to predict
what is the next word that comes after.
When you train a very large AI system on a lot of data,
a lot of data for LLMs
means hundreds of billions of words,
and in some cases,
more than a trillion words,
then you get a large language model
like ChatGPT that's given
a prompt is very good at generating
some additional words in response to that prompt.
For now, I'm omitting some technical details.
Specifically, next week,
we'll talk about a process that
makes LLMs not just predict the next word,
but actually learn to follow instructions and
also be safe in what it outputs.

But at the heart of LLMs is this technology that's
learned from a lot of data
to predict what is the next word.
That's how large language models work;
they're trained to repeatedly predict the next word.
It turns out that many people, perhaps including you,
are already finding these models useful for
day-to-day activities at work to help with writing,
to find basic information,
or to be a thought partner to help think things through.
Let's take a look at some examples in the next video.
There are quite a few web interfaces
where you can access a Large Language Model.
ChatGPT is the best known one and Google's Bard,
Microsoft Bing and quite a few others also work well.
Let's take a look at how people are
using these LLM applications.
Whether or not you're already using them regularly,
I hope that this will give you some new ideas
for when and how they could be useful to you.
LLMs are giving a new way to find information.
For example, if you ask
it what's the capital of South Africa,
it may give a response like that.
Now as we'll see later,
LLM can sometimes make facts up.
We call this hallucination..
If you're really relying on
getting the right answer the question,
it may be useful to double check the answer
with an authoritative source before counting on it.
But in this case, it does get
the capital or three capitals of South Africa right.
And sometimes a back and
forth with an LLM is helpful too,
for example, if you ask what does LLM stand for,
it might answer,
LLM stands for Legum Magister,
which is a term used in law,
and it's actually a pretty common use
of the acronym LLM on the Internet.
But if you were to then
say what about in the context of AI?
Then hopefully it'll say in the context of
AI and LLM refers to a Large Language Model.
Sometimes this back and forth can
help you give the right context,
the LLM to give you the information you're looking for.
LLM can also sometimes be
a thought partner to help you think things through.

For example, I often use an LLM to
help me refine my writing.
If you were to tell it, rewrite this for clarity,
since all around the world are
realizing learning has happened and so on.
Leading LLMs are actually pretty good
at rewriting texts for you.
Or here's a fun example,
if you were to tell it, to write
a three did word story involving trucks.
Maybe because you have a child that likes
trucks like I do, my son love trucks.
But to encourage them to brush the teeth,
then leading LLMs can actually
create pretty fun and interesting stories.
I don't think this is nearly as good
as the stories written by the great novelist,
but for a quick fun thing, I think it's not bad.
Now, there will be times when you're looking for
a piece of information and you might be wondering,
"Should I use web search or use an LLM?"
So If you're playing a sport and unfortunately
wound up with a sprained ankle
and want to know what to do about it,
web search can lead you
to pretty authoritative and I think
trustworthy sources that can give
advice on how to approach medical matters.
For example, web pages from
the Mayo Clinic or from Harvard Health.
These seem like they would be trustworthy sources
for what to do about the sprained ankle.
You could also ask an LLM what to do
about the sprained ankle and
it will generate some answer.
But given the propensity of
LLM's to make things up and sometimes
sound very authoritative and
confident when making things up,
I would probably want to double check anything it says
about healthcare or medicine
before following the suggestions.
Here's one more example.
If you want to make
a pineapple pie and they're looking for a recipe,
it turns out there are lots of recipes
on the Internet for a pineapple pie,
and picking one created by
a trusted website or a trusted chef,
that might get you pretty good results.

Or you can ask an LLM to
make one up for you, and in that case,
it'll come up with something that frankly might be okay,
but also has a high chance
of being a somewhat strange recipe.
So if you want to bake a pineapple pie,
I would probably go find a web page because there are
multiple web pages that will give
a good solid answer to
what's a good pineapple pie recipe.
But if you were to look for something more esoteric,
say your friend challenges you to make coffee,
infuse pineapple pie,
there aren't any web pages that I could
find really on coffee infuse pineapple pies.
I don't think there is currently
a single web page that gives a good answer to this.
This would be one example where
an LLM can help be a thought partner to think
through how you might go about
baking a coffee, infuse pineapple pie.
These are just some of the tasks for which you might find
the web user interface for LLM useful.
We'll explore more examples,
discuss strengths and weaknesses of LLMs,
and go through some best practices later this week.
But as you can see from this video,
Generative AI is capable of many different things.
In the next video, we'll more
systematically discuss Generative AI as
a general purpose technology as well as start to come
up with a way to
organize all of these things that they can do,
which includes writing, reading, and chatting tasks.
Let's go take a look at that in the next video.
What is Generative AI good for?
One of the reasons that question is a bit hard to
answer is because AI is a general-purpose technology.
Unlike a lot of technologies like a car which is good for
transportation or microwave oven good for heating food,
AI isn't useful just for one thing.
It's useful for a lot of things and that
almost makes it harder to talk about.
But let's take a look at what
a general purpose technology really means.
Similar to electricity, AI is useful for many tasks.
If we were to ask you, what is electricity good
for or what is the Internet good for?
These are other general-purpose technologies
and it's almost difficult to think what is

electricity good for because it's so pervasive
and it's used around us for so many different things.
In fact, as you saw earlier,
supervised learning is useful
for many tasks like spam filtering,
advertising, speech recognition, and many others.
Generative AI is like this too.
In the last video, you saw a few of
the tasks that an LLM can carry out,
answering certain questions and
hoping with writing for example.
Let's discuss more broadly a framework
for what tasks LLMs can do.
First off, Generative AI generates texts.
Not surprisingly, perhaps it's useful for writing.
I routinely use Generative AI tools
as a brainstorming companion.
If you're trying to name a product,
you can ask it to brainstorm some names
and it comes up with some creative suggestions.
LLMs can also be good at answering questions,
and if you give them access to
information specific to a company,
they can help members of your team
find information that they need.
In this case, about the availability
of parking at the office.
In addition to writing,
Generative AI is also
good for what I'm going to call reading task,
where you're going to give it a relatively long piece
of information and have it generate a short output.
For example, if you run
an online shopping e-commerce company
and you get a lot of different customer emails,
Generative AI can read the customer emails and help you
very quickly figure out is
this email a complaint or not,
which can be used for helping to
route complaints to the appropriate department
to be handled quickly.
Given, I love my new llama t-shirt,
the fabric is so soft, that's not a complaint.
But if someone emails,
I wore my Ilama t-shirt to
friend's wedding and now they're mad
at me for stealing the show.
Well, maybe that is a complaint.
But Generative AI can help you
route emails to the right departments.

I call this a reading task because
it's looking at a relatively long piece of text,
that is a customer email,
and then generating a relatively short output,
just yes or no, is this a complaint or not?
While supervised learning can
also be used for this particular task,
we'll see later that Generative AI
is allowing this source of reading tasks
and other examples that we'll see later this week to
be built much more quickly and inexpensively.
Lastly, Generative AI is also
used for many chatbot types of tasks.
Whereas ChatGPT, and Bard,
and Bing chat are general-purpose chatbots.
Generative AI technology and
large language models is also
enabling many special-purpose chat bots to be built.
In this example, here's what a chatbot might be like
for taking online orders where a user can say,
I like a cheeseburger for delivery and the chatbot
acknowledges and puts the order through for the user.
Now in talking through these tasks,
I find that it's sometimes useful to distinguish between
two different types of LLM-based applications.
One is examples like this brainstorming one,
where it could be quite natural for you to type
a prompt like this into ChatGPT,
or Bard, or Bing chat,
or one of the other free or paid
large language models on
the Internet and get a result back.
I'm going to call an application like
this a web interface-based application.
In contrast, in the example of
recognizing if an email is a customer complaint,
this fits more into a company's email routing workflow,
and it doesn't really make sense for anyone
to cut and paste customer emails one at
a time into a web interface to get
back answers as to which
ones are actually complaint emails.
This is an example of
an LLM that would make sense when it's
built into a larger software automation,
that in this case helps with
a company's automated email routing.
I'm going to call this a LLM-based software application,
the second writing example of answering HR questions.
It turns out this also will make

more sense as a software-based LLM application
because it'll need access to information
about your specific company's
parking policy for employees,
whereas a general large language model on
the Internet probably doesn't have that information.
We'll talk more later in this course about
how this technology is built and most
of the specialized chatbots will also
be software-based LLM applications.
In the rest of this course,
I'm going to use these two symbols to distinguish between
web interface use cases
and software-based LLM applications.
For many people, it may be
easier to get started with some of
the web interface use cases
because you can just go to a website like ChatGPT,
or Bard, or Bing and type in a prompt,
and get the result back.
But I think both the web interface-based applications and
the software-based LLM applications are
important and will be very
useful for individuals and for companies.
I found that the framework of writing, reading,
and chatting as a useful way to think about
the many different tasks that
LLM a large language model can do.
In the next three videos,
we'll dive more deeply into
many different examples of writing,
reading, and chatting tasks.
I hope that you'll find some of them
potentially useful for your own work.
I look forward to seeing you in the next video,
we'll talk more about writing,
and until then, I look forward to enjoy my burger.
In the last video, we looked at writing, reading and
chatting as three major categories of tasks that you can get an LLM to do.
Given that large language models were trained to repeatedly predict
the next word,
maybe it's no surprise that they're pretty good at writing at generating words.
And it turns out that many writing tasks can be done via a web user interface.
So I hope you find this video where we'll dive more into writing tasks
immediately useful.
For writing tasks broadly, what we typically do is start with prompts and
use a relatively short prompt to write or to generate a much longer piece of text.
So let's take a look at some writing applications.
I often use the web interface of large language models as a brainstorming
partner.

If you ask it brainstorm 5 creative names for peanut butter cookies, it actually
comes up with some pretty creative names Nutty Nirvana Nibbles, I would eat that.
Or if you ask it to brainstorm ideas for increasing cookie sales, then it comes
up with a few ideas and you can take a look to see if any of these may be useful.
You can also use a large language model, again,
maybe the web interface version, to write some copy for you.
Let's start with an example.
If you were to ask it to write a press release announcing the hire of a new COO,
a new chief operating officer for your company, it may come up with a piece of
text like this Company Name Welcomes, New COO's Full Name as, so on and so
forth.
And this is a pretty generic press release.
When writing a prompt, you find that if you can give the large
language model more context or more background information,
then it will write more specific and better copy for you.
If all that the large language model sees is this writer a press release,
at this point in time, it doesn't know anything about your company,
about the new CEO's name or their qualifications and so
it ends up writing something very generic like this.
If you end up prompting a large action model like this, it's not a problem.
You may realize that you wound up with a very generic press release and
decide to update the prompt to give it more information.
And so if you were to prompt it and say, use the following information for
the press release, this is a COO bio, this is the name of our company and
some details about our company, then it will write a much more detailed and
insightful press release specific to the CEOs joining this company.
I find that when prompting an LLM, I'll often not get the prompt right the first
time, like what we saw just now, where we had the prompt press release
announcing the new hire of COO without giving any context.
And that's totally fine.
If you see the result isn't what you want, just revise the prompt and try again.
I'll say more about this in a later video this week,
when we talk about tips for writing effective prompts.
Let's look at one more example.
Another writing task that I sometimes use LLMs for is translation.
In fact, some of the large language models you can access via Web UI are
competitive
and sometimes even better than the dedicated machine translation engines
already, especially for languages with a lot of text on the internet.
And so where the large language model has a lot of data to learn how to generate
text in that particular language, it tends to do less well in languages, also
called low resource languages with less text on the Internet in that language.
But if you're operating a hotel and
you want to translate the welcome message into formal Hindi to welcome guests,
then a large language model may be able to output text like this for you.
Unfortunately, I don't speak Hindi, I wish I did, but it turns out that this
particular translation is only so, so the word front desk,
it translates into the desk at the front rather than the reception,
which is what we mean when we say the front desk of a hotel.
So if you're working with a Hindi speaker, and I was when preparing the slide,

then they may be able to give you some tips to say,
this is some sort of not quite the best formal Hindi, but
you were to tell it to translate this into formal spoken Hindi.
Then it updates this text to make front desk translate into the Hindi word for reception, which is a much better translation.
Now, here's one fun thing I've seen recently in the AI community,
which is a lot of us that are working with translation often need to translate text into languages that we don't speak ourselves.
So how can we tell if the large language model is doing something reasonable?
And in fact, even if you have, say, one Hindi speaker on your team, if other members of the team don't speak Hindi, how can they figure out what's going on?
So what I'm seeing multiple teams in the AI community do
is translate text into pirate English for testing purposes.
And so if you were to prompt a large language model to translate this
into pirate English, you get, ahoy matey,
we be hoping you'll relish your time aboard the Oceanview Inn.
That sounds like pretty good pirate English to me.
So that hobby grand worthy models be used for writing.
Let's move on to peepin at Reading House.

In the last video, we looked at writing tasks

where you would specify

a prompt to the last large language model,

and have it generate a comparatively longer output

than the input prompt.

It turns out, is also useful for many reading tasks,

and by that I mean tasks where you would input a prompt,

and then have it generate usually

a similar length or

often shorter output than the input prompt.

Let's take a look at some reading tasks

starting with something that I use myself all the time,

which is proof reading.

Many times if I'm writing a piece of text,

I will read through it carefully three or four

times myself for spelling and grammatical errors,

and even though I thought I

prove read it carefully myself,

a line model will still find

errors in it that somehow I had miss.

Here's an example of a prompts that you could try.

Proofread the following text,

and I find that if you tell
it what you want the text before.
Here's text intended for
a website selling children's stuffed toys,
and sometimes I ask you to check for spelling and
grammatical errors as well as awkward sentences,
and then have it rewrite it with corrections.
This is a piece of text with some errors,
and the output of the large language model fixes,
snuggle was misspelled and it
fixes this little piece of grammar over here.
When I'm writing text myself,
that I want to be quite confident it's
free of spelling and grammatical errors,
and sometimes also awkward sentences,
I actually use this myself to proof read what I write.
A second reading task
that large language models are often used for,
is to summarize long articles.
One of my collaborators,
Erik Brynjolfsson, who's a Stanford professor,
once sent me an email linking to an article
that he had written titled The Turing Trap.
I knew it was a good article,
but it was a very long article and I didn't have
time to read the whole thing
before I responded to his email.
I actually use the following prompt and copy pasted
his entire article into
an web interface of a large language model,
and had it quickly generate a summary for me.
It turns out this paper that he had written talks
about how human-like AI offers benefits,
but there's a lot to be done by having
AI augment humans rather than automate.

But the point of Brynjolfsson's article
on the Turing Trap was he was
advocating that instead of having
AI automate or replace human work,
we should put more effort into having
AI complement or to augment human work.
With a large language model
summarizing this long article,
I was able to get back on this
faster than if I had to read the entire article myself.
By the way, this is a good article.
Eventually I did read the entire article myself,
and I really enjoyed it, but today,
I do sometimes use large language models to summarize for
me things that I don't have time to read in its entirety.
This is a use case that you could go to one of
the web interfaces of a large language model
and use relatively quickly yourself.
Now it turns out there's
a software application version of
this too that is taking off in businesses.
Let me illustrate this with an example.
Say you're a manager of
a customer service call center
where you have many customer service agents,
like this person shown on the left with the microphone,
having phone calls with customers,
like this person shown on the right.
If you have permission to record
these phone calls between the agents and the customers,
you can then run the phone calls through
a speech recognition system to get
a text transcript of the conversation.
If you have many customer
service agents having conversations,

you end up with a lot of text transcripts.
If you want to review
what's going on in your call center,
you probably end up with too much texts to read.
Given a text transcript like
this between the customer and an agent,
what really happened in this call?
One use of large language models would be to have it
summarize this entire conversation
and generate a short summary.
Like MP401-27KX was reported as broken, and so on.
If you were to take all
of these different text transcripts
and have a software application
to generate the summaries,
then you as a manager of this can take a quick look
at all of these summaries and just
spot if there are any issues,
or any trends that you want to be aware of.
A system like this would be implemented as
a software application that uses a large language model,
because it doesn't really make sense for
you or anyone else to copy paste
these conversations one at a time into
the website of a large language model provider.
In terms of customer service interactions,
large language models are also
used for customer email analysis.
In an earlier video, you saw
the example of taking a customer email,
and deciding if it's a complaint, in this case,
no, as well as what department to route this email.
This will be another software application
that uses a large language model.
Let's take a deeper look at

how one could build this application,
focusing on the part of
deciding what department to route this email.
One thing you could do is write a prompt to tell the LLM
to read the email and decide
which department to route it to.
You can specify the task and provide the email.
But it turns out that with a prompt like this,
you may find that the algorithm
routes it to the complaints department in this case,
which may or may not be
a department that exists in your organization.
This would be an example of where the LLM has been given
insufficient context to know what are the names
of the actual departments in
your company that it should choose from.
In contrast, if you were to
update the prompt as follows, and say,
read the email,
choose the most appropriate department to rouse it to,
and choose department only from the following list.
In this case, given
the set of choices you wanted to choose from,
routes it to the apparel department correctly.
The process of building an application using
a large language model is again, not a lot uncommon.
To write a prompt that doesn't
quite work right the first time,
and when you find it,
route it to a nonexistent complaint department,
then just update the prompt and that fixes the problem.
One last application that I want to touch
on is reputation monitoring,
where you can use an LLM to build
a dashboard to track your customer sentiments,

positive or negative of
your business or your products over time.
For example, if you run
a restaurant and occasionally your customers,
write online reviews or send
your emails describing their experience,
you can then use a prompt like this,
read the following review and
classified as having the positive,
negative sentiment, to have it decide
automatically if each review was positive or negative.
In this case, if the food was
amazing or service a friendly,
that would be classified as having a positive sentiment.
Then by having software
count the number of positive reviews per day,
as well as the number of negative reviews per day,
you can build a dashboard that tracks per day,
all the time, how the sentiments are trending.
Looks like the customer sentiment is pretty positive,
but if ever it starts trending negative like this,
with more negative reviews,
then this dashboard can alert you to that
maybe something's happening that
we should pay attention to,
and see if there's something
we need to fix at the restaurant.
In this video, we looked at
a number of reading applications,
including proof reading, summarization,
email routing, restaurant review, sentiment analysis.
If you can think of toss where you wish you
had someone that could read a piece of text,
and just say a few things or give
a few quick indications of

what was in that piece of text,
that could be a good candidate for
a reading task to get an LLM to do for you.
Next, let's go onto the next video
to take a look at chatting task.
Required
English

Help Us Translate

In the last two videos,
we looked at writing and reading applications.
In this video, we'll look at chatting applications.
In addition to the general
purpose chat bots like ChatGPT,
Bard, and Bing chat.
Many companies are looking at whether they
can build specialized chat applications.
If you're involved in a company where
you have many people interacting
with customers or having
certain types of conversations of similar nature,
this may be a case where you can consider whether or not
a specialized Chatbot can
help with those types of conversations.
Let's take a look. Earlier we already saw the example
of a customer service Chatbot that might
better take orders for cheeseburger.
Another example of a specialized Chatbot would be
one that specializes in helping you to plan trips.
How can vacation in Paris inexpensively?
A bot could be built to have
specialized knowledge about travel.
Today, there are companies exploring
a wide range of advice bots.
For example, can a bot give you
career coaching advice or give advice on cooking a meal?

A large variety of specialized bots that are really good
at answering questions about one thing
are being developed by different companies today.
Some of these bots are capable of just
having a conversation and giving advice.
Some of these bots can also interface with the rest of
a company's software system and take actions such
as to put in an order for a cheeseburger to be delivered.
Another example of a bot that might be
able to take action would be a customer service chat bot.
Where it turns out that
many IT departments get tons of postware reset requests.
If a bot can take care of that,
then it may take some of
the workload off your IT department.
A bot like this that needs to be send
a text message to verify
identity and actually help reset the postware.
This is a bot that would need to be empowered,
to actually take action in the world
such as to get a text message to be sent to someone.
Next week we'll discuss more how
Chatbots like these are
builds that don't just generate text.
But we can actually take action.
Because of the number of
customer service organizations
exploring the use of Chatbot,
I want to share with you a range of the spectrum of
common design points being used by different businesses.
For this slide, I want to focus on
text based chat rather than voice or phone based chat.
At one end of the spectrum would be
a customer service center with only humans.
You would have human service agents

typing back and forth messages like,
welcome to Better Burgers
and let me play the order for you.
At the opposite end of the spectrum would be Chatbots
only where you just have
software responding directly to customers.
But between these two ends of the spectrum
of humans typing of the keyboard or Chatbots only,
there are a couple common design points.
One common design points would be to
have bots support humans.
In which a bot will generate
a suggested message for a human but
the human service agent will reopen
the message and either approve it if it looks good.
Or have a chance to edit the message before
it is actually sent back to the customer.
This type of design is
often also called human in the loop.
Because as a human,
there's lot in and it's part of the process before
the message actually gets sent back to your customer.
This is a way to mitigate the risk
of the Chatbot maybe saying the wrong thing.
Because a human can check over it before
it's actually sent back to your customer.
In the next video,
when we talk about what LLMs can and cannot do.
We'll go over some of
the mistakes that LLMs can sometimes make.
This design helps protect against those mistakes of LLMs.
A little bit further on the automation spectrum
would be if you have a bot to treach messages for humans.
Maybe the bot answer the easy messages but
escalate to a human for

the things that isn't quite ready to handle yet.
Sometime back, I actually let
the team that build a bot that would
automatically detect if the customer
was asking for a refund request.
It turns out that was about 10% of
our total chat call volume.
By just detecting that and automatically giving
the customer instructions just routed 10% or
so of the traffic away from
the human agents and so to save
the agents a lot of time and let
the humans focus on servicing the harder requests.
But this type of triagging is another common design to help
your human service agents save
time and have to focus only on the harder cases.
That they're more uniquely qualified to handle.
In many customer service centers,
a single human may be
simultaneously having chat conversations
with four or eight,
or in some extreme cases,
maybe in 16 customers at the same time.
With bots supporting the humans,
it becomes easier for a human to manage
a larger number of parallel conversations.
Given that bots sometimes say the wrong thing,
I want to share with you what
building and deploying a bot
often feels like in companies
that want to do this in a safe way.
Often, companies will start with
an internal facing Chatbot.
Many times I would build a Chatbot,
but let only my own team use it to say,

answer the questions about travel or
whatever the bot is supposed to do.
Assuming your internal team will be
more sympathetic and more understanding of mistakes and
be more forgiving if the bot
says something wrong at one time,
this gives you some time to
assess the behavior of the bot and
also avoid public mistakes
that could be embarrassing for the company.
After this looks safe enough,
a common next step would
be to deploy with human in the loop.
To let the human check over many of the messages.
If feasible, before it actually goes out to the customer.
After doing this for a while if it looks like
the bots messages are
generally safe to send to customers,
then you might allow the bot to
communicate directly with customers.
Of course, the details of every business differs.
For some applications, it may not be practical to
have humans check over
every message because of the sheer volume of traffic.
But depending on the risk of the bot saying
the wrong thing as well as the volume of traffic,
and thus whether or not human in the loop is feasible,
these are some of the design patterns I've seen
companies use to try to deploy bots safely.
To summarize, we've seen how LLMs can
be used for writing, reading, and chatting.
These three categories are not meant to be
an exhaustive list of what LLMs can do,
but they are just a few broad categories
of what you might really use them for.

LLMs can do a lot,

but they can't do everything.

In the next video, let's take a look at what LLMs

can and cannot do and better understand the limitations.

Let's go on to the next video.
Required
English

Help Us Translate

Generative AI is an amazing technology, but it can't do everything.
In this video, we'll take a careful look at what LLMs can and cannot do.
We'll start off with what I found to be a useful mental model for what it can do,
and after that, let's look together at some specific limitations of LLMs.
I found that understanding the limitations can lower the chance that you might get tripped up trying to use them for something that they're really not good at,
so that let's dive in.
If you're trying to figure out what prompting an LLM can do,
here's one question that I found to provide a useful mental framework.
Which is I'll ask myself, can a fresh college grad,
following only the instructions in the prompts, complete the tasks you want?
For example, can a fresh college grad follow instructions to read
an email to determine if an email is a complaint?
Well, I think a fresh college grad could probably do that, and
LLM could do that pretty well too.
Or can a fresh college grad read a restaurant review to determine if it's
a positive or negative sentiment?
I think they could do that quite well too, and so too, can prompting an LLM.
Here's another example, can a fresh college grad write a press release
without any information about the COO or your company?
Well, this fresh college grad just graduated from college.
They only just met you and don't know anything about you or
your business, and so the best they could do is maybe write a really generic and not quite satisfying press release like this.
But on the flip side, if you were to give them more context about your business and about the COO, then we can ask,
can this fresh college grad write a press release given the basic relevant context?
And I think they may be able to do that decently well, and so too,
can the large language model.
When you're picturing an LLM as doing many of the things that a fresh college grad might be able to do, think of this fresh college grad as having lots of background knowledge that they know, lots of general knowledge off the Internet.
But they have to complete this task without access to a web search engine, and they don't know anything about you or your business.
For clarity, this mental model thought experiment, fresh college grad
has to complete a task with no training specific to a company or your business.
And every time you prompt your LLM,
the LLM does not actually remember earlier conversations.

And so it's as if you're getting a different fresh college drag for
every single task, so you don't get to train them up over time on the specifics
of your business or the style you want them to write.
This rule of thumb of asking what a fresh college grad can do is an imperfect rule
of thumb, there are things college grads can do that LLMs cannot and vice versa.
But I found this to be a useful starting point for
thinking through what LLMs can and cannot do.
And while we're focused on this slide on what prompting an LLM can do,
next week when we talk about generative AI projects,
we'll talk about some slightly more powerful techniques that might be able to
expand what you can do with generative AI beyond this fresh college grad concept.
Now, let's take a look at some further specific limitations of LLMs.
First, is knowledge cutoffs.
An LLM''s knowledge of the world is frozen at the time of his training.
More precisely, a model trained on Internet data scraped by January 2022,
will have no information about more recent events.
So given such a model, if you were to ask it,
what was the highest grossing film of the year 2022?
It would say it doesn't know.
Even though now that we're well past 2022, we know that it was the movie Avatar,
The Way of Water that was the highest grossing film.
Around July 2023, there were claims of research lab having
discovered a room temperature superconductor called LK-99.
You may have seen this picture in some of the news,
this claim turned out not quite to be right.
But if you were to ask an LLM about LK-99,
even though it's widely covered in the news, if the LLM learned only from
text on the Internet as of January 2022, it won't know anything about this.
So this is called a knowledge cutoff,
where the LLM knows things about the world only up to a certain moment in time.
When it was trained, or when text from the Internet was last downloaded for
the LLM's training.
A second limitation of LLMs is that they will sometimes just make things up,
and we call these hallucinations.
I found that if I ask an LLM to give me some quotes from well-known people in
histories, it will often make up the quotes.
For example, if you ask it,
give me three quotes that Shakespeare wrote about Beyonce.
Since Shakespeare lived and died well before Beyonce,
I don't think Shakespeare said anything about Beyonce.
But LLM will confidently give you back some quotes like her vocals shine like
the sun, or all hall the queen, she is most worthy of love.
So these are hallucinated Shakespearean quotes.
Or if you ask it to list court cases tried in California about AI,
it might give authoritative sounding answers like this.
And in this case, it turns out the first case is real,
there was a Waymo versus Uber case, but I was not able to find an Ingersoll
versus Chevron case, and so the second case is a hallucination.
Sometimes LLMs can hallucinate things or make things up in a very confident,
authoritative, sounding tone.

And this can mislead people into thinking that this made-up thing may
actually be real.
Hallucinations can have serious consequences.
There was a lawyer that unfortunately, used ChatGPT to generate text for
a legal case and actually submitted to the court, not knowing that he was
submitting to the court illegal fouling with lots of made-up court cases.
And in this New York Times headline, we see in this cringe inducing court hearing.
The lawyer who relied on AI said she did not comprehend that the chat bot
could lead him astray, and this particular lawyer was sanctioned for
submitting a co-filing for made-up things.
So understanding of limitations is important if you are using this for
documents of real consequence.
LLMs also have a technical limitation in that the input length, that is, the length
of the prompt is limited, and so is the output length of the text it can generate.
Many LLMs can accept a prompt of up to only a few thousand words, and so
the total amount of context you can give it is limited.
So if you were asking it to summarize a paper, and
the paper's length is much longer than this input length limitation,
the LLM may refuse to process that input.
In this case, you may have to give it one part of the paper at a time, and
ask it to summarize parts of the paper at a time.
Or sometimes you can also find an LLM with a longer input limit length,
some will go up to many tens of thousands of words.
And technically, LLM's have a limitation on what's called the context length,
and the context length is actually a limit on the total input+output size.
When I use LLMs, I rarely have it generate so
much output that I run into limitation really on the output length.
But I do hit input length limits sometimes if I have many,
many thousands of words of context that I want to give it.
Lastly, one major limitation of generative AI is that they
do not currently work well with structured data.
And by structured data I mean tabular data,
like sort of data that you might store in an Excel, or Google Sheets, spreadsheet.
For example, here is a table of home prices with data on both the size of
the house in square feet, as well as the price of the house.
If you were to feed all of these numbers into an LLM and then ask it,
I have a house that's 1,000 square feet what do you think is a good price?
LLMs are not really good at that, instead, if you call the size the input A,
and the price the output B, then supervised learning would be a better
technique with which to estimate the price as a function of the size.
Here's another example of structured data of tabular data showing when different
visitors may be visiting your website, how much you offered a product to them, and
whether or not they purchased it.
Then again, supervised learning would be a better technique than trying to copy
paste all of this time, and price, and
purchase information into the prompt of a large language model.
In contrast, to structured data,
generative AI tends to work best with unstructured data.
Structured data refers to tabular data of the soil you would store in a spreadsheet,
whereas unstructured data refers to text, images, audio, video.

And generative AI does apply to all of these types of data, although the impact
is the largest and that's why we'll focus mostly on text data in this course.
Finally, large language models can bias output and
can sometimes output toxic or other harmful speech.
For example, large language models were trained on text off the Internet.
And unfortunately, text on the Internet can reflect biases that exist in society.
So if you were to ask an LLM complete the sentence, surgeon walked to the parking lot and took out, the LLM might output his car keys, but
you'll say the nurse walked to the parking lot and took out, it may say her phone.
So in this case, the LLM has assumed that the surgeon is male, and the nurse
is female, whereas we know that clearly surgeons and nurses can be any gender.
And so if you're using an LLM in an application where such
biases could cause harm, I would use care in how we prompt and
apply the LLM to make sure we don't contribute to such undesirable biases.
Finally, some LLMs can also occasionally output toxic or other harmful speech.
For example, some LLMs will sometimes teach people how to do undesirable, sometimes even illegal acts.
Fortunately, all the major large language providers have been working hard
on the safety of these models, and so most models have gotten much safer over time.
And if you use the web interfaces so the major LLM providers, it's actually been
getting much harder over time to get them to output these types of harmful speech.
So that summarizes what prompting an LLM can and cannot do.
And as I mentioned, next week we'll take a look at some techniques for
overcoming some of these limitations to make what LLMs can do even broader and more powerful.
But first, let's take a look at some tips on prompting LLMs.
And I hope that the tips I share in the next video will be useful right away to
how you use LLMs, I'll see you in the next video.

I'd like to share with you some tips for prompting large language models.

If you're using the web user interface of a LLM provider,

hopefully these tips will be useful to you right away.

And it turns out that similar tips are also useful for if you're ever involved in

building a software application that uses LLMs, let's dive in.

In this video, we'll go through three main tips for prompting.

First is detail and specific.

Second is guide the model to think through its answer.

And third is experiment and Iterate.

This starts with be detailed and specific.

Using the Fresh College Grad analogy,

I would often think about how to make sure the om has sufficient context or

sufficient background information to complete the task.

So for example, if you were to ask it help me write an email asking to be assigned to the Legal Documents project.

Well, given only a prompt like this, an om doesn't really know how to
write a compelling case to advocate for you to be assigned to that project.
But if you give additional context such as I've applied for
a job in the Legal Documents project.
We check legal documents I have for
experience I'm prompting LLM's to get accurate text or professional tone.
Then this gives the LLM more relevant context to write that email to help
you ask to be assigned to the project.
Further, if you can also describe the desired task in detail.
So if you tell it instead of saying help me write in the email,
if you ask it, write a paragraph of text explaining why my background makes
me a strong candidate on this project, an applicant of the candidacy.
Then this type of prompt would not only give the LLM's sufficient context, but
also tell it quite clearly what you wanted to do, and
this is more likely to get you the result that you want.
Second tip is to guide the model to think through its answer.
So if you were to tell it brainstorm 5 names for a new cat toy,
it actually could do pretty well.
But if, say, you have in mind you want a rhyming cat toy name with
a relevant emoji, this is what I might try.
I might tell it brainstorm 5 names and tell it step 1,
come up with five joyful words already to cats.
Then for each word, come with a rhyming name.
And finally, for each toy name, add a fun relevant emoji.
And with a prompt like this you might get a result like this where the LLM follows
your instructions to first come with purr, whisker and so on.
And then purr-twiri, whisker-whisper,
feline-beeline with fun emojis added to the end.
So if you already have in mind a process by which you think the arm could get to
the answer that you want,
giving it clear step-by-step instructions to follow could be quite effective.
Finally, there have been a bunch of articles that are seen on social media
that say things like 20 prompts that everyone must know or
17 prompts that will help you grow your career.
I don't think there's a perfect prompt for everyone.

Instead, I find it more useful to have a process by which you can
write the prompt that will generate the result for you.
So when I'm prompting an LLM myself, I will often experiment and
iterate and try something like my solid off say, help me rewrite this.
And if I don't like the result I might clarify and
I say correct any grammatical and spelling errors in this.
And if it still doesn't give me exactly result I want,
I might clarify even further to say, correct any grammatical and
spelling errors and rewrite in the tone appropriate for a professional resume.
So very frequently the process of prompting is not about starting off with
the right prompt.
It's about starting off with something and
then seeing if the results are satisfactory and
knowing how to adjust the prompt to get it closer to the answer that you want.
I think of the process of prompting as like this, you start off with an idea of
what you want the LLM to do and you just express that in a prompt.
And then based on the prompt, the LLM will give a response and it may or
may not be what you want.
If it is, then great, you're done.
But if it isn't satisfactory,
then that initial response helps you shape your idea and modify the prompt and
iterate maybe a few times before you get to the result that you want.
So I think of the prompting process as when I start off,
I try to be reasonably clear and specific.
But to save time, I'll often start off with a short prompt that maybe is frankly
less specific than it's desired, but I just want to get going quickly.
After you get a result back, if it's not what you want,
then think about why the result isn't the desired output.
And based on that, refine your prompt to clarify your instructions and
keep on repeating until hopefully you get the LLM response that you want.
One tip I want to share is I've seen some people overthink the initial prompt.
I think it's better to usually just try something quickly and if it doesn't give
you the results you want, it's fine, go ahead and improve it over time.
You will not break the Internet by just accidentally having a slightly incorrectly
worded prompt.

So go ahead and try what you want.
Two important caveats, first,
if you are in possession of highly confidential information.
I would make sure I understand how a large language model provider does or does not use or keep that information confidential.
Before copy pasting highly confidential information into the Web user interface of an LLM.
And second, as we saw in the last video with the lawyer,
they got into trouble submitting court filings with facts made up by an LLM.
Before you counter the LLM's result, it may be worth double checking and deciding for yourself whether or not you can trust and act on the LLM's output.
But with these two caveats when prompting, I will often just jump in and try something and see it not work, but then use the initial result to decide how to refine the prompt to get a better result.
And that's why we say prompting is a highly iterative process.
Sometimes you have to try a few things before you get the result you want.
So that's it.
For tips on prompting, I hope that you go to some of the web user interfaces of the large language model providers and try out some of these ideas yourself.
And in this course, we provide some links to some of the popular LLM providers and hope you go play with them and have fun with them.
That brings us to the end of the main set of videos for this week.
There's one optional video to follow where I'll talk a little bit about image generation or diffusion models.
So take a look at that if you want, and
then I look forward to seeing you back next week
where we'll talk more about how to build projects using large language models.
Look forward to diving into that with you next week.
Required
English

Help Us Translate

Thanks for sticking with me for
this final optional video on image generation.
So far this week we'll focus most
of attention on text generation.

Text generation is what a lot of users are using and is
having the biggest impact of
all the different tools of generative AI.
But part of the excitement of
generative AI is also image generation.
They're also starting to be some models
that can generate either text or images,
and these are sometimes called multimodal models,
because it can operate in multiple modalities,
text, or images.
What I'd like to do in this video is share with you
how image generation works. Let's take a look.
With just a prompt,
you can use generative AI to generate
a beautiful picture of a person that had never existed,
or a picture of a futuristic scene,
or a picture of a cool robot like this.
How does this technology work?
Image generation today is mostly
done via a method called a diffusion model.
Diffusion models have learned from
huge numbers of images
found on the Internet or elsewhere.
It turns out that at the heart of
a diffusion model is
supervised learning. Here's what it does.
Let's say the algorithm finds a picture on
the Internet of an apple like this,
and it wants to learn from pictures like this
and hundreds of millions of
others how to generate images.
The first step is to take this image,
and gradually add more and more noise to it.
You go from this nice picture of an apple, to a noisier,
to an even noisier,

to finally a picture that looks like pure noise.
Where all the pixels are chosen at
random and it doesn't look at all like an apple.
The diffusion model then uses pictures like these
as data to learn using supervised learning,
to take as input,
a noisy image and output a slightly less noisy image.
Specifically, it would create a dataset where,
the first data point says if
it's given the second input image,
what we want the supervised learning algorithm to do
is learn to output a cleaner version of this apple.
Here's another data point,
given this third image of an even noisier image,
we would like the algorithm to learn to
output a slightly less noisy version like this.
Finally, given an image of
pure noise like this fourth image,
we would like it to learn to output
a slightly less noisy picture
here that suggests the presence of an apple.
After training on maybe hundreds of millions of images,
viral process like this,
when you want to apply it to generate a new image,
this is how you would run it.
You will start off with a pure noise image.
Start by taking a picture where,
every single pixel in the picture is
chosen completely at random.
We then feed this picture to
the supervised learning algorithm
that we trained up on the previous line.
When we feed in pure noise,
it learns to remove a little bit
of noise from this picture,

and you may end up with a picture like this
that suggests some sort of fruit in the middle,
but we're not quite sure what it is yet.
Given the second picture, we again feed it to the model,
and it then takes away even a little bit more noise,
and now it looks like we can
see a noisy picture of a watermelon.
Then if you apply this one more time,
we end up with this fourth image,
which looks like a pretty nice picture of a watermelon.
I'm illustrating this process using
four steps of adding noise on the previous slide,
and four steps of removing noise on this slide.
But in practice, maybe about 100 steps
will be more typical for a diffusion model.
This algorithm will work for
generating pictures completely at random.
But we want to be able to
control the image it generates by
specifying a prompt to tell
it what we want it to generate.
Let me describe a modification of
the algorithm that lets you add text,
or add a prompt to tell it what you want it to generate.
In this training data,
we're given pictures like this apple,
as well as a description or
a prompt that could have generated this apple.
Here, I have a text description
saying this is a red apple.
Then we will same as before,
add noise to this picture until we get the fourth image,
which is pure noise.
But we're going to change how we
build the learning algorithm, which is,

rather than inputting the slightly noisy picture
and expecting it to generate a clean picture,
we'll instead have the input A,
to the supervised learning algorithm
B, this noisy picture,
as well as the text caption or
the prompt that could have generated
this picture, namely red apple.
Given this input, we want the algorithm
to output this clean picture of an apple.
Similarly, we'll generate additional data points for
the algorithm using the other noisy images.
Where each time, given a noisy image,
and the text prompt red apple,
we want the algorithm to learn to generate
a less noisy picture of a red apple.
Having learned from a large dataset,
when you want to apply this algorithm to generating,
say, a green banana, this is what you do.
Same as before, we start off
with an image of pure noise.
Every single pixel is chosen completely at random.
If you wanted to generate a green banana,
you input to the supervised learning algorithm,
that picture of pure noise together
with the prompt, "green banana".
Now that it knows you want a green banana,
hopefully the ovum will output
a picture that maybe looks like this.
Can't see the banana that clearly,
but maybe there's a suggestion of
some greenish fruit in the middle,
and this is the first step of image generation.
The next step is, we then take this image on the right,
there was an output B,

and feed that is the input A, with again,
the prompt "Green banana" to get
it to generate a slightly less noisy picture,
and now we see clearly,
looks like there's a green banana,
but a pretty noisy one.
We do this one more time and
it finally removes most of the noise,
until we end up with that picture
of a pretty nice green banana.
That's how diffusion models work for generating images.
At the heart of this magical process of
generating beautiful images is,
again, supervised learning.
Thanks for sticking with me for this optional video,
and I look forward to seeing you next week where,
we'll dive much more into
applications being built using generative AI.
I'll see you in the next video.
Required
English

Help Us Translate

Welcome back. Last week we
discussed how generative AI can be used either
via a web user interface or
be built into a software application.
This week we'll take a look at
how many amazing software applications
are being built using generative AI,
and we'll also take a look at
some technology options that go beyond
just prompting and that allow
you to do much more with generative AI,
for example, having it operate
on your own proprietary documents
rather than just on what is learned
from public sources on the Internet.
Let's take a look, we saw last week
a few examples of generative AI applications,
such as writing answers to questions that may require

access to information about
your company's parking policy in this example.
Or reading restaurant reviews
on the Internet to help with
reputation monitoring or building
a chat bot to help take food orders.
It turns out that while some applications like this
did exist and were
built before the rise of generative AI.
Generative AI has made building these applications much
easier and in many cases
has made them work much better as well.
Let me illustrate with the example of
reading restaurant reviews for reputation monitoring.
A few years ago, if you wanted
to build a system for reading restaurant reviews,
it would have taken writing
a lot of software code that looks like this.
Pages and pages of software that you
would need machine learning engineers to write.
Specifically the process of building
a restaurant reputation monitoring
review system would have looked like this.
You would use supervised learning.
That's that technology that maps from inputs A to
outputs B and if I were building the system,
I would start by collecting
maybe a few hundred or a few thousand data points.
With examples like this,
I would have a review,
best soup dumplings I've ever eaten.
That sounds delicious.
Label that as a positive review.
The colorful table class made me smile.
That's positive. Or not worth the three month wait,
that would be a negative review.
The process of building the system
would involve first getting label data,
then finding an AI team to help
train an AI model on the data,
to learn how to output positive or
negative depending on different inputs A.
Then finally, you might have to find
a cloud service like AWS or Google Cloud or
Azure to deploy and run the model
so that when you then input
best bubble tea I've ever had,
that would hopefully recognizes
this as having a positive sentiment.
This process would often take months.

In contrast, if you were to use prompt based development,
this is the code you would need
to develop a sentiment classifier.
First, here's how we specify a prompt in code.
My prompt, which I've set equal to two parts of text.
There's the instruction text,
classify the following review as
having either a positive or negative sentiment.
Then here is the review text.
After specifying the prompt in code,
I just need one line of code
to call the large language model to get
a response back and then I'm going
to have it display or print the response.
This is pretty much
all the code it takes to build such a system.
In fact, in the next video,
I'll share of you an optional exercise
where you can try out this code yourself.
Whereas, with the traditional approach to
building a sentiment classifier,
using supervised learning,
the timeline for the project might have
been a month to get, say,
1,000 labeled examples with
1,000 reviews and the positive negative labels.
After collecting the data,
it might have taken a team, say,
three months to train the AI model
on data and then another three months
to deploy it and make sure it's
running well and rugged and robust.
I don't know if this seems like a long time to you,
but for many really
good machine learning teams I've worked with,
the 6-12 month timeline was pretty realistic for what it
took to build and deploy a valuable AI model.
This worked, and this
was very valuable for a lot of applications,
but this just took a long time.
In contrast for prompt base AI,
this is what it feels like.
You can specify a prompt in minutes or maybe hours,
and then deploy the model in hours or maybe days.
There are now many applications that had
previously taken me and very good machine learning teams,
maybe 6-12 months to build that today I think there are
millions of people around the world that can now
build in maybe days or a week.
This is fantastic, because

this lowering of the barrier to entry to building
such applications is leading to
a flourishing of a lot more AI applications.
With one important caveat,
which is that, as we discussed last week,
generative AI tends to work much better
for unstructured data like texts,
and images and audio.
But with that admittedly important caveat,
the number of applications built on top of generative AI
is just letting the community
do much more than ever before.
In the next optional video,
I'd like to invite you to try out some codes with
me for reading
restaurant reviews and classifying sentiment.
It's fine if you've never seen or
written a line of code before in your life.
But I'm hoping to convey to you how little code
is needed to do this now and let you try it out yourself.
I hope you take a look though also feel free
to skip it if you wish and after that,
we'll come back and talk about what building
a generative AI software project feels like
when we talk about the life cycle
of a generative AI project.

I'd like to share with you what the process of building

a generative AI software application

feels like. Let's take a look.

Here's what the lifecycle of

a generative AI project to

build a software application feels like.

We would start off by scoping a project to

decide what do we want this software to do.

For example, say you decide you want to

build a restaurant reputation monitoring system.

The next step would be to actually try to implement it.

And given the ease of

building AI applications using generative AI,

which you may have seen in

the optional video that came before this one,

very often you build a prototype quite quickly and

then plan to over time improve this software prototype.

For some applications I've worked on,
we would build the initial prototype in
one or two days and that initial prototype,
frankly, isn't that good initially.
But building it quickly lets us then take it into
internal evaluation where we
might have our own internal team,
different restaurant reviews and test the system to
see how often it is giving a correct response.
Sometimes the internal evaluation will turn out
some examples where it doesn't give the right result.
In this case with,
"My pasta was cold," it
outputs this as a positive sentiment
and sometimes cold pasta is delicious
but this sounds like a negative sentiment to me.
Based on problems that we discovered internally,
we'll then go back to continue to improve the system.
As you saw last week,
writing prompts is a highly iterative process
where you have to try something,
see if it works, and then improve it.
And building a generative AI software application
also tends to be a very iterative process.
After a sufficient internal evaluation to give
you confidence that the systems working well enough,
then we would deploy it out in
the world and continue to monitor its performance.
And it would not surprise me if you deploy something and
initially external users also
generates input that causes
the system to make some mistakes.
For example, maybe a user writes,
"My miso ramen tasted like
tonkotsu ramen." Is this good or bad?

Well, if you're not familiar
with ramen or Japanese cuisine,
you may not know is this a good thing or a bad thing.
And if your system rates this as a positive sentiment,
but it turns out that if
you're ordering miso ramen on the menu,
you probably don't want it to taste like tonkotsu ramen,
which tastes more like pork based soup broth.
When you find incorrect responses
like this out in the world,
you might decide to go back to internal evaluation.
For example, to systematically
understand if your system is,
say, underperforming on certain types of cuisine.
Or you might decide to go back to take
these learnings to improve
the prompt or improve the system further,
assuming you decide that
these types of errors are unacceptable.
It turns out that building
generative AI software is a highly empirical,
and by that I mean a highly experimental process.
Meaning that we're repeatedly try
something and then find and fix mistakes.
We've already seen how prompting
itself is a highly empirical process,
where you would have an idea, try the prompt,
see the element response,
then maybe update your idea and the prompt and go again.
But other than updating the prompts,
there are other tools that we'll talk about this week for
improving the performance of your generative AI system.
One tool that we talk about later this week is
RAG or retrieval augmented generation that
gives the large language model

access external data sources.
We'll also talk later this week about a technique called
fine tuning that allows you to adapt
a large language model to your task.
And then finally, pretraining models,
which refers to training
a large language model from
scratch. Don't worry about it.
If you don't know what any of these terms mean,
we'll go through each of them in depth later this week.
But they're all key techniques that,
in addition to prompting,
gives you different ways to improve the performance
of your generative AI system's performance.
Just to walk through a second example
of the life cycle of your generative AI project,
let's look at what's building
a system to take food orders might look like.
Say you decide to scope
a food order customer service chatbot to take orders.
What you would do, is start by building the system
and quickly throw together a chatbot to take food orders.
Then because we don't know
how well this is doing internally,
you might let your internal team try it
out and place different orders and see how well it
does and sometimes they'll generate
good responses like do you have
pickles on the cheeseburger
and they'll ask if you want some.
And sometimes it will give an unexpected poor response,
such as if you do have mushrooms on
your burgers but for some reason the chatbot says,
I'm sorry, we don't have mushrooms.
Similar to what we saw for

the restaurant reputation monitoring system,
it will be by discovering
mistakes like these that helps you to
improve the system and after you're sufficiently
confident that this is safe to deploy externally,
you can then deploy it and let
customers place real orders and monitor
the large language models responses
to make sure that if it still
says anything it isn't quite supposed
to that you can continue to improve its performance.
Having built a number of generative AI projects,
I've often been surprised and delighted by
the strange and wonderful things
that the users will try to do with your system.
For example, if a user asks,
how many calories are there in your burger?
Initially, the system may not know.
But if you discover this, you can then update
the system using perhaps a technique called RAG I
mentioned just now and it will go
into depth later this week to
allow your software application to give a correct answer.
So that's what building
a generative AI software application feels like.
And if you work at a company with
a few or a lot software developers,
and if you ever come up with a cool idea for
a generative AI application
that your company could build,
this hopefully gives you a sense of what
that process of getting it built might be like.
Now, one of the worries I sometimes hear about is,
is it really expensive to use
these large language models

hosted by companies on the Internet?
It turns out that the use of
these large language models is
probably cheaper than many people think.
In the next video, I'd like to share with
you some intuitions about how
expensive it is or isn't to
actually use these large language models.
Let's go on to the next video.
Required
English

Help Us Translate

In this video, I'd like to walk through
with you some quick examples to build
intuition about how much using
large language models in
the software application actually cost.
Let's take a look. These are
some example prices for prompting and getting
responses from different large language models
that are available to developers.
That is, if you call
these large language models in your code.
OpenAI/GPT3.5 charges 0.2 cents per 1,000 tokens.
GPT4 costs quite a bit more,
six cents per 1,000 tokens and Google's PaLM 2
and Amazon's Titan Lite are also pretty inexpensive.
What I'm showing here are the cost of
generating different numbers of tokens.
Technically, these large language models
charge for the length of the prompt as well,
but the length of the prompt,
sometimes called the inputs is
almost always cheaper than the cost of the outputs so
let's just focus on the cost

of the output tokens for now.
You may be wondering, what is a token?
It turns out that a token is
loosely either a word or a subpart of a word.
Because that's how large language models process text.
Common words like v or example
would be counted as a single token
when a large language model processes it.
Or my name, Andrew,
is a relatively common name,
and so that's also a single token.
But a less common word like translate might
be split by a large language model
into two sub-parts of words,
tran and slate,
and so having it generate
translate will cost you two output tokens.
Unlike the more common words,
which will cost you only one token.
Or programming, it turns out,
might be split by LLM into program and ming,
and also costs two tokens.
A less frequent word like tonkotsu might
be split into four tokens with ton and k,
and ots and u.
But average, over large collections of text documents,
roughly each token is about 3/4 of a word.
If you were to generate 300 words,
that would cost you about 400 tokens.
Don't worry about it if the math
doesn't totally make sense.
But the intuition I hope you take
away from this is the number
of tokens is loosely equal to the number of words,
but a little bit bigger.

It turns out to be roughly 33%
more than the number of words.
On the next slide, we'll do this calculation assuming
a cost of 0.2 cents 1,000 tokens.
But of course, if you were to use different LLM options,
the cost may be higher or lower.
Imagine that you're building
an LLM application for your own team,
maybe to generate text as useful for them to read.
Let's estimate how much it would cost to generate
enough texts to keep someone on
your team occupied for an hour.
Typical adult reading speed might be
about 250 words per minute.
To keep someone occupied for an hour,
you need to generate 60*250 words,
which is 15,000 words that the LLM has output.
But we need to prompt the LLM as
well to generate this output.
If we assume that the length of the prompt
is comparable to the length of the output,
that might add another 15,000 words.
That is, if we need to prompt
it in total for 15,000 words worth of input,
and then also generate 15,000
words of output to keep someone occupied for an hour.
Of course, this is a very crude assumption,
but perfectly good enough for
the purposes of building intuition.
In total, we need to pay for 30,000 words.
Ans as we saw on the previous slide,
because each token corresponds to roughly 3/4 of a word,
30,000 words corresponds to about 40,000 tokens.
If the cost is 0.2 cents per 1,000 or 1k tokens
then generating 40,000 tokens costs 40 times that,

0.002*40, which is equal to eight cents.
If your software application uses
a Cloud-hosted LLM service by OpenAI,
Azure, or Google or AWS or others,
that's maybe eight cents to
keep someone busy for an hour.
I have not made a lot of assumptions in this calculation,
but this seems decently inexpensive.
In the United States,
minimum wage for many places is
maybe around 10-15 dollars an hour,
so paying an additional eight cents per hour of someone
reading intensely seems like a small incremental cost,
especially if it helps them be more productive.
Of course, if you have a free product
that a million users are using,
then eight cents times a million with
no associated revenue can get expensive.
But I find that for many applications,
using an LLM turns out to
be cheaper than most people think.
I hope this gives
some useful intuition for the cost of LLMs.
Let's go on to the next video.
We'll learn about some more advanced technologies they
can use to make your LLMs even more powerful.
I'll see you in the next video.
Required
English

Help Us Translate
Well, we've already seen that
prompting a large language model can take you quite far.
But there's a technique called
Retrieval Augmented Generation or RAG,
that can significantly expand what you can

get an LLM to do by giving it
additional knowledge beyond what it may have learned
from data on the Internet or other open sources.
Let's take a look. If you were to
ask a general purpose chat system,
such as one of the ones on the internet,
a question like, is there parking for employees?
It might answer something like,
I need more specific information
about your workplace because it
doesn't know what is the parking policy for your company.
But RAG or Retrieval Augmented Generation will see,
is a technique that can give the LLM
additional information so that
if you ask it if there's parking,
it can refer to policies specific to your company.
How does it work? RAG has three steps.
Step 1 is given the question,
is there parking for employees?
It'll first look through a collection
of documents that may have the answer.
For example, if your company has different documents on
the benefits offered to employees and
the leave policy and some documents on the facilities,
and some documents on payroll processes,
then the first step in the rag system would be
to have a computer find out which,
if any, of these documents is
most relevant to this question.
Parking seems like a question about the facilities,
about the building that your team
works in and so hopefully
you'll select out the facilities
document as most relevant.
The second step is then to incorporate

the retrieved document of
the retrieved text into an updated prompt.
Let me construct a prompt as follows.
I'm going to say use
the following pieces of
context answer the question at the end,
and then I'm going to take the relevant text
from my facilities documentation with
the parking policy that
all employees may park on levels 1
and 2 and so on and put that into my prompt.
This is now pretty long prompt because it tries
to give a lot of context for the LLM.
Now remember last week we had spoken about
limitations to the prompt length
or the input length for a large language model.
That's why in practice,
rather than dumping
an entire very long document into the prompt,
you might pull out just the part of the document that's
most relevant to the question and
put just that into the prompt.
Then finally, we add the original question,
is your parking for employees?
This is called Retrieval Augmented Generation or RAG,
because we're going to generate an answer to this,
but we're going to augment how we generate
text by retrieving the relevant context or
the relevant information and
augmenting the prompt with that additional text.
Having constructed this prompt,
the final step is to then
prompt the LLM with this rich prompt.
Hopefully the LLM will then give us
a thoughtful answer telling us about where we can park.

In some applications using
RAG in the output shown to the user,
we would also add a link to
the original source document
that led to this answer being generated.
In this case, we might link to
that facilities documentation so the user can,
if they wish, go back and read
the original source document and
double check the answer for themselves.
RAG Retrieval Augmented Generation
is an important technique that is enabling
many LLMs to have context or to have information
beyond what it may have learned on the open Internet.
Here are some examples of RAG based applications.
There are many companies today,
they are offering software that let
you chat with a PDF file.
For example, if you're reading a white paper
but you maybe don't have
time to read the entire thing carefully,
but have a question that you want
answered based on that white paper.
There are many applications today like PandaChat,
AskYourPDF, ChatPDF,
and many others that let you
upload your PDF file and then ask questions.
And they'll use RAG to try to generate answers for you.
I find that some of these software packages
work better and some work worse.
So the results you get may vary.
But there's certainly been a lot
of excitement and interest about
building applications to let
you chat with your PDF files.

There are also more and more RAG applications that
will answer questions based on a website articles.
For example, Coursera Coach does multiple things,
but one of the things it does is use RAG to try to
answer questions based on
content on the Coursera site itself.
SnapChat also has a chat bot that uses texts
from Snap to try to answer
different questions you might have about their products.
Hubspot, which is a marketing automation company,
is another example of a company that
has a chat bot that lets you pose questions
and tries to generate answers for you based
on content from the company or from the website itself.
So these types of chats are becoming
an alternative way to let
users get answers to questions that they
may have about your company's offerings.
RAG is also leading to new forms of web search.
Microsoft Bing has a chat capability.
Google has a generative AI feature
as well that can generate text in response to
your queries and startup.com which
was actually started by one of my former PhD students,
Richard Socher is a web search engine that was
built centered on a chat like interface.
RAG is used in many applications today,
and excitingly, it seems to
be transforming even web search.
To wrap up this video,
there's one big idea I'd like to share of you,
which is to think of the LLM,
not as a knowledge store,
but instead as a reasoning engine.
LLMs may have read a lot of texts on the Internet,

and so it's tempting to think of them
as knowing a lot of things and they do,
but they don't know everything.
With the rag approach,
we provide relevant context
in the prompt itself and we ask
the LLM to read that piece of text and
then to process it to get to the answer.
In other words, rather than counting on it to have
memorized enough facts to get us the answer,
we're instead using it as a reasoning engine to
process information and not as a source of information.
Find that this way of thinking about LLM's as
a reasoning engine rather than as
a way to store and retrieve information,
can expand the set of applications that
we might brainstorm and
consider an LLM to be capable of doing.
Admittedly, LLM technology is
early and it doesn't always do that well.
But if an LLM isn't
just a database that stores a lot of information for you,
but it can process and reason through information.
I think that is an exciting direction
to think about where LLMs might go from here.
Even though I've talked mostly about RAG in
the context of building software applications,
this idea can also be useful
if you're using a web user interface.
Sometimes I would take a piece of text and just
copy it into the prompt of an online web UI of
an LLM and then tell it to use that context to generate
an answer for me and that
too can be an application of RAG.
I've found that RAG is

useful for many different applications,
and I hope that you will too.
In the next video, we'll talk about
another technique called fine tuning,
which is another way to expand what an LLM can do.
But before I wrap up, let me just say,
I hope you enjoyed this video on RAG and that you
can really clean up with this RAG stuff.
I'll see you in the next video.
Required
English

Help Us Translate

Whereas RAG gives you one way to
give additional information to a large language model,
there's another technique called fine-tuning,
which is another way to give it more information.
In particular, if you have context that is bigger,
that can fit into the input length or
the context window length of the LLM,
then fine-tuning gives you another way
to get LLM to absorb this information.
Fine-tuning also turns out to be useful for
getting the LLM output text in a certain given style.
But this actual implementation is a bit harder than RAG.
Let's take a look. Let's say you have
an LLM trained the way that we had
described previously with sentences found on
the Internet like my favorite food
is a bagel with cream cheese.
Then it may have learned from
hundreds of billions of words,
or maybe more than a trillion words,
to predict the next word like this.
An LLM like this will have learned to generate
text that sounds like what's on the internet.

This process of training a large language model on
a lot of data is often called pre-training.
Now, let's say I want to modify the LLM to have
a relentlessly positive and
optimistic attitude about everything.
There's a technique called
fine-tuning that we can use to cause
the LLM to do a little bit more
learning to change its outputs to be,
in this example, much more positive and optimistic.
To fine-tune the LLM,
we would come up with a set of
sentences or a set of texts that takes on a positive,
optimistic attitude, such as
what a wonderful chocolate cake
or the novel was thrilling.
Given texts like this,
you can then create
an additional dataset using
what a wonderful chocolate cake
you would have given what.
Next word, it will try to predict a,
what a next word is, wonderful,
what a wonderful chocolate, and so on.
It turns out that if you take an LLM that has
been pre-trained on hundreds of billions of words,
and fine-tune it on just an additional, say,
10,000 words or more,
could be 100,000 words if you have
more data or even 1 million words
if you have even more data.
Fine-tuning to
this relatively modest-sized dataset can shift
the output of your LLM to
take on this positive, optimistic attitude.

Now, maybe shifting an LLM to have
a relentlessly positive attitude
isn't that helpful an application,
but fine-tuning is used in many real applications.
One class of applications that fine-tuning is useful
is when the task isn't easy to define in a prompt.
For example, if you want to use
an LLM to summarize customer service calls,
a generic LLM may look at a call like this and summarize
it to say the customer tells the agent
about a problem with a monitor.
But if you run a customer call center,
you might want it to generate
specifics about what the conversation was about.
It was about the MK401-27KX
reported broken by customer 5402 and so on.
If you create a dataset with
maybe just hundreds of examples of human expert
written summaries and have
a large language model that's learned from
hundreds of billions of words on the internet,
so it's learned a lot of
general knowledge on the Internet.
But if you additionally fine-tune it
on maybe just hundreds
of carefully handwritten summaries of the specific style,
then that would shift the LLM's ability
to write summaries in the style that you want.
The specific style summary is
actually not that easy to define in a text prompt.
Maybe you could do it,
but fine-tuning would just be a very precise way
to tell the LLM what summaries you want.
Another example of when a task
isn't easy to define in the prompt is

if you want to mimic
a specific writing or speaking style.
So Tommy Nelson, who's
been working with me on this course,
actually tried just for fun,
to get an LLM to sound like me,
but it turns out that the way most individuals
sound is not that easy to describe in a prompt.
How would you give
someone clear instructions to sound like me?
So if you were to prompt
a general-purpose LLM and ask it to sound like me,
you'd get texts like this,
which I don't think it sounds that much like me.
But if you were to take a lot of
transcripts of the way I actually
talk and have an LLM be
fine-tuned to train it to really
sound exactly like me by learning on my actual words,
then asking it to write something
that sounds like me results in texts like this,
which I don't know, that
sounds more like how I would talk.
But because mimicking a
specific writing or speaking style
is very difficult to do via prompting
because it's just difficult to describe
a specific person's style by writing text instructions,
fine-tuning turns out to be
a more effective way to get
an LLM to speak in a certain style.
If you're building an artificial character,
maybe a cartoon character,
fine-tuning could also be a way to get
an LLM to speak in a certain style.

Other than ties that easy to defining the prompt,
a second broad class of applications of
fine-tuning is to help the LLM gain a domain of knowledge.
For example, if you want an LLM to
be able to read and process medical notes,
this is what a medical note
written about a patient by a doctor might look like.
This is really not normal English.
Pt is patient, c/o,
complaining of, SOB,
shortness of breath, DOE,
dyspnea on exertion, PE,
this is the results of the physical
examination and so on.
Treatment is the follow up
with the primary care physician,
STAT chest x-ray,
containing treatment as needed on oxygen.
But this is really not normal English and if you
were to take an LLM trained on normal English,
it wouldn't be very good at processing text like this.
If you were to fine-tune an LLM
on a collection of medical records,
then the LLM could get much better at absorbing
this body of knowledge about
what medical notes sound like.
You could then use that to build
other applications on top of it to better
understand medical records or legal documents.
Here's a piece of
legalese written by lawyers for lawyers,
that's really difficult for non lawyers to read.
Licensor grants to licensee per Section 2(a)(iii),
a non-exclusive right and so on and so on within
15 days hereof. I don't know about you.

I did not use the word hereof in
my ordinary day to day speech.
But this is what legal documents sound like.
If you want your LLM to gain a body of knowledge
about how to read and understand legal documents,
then taking LLM and fine-tuning it to
legal documents would help it
to gain that body of knowledge.
Similarly, financial documents too.
fine-tuning an LLM on a large set of
financial documents would help it to better gain
that body of knowledge about finance and make it
better at applications involving
processing documents that look like this.
Finally, another reason to fine tune an LLM is to get
a smaller model to perform
a task that may previously have required a larger model.
We'll discuss later this week
some of the pros and cons of
choosing a larger versus a smaller model.
But for some LLM applications that need
a lot of knowledge or need complex reasoning,
you might use a relatively large model,
say with over a 100 billion parameters.
But if we were to use a model like that,
such a model may have relatively high latency.
Meaning, after you're prompted,
you might need to wait a while to get back a response.
If you were deploying this on your own computers,
it could be quite costly.
Even though we said in the earlier video
that these models aren't that expensive,
maybe you want it to be even cheaper.
That's because a 100 billion parameter model may take
specialized computers such as

a GPU server or other really fast computers to run.
You probably have a hard time running
such a large model on a normal laptop or PC,
and certainly, not on a smartphone today.
But if you can get
your application to work on a much smaller model,
say one billion parameters,
then that's the range of model size that they would run
much more easily on
a laptop or a PC or on a mobile phone.
For example, if what you want is to
classify restaurant reviews as
positive or negative sentiment,
this is a simple enough task that you probably don't
need a 100 or 200 billion parameter model to run.
But maybe a one billion parameter
model would be just fine,
maybe even smaller, frankly.
But these smaller models aren't as smart
or they aren't as good as a really large models.
Which is why if you were to take
a small model and then fine-tune
it on the dataset like the one shown here,
not just three examples,
but maybe a few hundred or
maybe a 1,000 examples if you have that much data,
then you can get a small model,
say one billion parameters to do
really well on a task like this.
To summarize, fine-tuning gives you
another technique in addition to
RAG to help improve the capabilities of an LLM.
You might use it for tasks
that are hard to specify in a prompt.
Such as if you wanted to output texts in

a certain style or if you want the LLM to
gain a body of knowledge such as about
medical notes or if you want to get a
smaller and cheaper to run LLM to do
a task that might otherwise have required a larger LLM.
It turns out that RAG and fine-tuning are
both relatively cheap to implement.
RAG just is modifications
of your prompt and fine-tuning,
you might be able to get started with tens of dollars or
maybe even hundreds of dollars depending
on how much data you want to fine-tune on.
There's another technique, pre-training
your own model that turns out to be very expensive today,
almost no one other than reasonably large companies,
usually tech companies are attempting this.
But for completeness, let's take a look at
the next video and what pre-training involves.
Required
English

Help Us Translate

Many of the LLMs we've been using have been previously trained, or
we say pretrained by some company, often by a big tech company.
When should you pretrain your own model?
This turns out to be so expensive that when in doubt,
I would say probably don't do it.
But let's take a deeper look.
Many teams have been pretraining general-purpose LLMs by learning from
text on the internet.
These efforts to train very large language models may cost tens of millions of
dollars, need a large, dedicated engineering team, take many months, and
a huge amount of data.
Many teams have been open sourcing such models, and
that's been a fantastic contribution to the AI community.
If you have the resources to pretrain models and maybe even open source them,
please, by all means make that contribution to AI.
I think that could be fantastic.
But for building a specific application, given the time and expense of pretraining
a model from scratch, I think of this as often an option of last resort.
It could help if you have a highly specialized domain and a lot of data.
For example, Bloomberg is a company that offers software as well as

media articles centered around financial services.
Because of its access to a huge amount of text on finance,
it trained BloombergGPT, which is Bloomberg's custom built large
language model purpose-built for financial applications.
And Bloomberg reported that compared to general purpose
LLMs that had learned mainly from internet data,
this model does quite a bit better on processing financial text.
For many practical applications, unless you have a huge amount of resources and a huge amount of data,
it may be more practical to start with an LLM that someone else had pretrained.
Say, a general purpose LLM that's learned from a lot of internet data and
that someone has open source, and then to fine tune that to your own data.
And that will often give pretty decent performance, but
in a much more economic way.
Now, I am sincerely very grateful to the teams that have been putting a lot
of resources into pretraining LLMs on a lot of text data on the internet and
then open sourcing them.
And in fact, this gives us many different LLMs that we could choose from to use.
In the next video,
we'll actually take a look at the issue of what size LLM do you want to use?
And of all the different LLMs out there,
how do you think about choosing among different ones?
Let's go take a look at that in the next video.
When using an LLM to build software applications,
you find that there are a lot of different LLMs out there.
Some big ones, some small ones,
some open source, some closed source.
How do you choose from all of these different options?
In this video, let's take a look at some guidelines.
One way to estimate how capable
an LLM is is to look at the model size.
Loosely, if we look at models that are, say,
in the one billion parameter range,
we'll find that they're often good at
pattern matching and will have
some basic knowledge of the world.
If what you want is to
classify restaurant reviews for sentiment,
I think a one billion parameter model would probably
be able to do just fine in
terms of that type of pattern matching,
with basic knowledge about food types of words.
As you go to a 10 billion parameter model,
you find that the models have greater world knowledge.
They just know more esoteric facts about the world and
the models also get better
at following basic instructions.
So if you want to build a food order chatbot,
a 10 billion parameter model
might be okay especially if you were to fine-tune

it to become better at the types of
specific instructions you want it to follow.
Then the very large models,
say 100 billion-plus parameters,
will tend to have very rich world knowledge.
They'll know a lot of things about physics and
philosophy and history and science and so on,
and they'll be better as well at complex reasoning.
This is why if you're building a food order chatbot,
maybe you don't need the chatbot to know so much
about history and philosophy
and all of these other things under the sun.
Some of these models might be cheap
enough to deploy that it might be
okay to use a huge model even for a food order chatbot.
But where I would definitely tend to use
these larger models would be tasks
that involve deep knowledge or complex reasoning.
For example, if I'm looking for
a brainstorming partner to help me think through ideas,
I'll often use one of the larger models.
One of the things you've heard me say
earlier though is that development using
LLMs is often a highly
empirical, meaning experimental process.
So it's hard to know in advance exactly what
the performance of a given LLM will be.
While I'm sharing some general guidelines here,
in practice it might be worth just
trying a few different models and testing them.
Based on the results you see from testing a few options,
then pick what actually
seems to work best for your application.
Another decision you might have to make is whether to
use a closed-source or an open-source model.
Closed-source models are usually accessible
via Cloud programming interface and
I find that many of them are pretty
easy to build into applications.
You just have to write a few lines of code like we saw
earlier this week to
incorporate them into software applications.
Many of the largest and most powerful models today are
also available only via Cloud programming interfaces and
are closed-source models and they're
also relatively inexpensive to run because
the large companies hosting
these models will often have put
a lot of work into serving up
these API calls inexpensively.

A downside is that if you
develop using these closed-source models,
there is some risk of vendor lock-in.
Today, the switching costs from
one LLM to a different one is not very high.
But there is some cost to retesting
all your problems to see if they work on
a different LLM if you do switch vendors.
In comparison, there are also
many open-source models that are available now.
One advantage of using
an open-source model is you
have full control over the model.
You know you always have access to
that model and don't have to worry about one of
the company providing it were
to tire or deprecate
the model that you had built on top of.
You can also often run these models on your own device.
So if you want to run it on-premises or on-Prem,
that is, on your own service,
or on a PC or a laptop or a mobile device,
then open-source models may give
you a good starting point to do that.
Using an open-source model
might also let you build an application in
a way that retains
full control over data privacy and data access.
For example, I was recently working on an application
using electronic health records
and because of patient privacy,
we just could not upload
the patient records to a Cloud provider.
As for that project,
my team used an open-source model that we ran on
our own computers because we had to do that
to guarantee privacy of the patient data.
To summarize, this week we talked
about software applications built using LLMs.
We saw the life cycle of generative AI project,
as well as techniques like RAG and
fine-tuning that can make your LLM more capable.
Lastly, in this video we talked about
how to choose an appropriate model to build on.
There are also a couple of optional
videos after this one,
one that goes a bit deeper into the technology that
enables LLMs to not just
predict the next word found on the Internet,
but actually follow your instructions

and do so in a safe way.
The other optional video talks about some frontier,
cutting-edge technology that can use LLMs to
automatically decide what to do
and also use tools along the way.
Please feel free to check out those videos if you wish.
Then in the next and final week of this course we'll take
a look at how LLM technology
is affecting businesses and society.
For example, how can you identify LLM use
cases that could be useful for your company?
We'll take a look next week as well at
a systematic way to understand why jobs are more or
less affected by generative AI and
how both the individuals doing the jobs as well as
businesses employing people doing those jobs might
navigate the changes that
generative AI is bringing to work.
I look forward to seeing you next week.

We've been thinking of LLMs as having learned from

a lot of texts on the Internet to predict the next word.

But when you prompt an LLM,

it doesn't just predict the next word on the Internet,

it actually follows your instructions.

How does it do that?

In this optional video,

we'll talk about the technique called

instruction tuning that enables LLMs to do that.

Then also a technique called RLHF,

reinforcement learning from human feedback,

that has been instrumental to making

LLMs outputs more safe.

Let's take a look at what these techniques do.

We've discussed LLMs as having been

pre-trained on a lot of texts like this,

my favorite food is bagel with cream cheese.

An LLM trained on data like

this would be good at repeatedly

predicting the next word

based on what text on the Internet sounds like.

If you were to prompt an LLM with a question like,

what is the capital of France?
It is quite possible that it will reply,
what is the capital of Germany? Where is Mumbai?
Is Mount Fuji or Mount Kilimanjaro taller?
Because you do see lists of questions on
the Internet about say, geography.
If you see a web page
that says what is the capital of France,
it's actually quite plausible that what comes
after it is what is the capital of Germany?
But this isn't the answer you want.
You wanted to say that the capital of France is Paris.
In order to get an LLM to
follow instructions and not just predict the next word,
there's a technique called instruction tuning,
which is basically to take
a pre-trained LLM and to fine tune it on examples of
good answers to questions or
good examples of the LLM following your instructions.
We may give it a question response pair like this,
what is the capital of South Korea?
And fine tune it given
this input prompt to
output the capital of South Korea is Seoul.
Or help me brainstorm some fun museums to visit in
Bogota and fine tune it to an answer like this.
Or an instruction like,
write a Haiku poem about Japan's cherry blossoms,
and fine tune to generate that.
To try to make this safer,
we can also include some examples like,
tell me how to break into Fort Knox.
Fort Knox is a very secure facility in
the United States that stores
a massive amount of the US Treasury's gold.

Trying to break into Fort Knox would be a terrible idea.
Please, don't anyone try to do that.
But I think a good answer for
the LLMs to output will be something like,
I can't assist with that or please don't break the law.
Given a dataset like this,
you can then fine tune
a pre-trained LLM on
a set of good answers to different prompts.
Specifically, given an example about
brainstorming museums in Bogota,
we would turn that into a set of inputs A and output B,
where first the input A will be that prompt,
and the first word it should learn to predict
here is sure and the second word is sure,
here are some suggestions, and so on.
When you fine tune an LLM on a dataset of prompts and
good responses that LLM will
learn to not just predict the next word on the Internet,
but to answer your questions
and to follow your instructions.
This will do okay.
But it turns out that there's a technique called
reinforcement learning from human feedback or RLHF,
that can improve the quality of answers further.
Many companies training LLMs want
the LLM to give results
that are helpful, honest, and harmless.
Sometimes we call this the triple H,
and the technique RLHF
is a way to try to accomplish that.
The first step of RLHF is to
train an answer quality model.
In other words, will you
supervised learning to learn to rate the answers of LLM.

For example, given a prompt
like advise me on how to apply for a job,
we might have an LLM generate multiple responses such as,
I'm happy to help, here are some steps to follow,
and then have a bunch of useful steps after that.
Or it might say, just try your best,
which is not that hopeful but not that terrible.
Or it might say, it's hopeless, why bother?
That's clearly not a great response.
We would then get humans to help
rate these responses according to how helpful,
honest, and harmless the LLM's output
is so that better answers are given higher scores.
Where the first really helpful answer
might get a score of five,
the second step's answer might get an intermediate score,
and the final answer,
which is terrible, would get a very low score.
If we treat the responses and scores as
the input A and the output B
for a supervised learning algorithm,
then we can train an AI model
using supervised learning to take as
input response from an LLM and
score it according to how good the response is.
The second step of this RLHF process is to
then have the LLM continue to generate
a lot of answers to a lot of different prompts.
We now have this AI model to
automatically score every single response
that the LLM has generated,
and this can be used to tune
the LLM to generate more responses,
they get higher scores.
The reason this technique is called

reinforcement learning from human feedback is because
the scores correspond to the reinforcement or
the reward that we're giving
the LLM for generating different answers.
By having the LLM learn to generate answers that
merit higher scores or
higher rewards or higher reinforcements,
the LLM automatically learns to generate
responses that are more helpful, honest, and harmless.
So that's how an LLM learns to follow instructions.
The first step is basically fine tuning,
where you fine tune it to follow
instructions and to answer questions,
and then second is RLHF,
reinforcement learning from human feedback to
further train it to generate better answers.
In the last final optional video,
we'll also take a look at
some cutting edge ideas in
the technology development of LLMs.
Thanks for sticking with me in this video,
and I hope to see you also in the next optional video.
Required
English

Help Us Translate
Speaker 2: Welcome to the final video of this week.
I'd like to share with you in this video how LLMs are starting to be able to use
tools and then also discuss a cutting edge topic of agents, which is where we
let LLMs try to decide for themselves what action they want to take next.
Let's take a look.
In the early example of a food order taking chatbot, we saw that if you
were to say send me a burger, the bot may reply okay is on the way.
In order for a chatbot to enter the order and send it to you.
This is what actually is happening behind the scenes.
The LLM can't just say OK is on its way because it needs to take

some action to actually send the burger to you.
And so an LLM might output this response order burger for
user 9876 to send to this address and
then also say the user message is to say OK, is on its way.
An LLM that's been fine tuned to output text like this will be able
to generate an order which in this case would trigger a software
application that passes the restaurant ordering system
a request to deliver a burger to this user at that address.
And what is shown to the user is not the full LLM output.
The full LLM output is all four lines of text here, but only the final
line OK is on its way is what gets sent to the user as the response.
So this is an example of tool used by an LLM, where detects the LLM
outputs can trigger calling a software system to place a restaurant order.
Now, placing an incorrect order can be a costly mistake.
So perhaps a better user interface would be, before finalizing the order
to pop up a verification dialog to let the user confirm yes or no if you've
got the order right before charging the credit card and sending it to them.
And clearly, given that LLM's outputs are not completely reliable for
any safety critical or mission critical action,
it would be a good idea to let a user confirm the desire action before
letting the LLM trigger some potentially costly mistake by itself.
In addition to tools for taking actions, tools can also be used for reasoning.
For example, if you were to prompt an LLM, how much would I have after eight
years if I deposit $100 into bank account that pays 5% interest?
An LLM might generate an answer like this which sounds plausible, but
the number 147 dollars forts is not actually the right answer.
It turns out LLMs having learned to predict the nick's worth or
maybe even instruction tuned, are not great at precise math.
And just as UI might use a calculator to calculate the right answer
to a problem like this,
we can also give the LLM a calculator tool to help it get the right answer.
So rather than having the LLM output, the answer directly.
If the LLM were to output this after compounding and so
on, you would have calculator 100 times 1.5
that's 5% interest rate compounded to the power of 8.

This can be interpreted as command to call an external calculator program to
explicitly compute the right answer, which turns out to be $147.74.
And plug that back into the text to give the user the correct dollar figure.
So by giving LLMs the ability to call tools in its output,
we can significantly extend the reasoning or the action taking capabilities of LLMs.
Tool used today is an important part of many LLM applications and of course,
designers of these applications should be careful to make sure that tools
aren't triggered in a way that causes harm or causes irreversible damage.
Going beyond tools into a more experimental area,
AI researchers have been examining agents which go beyond triggering a tool to
carry out a single action, but is exploring whether arms can choose and
carry out complex sequences of actions.
There's a lot of excitement and research on agents, but
this is at the cutting edge of AI research.
It is not yet mature enough to count on for most important applications.
But I want to share with you what many in the AI community are excited about.
If you would ask an agent that's built on top of an LLM help me research better
burger's top competitors, then an agent might use an LLM as a reasoning
engine to figure out what are the steps it needs to carry out to do your task
of researching better burger's competitors.
And this reasoning engine LLM might decide it needs to search for
the list of the top competitors, then visit the website of each competitor,
and finally, for each competitor, write a summary based on the homepage content.
And then perhaps by making a sequence of calls to this reasoning engine,
it may figure out that to search the top competitors it has to trigger a tool to
call web search engine on the query BetterBurger's competitors.
And then after that it may visit the websites
of some of the top competitors to download their homepages.
And then additionally call an LLM yet
again to summarize the text that they found on the website.
On the internet there have been some nice demos of agents, but
this technology is not really ready for primetime yet.
But perhaps in the future, as researchers make it better and better,
it'll become more useful.
And I think that would be an exciting future if LLMs as a reasoning

engine can help decide what's the sequence of steps to take safely and responsibly, of course, to help a user carry out a task.

Thank you and congrats on making it to the very end of week two.

We have just one more week to go in this course.

Next week, we'll look at how generative AI is affecting companies, including how you might be able to come up with generative AI use cases for your business, as well as look at how generative AI is affecting society and its impact on jobs.

I look forward to seeing you next week.

Required

English

Help Us Translate

Welcome back. This week we'll
start by taking a look at the role of
generative AI in business and then
after that and its impact on society,
for example, its impact on jobs.
We'll start by looking at how people
in many different job roles can already
use the web user interfaces to
access generative AI in the day to day work.
After that, we'll take a look at
a systematic framework for
analyzing a business to identify
opportunities to use generative AI to
augment or to automate different tasks in the business,
maybe in your business to identify where building or
buying an LLM based software application might add value.
Let's dive in. As you've seen previously,
an LLM can be a pretty good writing assistant
or copy editor.
If you ask it to rewrite a piece of
text to be suitable for a professional business report,
you'll often do a pretty good job.
I use it pretty often myself for this purpose,
although I will double check
its output before using it in my own writing.
Because generative AI is a general purpose technology,
I'm seeing also that it's used by
many different people in many different job roles.
For example, marketers are
using it to help brainstorm ideas.
If you ask it, help me brainstorm and
email campaign to reactivate lost users,
then it might come up with ideas
for email campaigns like this.

If you want, you can even ask it further details
of what a we miss you
email campaign might look like and so on.
Or if you're a recruiter,
I've seen recruiters use LLMs to summarize reviews.
Here it says summarize the finally review of
a job candidate in 50 or fewer words.
It does usually a pretty decent job.
Although again, I would recommend double checking
a summary before using and fully relying on it.
But LLMs are pretty good at summarizing text.
For programmers or software engineers as well,
LLMs turn out to sometimes be helpful at
writing an initial draft of some types of code.
Write Python code to calculate
something fairly technical.
In this case, the LLM generates a correct piece of code.
But once again, it actually often generates buggy code.
When I'm using this for software developments,
I will often end up having to fix it.
But it's helpful to get a programmer started on a task.
People doing many different types of job roles are
already finding LLMs useful in their day to day work.
I find myself often using an LLM
as a thought partner to help me think things through.
I hope that you too will find
it useful in your day to day work.
When you look across an organization,
what are the most valuable opportunities
for using generative AI technology?
Or for even trying to build or
buy software applications built using LLMs.
In the next video, we'll start to take a look
at the framework of looking at jobs,
and tasks done by an organization to
try to identify such opportunities.
Let's take a look at that in the next video.

Many businesses think of a large or a small company, say,

have many people doing many different tasks.

There's a framework that had originated in economics due to Erik Brynjolfsson,

Tom Mitchell, and Daniel Rock for analyzing the work tasks for

possible automation using AI.

And this framework turns out to be useful not just for

economists to understand the financial or economic impact of AI, but also for

businesses identify specific opportunities to use generative AI.

Let's take a look at how to do this.

Whereas there's been a lot of discussion, for example, in the media about will
AI automate jobs, it turns out that from a technical and business perspective,
it's more useful to think of AI not as automating jobs, but as automating tasks.
And it turns out most jobs involve a collection of many tasks.
Let's look at an example.
A customer service representative will do a number of different tasks,
including maybe answer inbound phone calls from customers.
Answer customer chat queries via a text rather than a voice or a phone interface.
They may check status of customer orders, keep records interactions, and
assess the accuracy of customer complaints.
And if you work in a company with, say, many customer service representatives,
the first step to analyzing the potential for
using genitive AI would be to understand for
your business what are the tasks that the representatives in your company do.
After that, we can then take a look at these different tasks and
try to assess their potential for genitive AI to either hope or
augment, or to automate these tasks.
For example, for genitive AI to pick up the phone and have a long conversation,
that's still pretty difficult.
So we assess that to be a lower potential opportunity.
But answering text chat with customers that might have a higher potential,
maybe checking status of customer orders is medium,
whereas keeping records of customer interactions could be high, and
assessing accuracy of customer complaints may be low.
All of these examples in the rightmost column are hypothetical, and
the actual impact on your business will be different and
will depend on the specifics of your business.
But after an analysis like this and I'll go in a little bit into the specifics of
how to carry out this analysis, you might then decide that answering customer chat
queries and keeping records of customer interactions have the highest potential,
and therefore, focus your efforts on those two tasks.
Now, the opportunity for
generative AI could be either augmentation or automation.
By augmentation, I mean, we can use AI to help a human with a task.
In the customer service representative context, we might have generative AI

recommend a response for a customer service agent to edit or approve, but
not fully automate the sending of a message back to the customer.
So if we're not sure yet if the generative AI would give good answers,
then recommending a response could speed up the people doing the work, but
not fully automated, and this would be an example of augmentation.
And automation would be if we have an AI system fully automatically perform a task.
So if we were to automatically transcribe and summarize records of
customer interactions, that could be an example of automation.
What I see in many applications is that businesses will
sometimes start with augmentation to maybe let a human double check or
finalize the output before it is used.
But then as you gain trust and gain confidence in the output of the generative
AI, then the user interface can be adapted to make the process more and
more efficient for humans and to then gradually shift to what?
Higher and higher degrees of augmentation and perhaps eventually,
to full automation.
Now, given a list of tasks like this,
how do you come up with this column on the right?
How do you evaluate the different tasks for generative AI potential?
The potential for augmenting or automating a task depends mostly on two things,
technical feasibility and business value.
So technical feasibility refers to can AI do it?
And also how costly is it to build an AI system to do it?
And with regard to using an LLM,
I found the framework we discussed last week of asking can a fresh college
graduate following the instructions in the prompt complete the task?
That could give you a first guess, an imperfect,
not necessarily fully accurate guess, but
it gives you a way to think about whether a certain task may be doable or not.
And sometimes, if you're not sure if an LLM can do a certain task, I would
encourage you to try prompting an LLM to see if you can get the LLM to do that
task.
And this would be an experiment that you might be able to do quite quickly, so
long as you're not revealing confidential information.
If you take some prompts for, say, answering customer chat queries and

paste them into a large language model,
you can maybe quickly get a sense of how good Rs responds is.
And this could help you relatively quickly assess technical feasibility of using generative AI for a particular cost.
And an AI engineer can also help you assess if more advanced techniques like Rag retrieve, augmented generation, fine tuning or other techniques can help.
And also give you a sense of perhaps, the complexity and
therefore, the cost of building an AI system to tackle a certain task.
In this course, I'm focusing mainly on technical feasibility using
generative AI technology.
If you or your team is familiar with other AI tools such as supervised learning,
you can also assess the technical feasibility of using other tools
as well for augmenting or automating different tasks.
Other than technical feasibility,
the second criteria I urge you to think through is the business value.
So how valuable is it to use AI to either augment or automate a particular task?
And so the questions I would ask to frame up my thinking on business value would be things like how much time is spent on this task?
So how much time savings can we actually realize?
Second, I'd also ask, does doing this task significantly faster, cheaper, or
more consistently using AI create substantial value?
While it may seem like augmentation and automation helps lead to cost savings,
we'll see later this week as well that when you automate a task,
sometimes the benefits are much greater than just cost savings.
Because it also leads to rethinking the workflow around that task.
But if what I'm saying doesn't make sense yet, don't worry about it.
We'll see some specific examples of this later this week.
Before I wrap up this video, there's one more resource I want to share that may be useful for your analysis of how to break job roles down into tasks.
Which is that there are online occupation databases that you can
look up to see what are the tasks that comprise a certain row.
Here's a screenshot from a website called Onet,
which is a US government funded website.
That for the customer service representative row lists lots of different
tasks, including confirm with customers by telephone or in person,

keep records of customer interactions, and so on.
I found that occupation databases like this tend to be general and
not necessarily specific to your company.
And so I wouldn't recommend just using the results from, say,
this Onet database and assuming it's accurate for your company.
There'll usually be some entries there that you read and feel like, no,
this doesn't seem like it applies to my company.
But I found that this is a useful resource to take a look at just for ideas and
to help make sure that maybe you haven't missed anything when thinking through what
are the tasks done by people in different job roles in your company?
Onet is a little bit US-centric, but has a nice,
easy to use user interface, so I encourage you to play with it.
And there are some other countries as well that have some other country or
region specific databases that you may be able to find online as well.
But I found that for many job roles,
Onet is maybe a reasonable initial starting point.
So that's how you can look at different job roles and
start to break them down into tasks and analyze the individual tasks for
potential for augmentation or automation.
And I hope you play with the Onet website and get a feel for
what different tasks in different job roles look like.
In this video, we went through the customer service representative example.
I'd like to go through with you a few examples of other job roles as well.
So let's go take a look at that in the next video.
Required
English

Help Us Translate
I found that for many job roles, people have a mental picture
of that iconic task that uniquely defines that job role.
For example, computer programmers write code.
Doctors maybe see patients.
Lawyers go to court to argue court cases.
And I think that when people think about AI opportunities, often it's instinctual
to say, can AI do that most iconic role, or that most iconic tasks of that job?

But I found that when we actually systematically analyze the tasks that
a particular job is made up of, the best opportunities may or
may not be that first initial instinct.
Let's take a look at a few examples.
If you look at the tasks that a computer programmer does, they do write code.
And so it's tempting when thinking about AI for computer programming to ask,
can AI help write code?
But it turns out computer programmers do many other things.
They have to write documentation, they sometimes respond to user support
requests, they often review others code, and
they often gather requirements for what a piece of software is intended to do.
And if you were to evaluate the generative AI potential for
this job, you may find that writing code can be hard with AI, but
it's a relatively difficult task, but
maybe writing documentation is actually easier to do with generative AI and so on.
Don't take the generative AI potential column too seriously in these examples,
since these are informal evaluations.
And if we're to do a rigorous evaluation based on technical feasibility and
business value, your specific conclusions may be different.
But I think that it's actually easier to get generative AI to write documentation
for code than to actually write the code itself.
And in many different job roles, the best potential for
AI may not be the most obvious first task you might think of.
Let's look at another example.
Lawyers spend a lot of time drafting and reviewing legal documents.
They often have to answer clients questions on how to interpret laws.
If preparing for a court case, they'll have to review evidence, and
sometimes they're involved in negotiating settlements, and
sometimes to represent clients in court.
And I find that a systematic listing out of these tasks, as well as a systematic
evaluation of their potential, may sometimes lead to interesting conclusions.
So I think that there's a high potential for generative AI to help with drafting
and reviewing legal documents, as well as maybe with interpreting laws.
Whereas I can't see a lawyer sending a robot to court to argue on their behalf,
at least not for some time.

And so if you work with a law firm, an analysis like this might help you
decide where you actually want to use generative AI.
One more example, landscaping.
A landscaper has to maintain and care for plants, purchase and transport plants,
maintain equipment, communicate with clients, maintain a business website,
and so on.
I'm listing, of course, just a subset of the tasks that any of these job roles do.
If you were to do an analysis yourself,
you may end up with anywhere from 5 to 15 to 30 tasks per job role.
And in this case,
I think most of these tasks actually have pretty low generative AI potential.
And so the work of a landscaper may be less impacted in the next few years
by generative AI compared to computer programmers and lawyers.
So that's how you can analyze jobs by breaking it down into tasks.
And I encourage you to think through what are the tasks in your work and
where generative AI may be able to help.
Or for the business you may be involved in, to think about how generative AI
could help many different tasks in that business.
When people think about augmentation or automation,
people's minds often go initially to cost savings,
because if you automate something, seems like you can save money.
But in most ways of technology innovation, going back all the way to, say,
the invention of the steam engine, to electricity, to the computer.
Many companies started off thinking about cost savings but
ended up actually putting even more of their effort into pursuing revenue growth.
And that's because growth has no limit, but you can only save so much money.
And when certain tasks are automated, it turns out sometimes
you can rethink the workflow of how the business creates value.
So, for example, if you could do something 1,000 times cheaper because of
automation,
say, answering queries from customers.
Then rather than just taking the cost savings, you may be able to build a new
type of customer service organization that serves people 1,000 times better.
And this type of thinking can lead to growth opportunities that go well beyond
cost savings.

Let's take a look at some examples in the next video.
Required
English

Help Us Translate

There are many ways that generative AI is leading,
not just to cost savings but to revenue growth.
But there's so many ways for general-purpose technology,
like generative AI to create value,
to talk about all of these ways.
But in this video, I'd like to take
a look at a couple of the emerging,
or the more common paths
I'm seeing toward this type of growth.
Let's consider the hypothetical example of a surgeon
preparing to carry out the surgical operation
and then actually carrying out that operation.
Without generative AI tools,
they may need a long time
to do background research to understand
the medical procedure and how to
carry that out in the context of a specific patient.
Then they will go into
the operating theater to carry out that surgery.
If generative AI,
maybe with custom rag or
retrieval augmented generation tools
can help with the research portion of this work.
Then perhaps the time and effort needed to
deeply research the medical procedure could shrink,
and then they still need to go in
to carry out the surgery after that.
This would be an example of
how generative AI could hypothetically help
a surgeon or some other job role where you
need to do some research

and go and carrying out the task,
with this diagram illustrating how
the workflow or the effort needed might change.
Let's look at the second example.
A lawyer reviewing a complex legal document
without generative AI,
first have to gather information from the client.
Sit down with the client to really
understand what's the purpose of this contract?
Who are the people signing it?
What are the key business terms?
Then they might review the documents and then
finally sit down with the client to give them feedback.
With generative AI,
maybe gathering information takes
just about as much effort as before.
But if generative AI can help read over the document,
perhaps this step can become shorter.
But then maybe lawyers still needs to sit
down with the client to give them detailed feedback.
Again, this is a hypothetical example
based on what I'm seeing complies to,
but the details of your workflow may be different.
In contrast, this is what
changes the workflow might look like for
reviewing simple as opposed to complex legal documents.
For comparatively simple legal documents,
say a typical non-disclosure agreement or NDA,
there may not be much information to gather,
and so the lawyer could just review
it and then give the client feedback.
Now, with generative AI,
maybe the review of the document can become much faster.
Then we use generative AI to
create a document to summarize

the issues that the lawyer can
send to the client and
discuss with them more efficiently.
Then the process of giving client feedback
could also become maybe more efficient.
What I'm seeing is that companies
might explore building a system like this,
but after having built such a system,
might decide to further change the workflow,
where after reviewing a document may decide to add
an extra fast human verification step
in the middle just to double check
that what the generative AI said about
the document is comprehensive and correct.
Before then, having generative AI and the lawyer
together try to go to
that last step of giving a client feedback.
This type of re-engineering of workflow is quite common
once both generative AI or
other AI tools are inserted into a system.
Because when one task is automated or augmented,
it often makes sense to re-think
what are the other tasks that need to be done
together in order to deliver a valuable work product,
such as a surgical operation
or a review of a legal document.
Let's look at a third example where the redesign of
the workflow can become even more sophisticated.
If a marketer has to write copy for a website,
that often is very time
consuming and requires a lot of thought.
Then after that, they might push the
new website copy to the website,
and that's how a marketer might market a new product.
But with generative AI tools,

let's say that the marketer finds a way to significantly
accelerate and make more
efficient the writing of the website copy.
Once writing the website copy is so efficient,
it may be worth also investing in better processes
of software to make pushing
to the website more efficient.
We could end here, and if we do this,
then we just have a marketer that is able
to do their work much more efficiently than before.
But now they're able to write
copy and push the website much more efficiently.
Here's how we could rethink the workflow: which is,
when the marketer wants to market the new product,
they could write some copy, push the website,
but then maybe write a different variation on
the copy and push that second version to the website so
they can start to launch
an A/B test where you can now have
two different versions of
a website copy and
start to measure which one works better.
In fact, rather than just testing two versions,
if we are so efficient at
writing copy and pushing copy of the website,
maybe we can now generate four different versions.
So it's not testing two,
but now four versions.
And after collecting data from
how all four versions of the website copy perform,
we might end up sitting down and
actually spending quite a long time analyzing
the campaign performance to see
which website copy works the best.
Then based on that understanding,

go back to further improve the website copy,
maybe by improving the prompts that
we use to generate that copy.
In this example, even though it
seems like generative AI can be used just for
cost saving to help the marketer be much more
efficient in the process of writing and pushing copy.
We can also take advantage of
this efficiency in a totally different way
and write and push and
test many more versions of the website copy.
This leads to other changes as well to
the workflow of the marketer so that
this way hopefully the marketer can
deliver much more compelling marketing campaigns.
Now in this video,
we've seen a few ways that
generative AI can not just save costs,
but lead to significant growth
through new types of service offerings or products.
In addition to searching for growth opportunities by
looking at tasks done by the employees of a company,
there's one other framework for analysis that I found
useful that frankly isn't done as much today,
but may be worth considering for your work,
which is rather than analyzing employees tasks,
consider also analyzing the tasks of your customers.
For example, let's say you offer a product or
a service that helps your customers to build websites.
Then I would urge you to consider
listing out what are the tasks
that your customers have to do.
Not the tasks your employees have to do,
but the tasks your customers have to do.
In this case, maybe your customers have to

first select a template for
the website to get the initial design.
Then they have to write a title for the website.
Then select images, write a copy for the home page,
and optimize the copy for search engine optimization
to make it easy to find via
search engines and maybe other tools as well.
If you were to analyze
the generative AI potential for your customers tasks,
then you may also identify opportunities,
in this case, writing
the title and SEO search engine optimization
as places where generative AI could help your customers.
I found that this framework for
analysis and brainstorming can sometimes
lead companies to build
very different product or
services to deliver to the customers.
This can be another recipe
for getting lots of happy customers,
and therefore pursuing growth for your business.
There are many ways that generative AI is
creating value and can lead to growth.
In terms of brainstorming frameworks,
I found that looking at how you can
automate the augment tasks by employees or
automate the augment tasks by your customers to be
useful ways to approach
thinking about how to create that value.
But if you have some other idea that
isn't arrived at by some
of these brainstorming frameworks,
that's great too and I hope
you explore building those too.
Now, after coming up with

an idea for how to use generative AI in your business,
some probably could be done via a web user interface,
but some may require building
a custom software application.
In the next video, we'll take
a look at common team structures and
some best practices for building
generative AI software applications.
The good news, which I've
alluded to in the first week as well,
is that it may take less resources than you might think,
because generative AI allows
for very efficient building of
AI applications compared to
some earlier ways of AI technology.
But let's go take a look at that in the next video.
Required
English

Help Us Translate
My teams have worked with or advised many companies,
both large and small,
on building a large variety
of generative AI applications.
I'd like to share with you in this video
some of the best practices I'm seeing as
well as what the typical team to
get started on such a project might look like.
The most common roles for building
generative AI applications would be
a software engineer who would be responsible for
writing the software application
and making sure that it runs robustly.
I've seen that when a software engineer puts in
a bit of effort to learn
at least the basics of

large language models and prompting,
then they can be very effective in
a small team building LLM-based applications.
If you're on a team that
already has some software engineers,
it might be worth encouraging them to consider spending
a little bit of time to learn
at least the basics of LLMs and prompting.
A second role that is quite
common on teams to build applications would be
the machine learning engineer
and machine learning engineers
are typically responsible for implementing the AI system.
Many machine learning engineers have been building
AI systems even before Gen AI took
off and found that a machine learning engineer
that spends a bit of
effort to learn about LLMs and ideally not prompting,
but some of the more advanced techniques
like RAG and fine-tuning,
such a person can be very effective,
playing a role building LLM applications.
Finally, one other role that I see in some teams,
but less common than the software engineer and
machine learning engineer would be
the product manager and they would be
the person with primary responsibility
for identifying and scoping
the project and making sure that whatever
is built is useful for customers.
Lastly, how about the prompt engineer role?
There's been a bit of media hype about this role,
but what I'm seeing is that
very few companies are hiring those as a dedicated role.
What happened was a small number of companies

advertised a small number of job openings for
very well-paid prompt engineers and generated a lot of
media hype that someone
could make a lot of money by prompting.
But if you look at the actual job description
of prompt engineers,
prompt engineering jobs actually
require doing a lot of other tasks than writing
prompts and they actually look more like
machine learning engineers that
have additionally learned to prompt.
Don't buy into hype prompt engineer role.
What actually happens in practice is that
most companies are counting on machine learning engineers
that have also learned LLMs or learned
prompting and it's actually not
that easy to get a job and no other company's hiring
that many people whose only job is to write prompts.
If you're building an LLM-based application,
it's often possible to get
started with a pretty small team.
I definitely see companies start to
experiment with even a one-person team,
such as a software engineer
who has learned some prompting.
Or machine learning engineer
who's learned a bit about prompting OMs.
Or maybe you could start by
yourself by experimenting and prototyping,
using some of the web interfaces to
try to get a sense of what might be feasible.
I do see a lot of two-person teams as well.
If you have two persons in a team,
probably the most common configuration is
a machine learning engineer plus a software engineer.

But I've seen many other configurations also work well,
such as a software engineer
who's learned prompting and a product manager,
or two generally enthusiastic
people that know a bit about how to write
software and they're willing to learn how to use
these tools to build new and exciting applications.
Sometimes for the larger teams,
you also see some other roles like data engineer,
data scientist, project manager,
or machine learning researcher.
Let me quickly talk over these roles as well
in case you see them in a company.
Data engineer is usually
responsible for organizing the data
and ensuring data quality and
often also the security of the data.
A data scientist is usually responsible for analyzing
data to make recommendations to
guide project or business decisions.
Project manager can be responsible for
coordinating project execution and machine
learning researchers are
usually responsible for developing
advanced AI technologies or adapting
advanced AI technologies to
the particulars of your business.
Generative AI has lowered the cost,
lowered barrier to entry
to building AI-based applications.
If your team have an idea,
I'd encourage you to try to find
the resources to prototype and try
something out and see if you can build
something for yourself or for your business.

Before we wrap up the section

on generative AI and businesses,

I'd like to go through an analysis of how

AI is affecting different job roles

as well as different industry sectors.

Let's take a look at that in the next video.
Required
English

Help Us Translate

We've looked at how generative AI may be useful to your work and
also talked about analyzing its impact on a business.
Let's now zoom out and take a look at its impact on job roles across different
companies as well as its impact on different industry sectors.
The results from this video may be less directly actionable for
a particular business, but maybe this will help you to think through and
try to forecast some of the large macro changes that may take place over time.
Let's dive in.
A study by Eloundou and others at OpenAI and the University of Pennsylvania,
examined how much different job occupations are exposed to AI,
augmentation, or automation.
That study resulted in this graph showing that higher wage jobs
tend to be more exposed to AI augmentation or automation than lower wage jobs.
In this graph, which has a slightly funny horizontal axis, because it's on
a log scale, we plot salaries ranging from about 30K up to about 163k like this.
And the vertical axis measures the degree to which these jobs
are exposed to automation.
Earlier waves of automation tended to have lower wage jobs more exposed,
because AI could do more of the routine, repetitive work.
So supervised learning, for example,
tended to automate more of the lower wage jobs.
But large language models and generative AI more broadly,
are exposing in this wave the higher wage occupations to automation.
And we'll say more later this week as well about the impact of generative AI on jobs.
Let's look at a second study due to McKinsey,
which carries out an analysis by functional row.
So this graph plots different functions that tries to estimate how much will
sales be impacted, how much will marketing be impacted,
how much will customer operations, including customer service, be impacted.
The vertical axis here, shows the total impact in terms of billions of dollars.
And so, the points in the upper portion of this graph,
correspond to the functional rows where the total value of the impact will
be large in terms of total number of dollars.
The horizontal axis, measures the impact as a percentage of the functional spend.
So customer operations, according to this study,
will have a very large absolute dollar value impact, maybe around $400 billion.
I would take the exact numbers not too seriously since these are frankly loose

estimates, but the total dollar value seems like it will be large because
generative AI is automating or augmenting a lot of customer service.
Moreover, as a percentage of all the spending on customer operations,
generative AI impact will be pretty large as well,
maybe approaching 40% of the total spend on customer operations.
In contrast, it will also have hundreds of billions of dollars impact on sales.
But as a percentage of the total spend on sales, it is much smaller.
And the McKinsey study also estimates that these yellow dots shown on top together,
might represent 75% of the total annual impact of generative AI,
and it will be a significant impact.
Now, this isn't to say that if you work in some of these other functions you shouldn't pay attention to generative AI.
For example, if you work in the legal function and generative AI will impact 15% to 20% of the functional spend on legal, that's still a significant shift for
the industry, even if the total spend on lawyers,
on legal services is not nearly as big as the total spend on sales, or marketing, or software engineering and so on.
But if the McKinsey study is correct, then these are some of the functional roles across many different companies that will have a huge impact through generative AI.
Lastly, let's take a look at its estimated impact by industry sector.
So McKinsey had carried a study on the potential of AI automation with and without generative AI.
And we re-plotted the McKinsey data to show the impact of generative
AI only on automation, leaving out other forms of AI such as supervised learning.
Some of the sectors impacted include, educator and workforce training,
business and legal professions, STEM professionals and so on.
And one remarkable thing about this data is that there are sectors
that were not highly exposed to automation before generative AI.
But with the rise of generative AI is now seeing a much
greater potential of automation or augmentation.
And so depending on what sector or sectors you either work in or work with,
perhaps this type of analysis may give you a sense of what may happen in industries
that you touch as well.
If you look at the top few lines on this graph, one theme that pervades both this as well as other studies is that, it looks like a lot of the impact of generative AI, will be on knowledge workers.
Meaning, workers who generate value primarily through their knowledge,
including their expertise, their critical thinking, and their interpersonal skills.
This is in contrast to, say, workers that create value mainly from
performing physical tasks rather than knowledge tasks.
This wraps up our section on generative AI and business.
There are lots of opportunities for individuals, businesses, and for society.
The huge impact of generative AI is also raising questions about how generative AI will affect society.
And has also made some people anxious about what the future will be like for them in the world with these amazing AI capabilities.
Let's go on to the next video to examine how AI is impacting society as

well as how we can mitigate risk and how we can build beneficial, responsible AI.
I'll see you at the next video.

In a short time, access to generative AI has spread around the world and

given many people the ability to generate high quality essays, pictures, and audio.

With these amazing capabilities have also come many concerns about AI.

I think even before the rise of generative AI,

we've been living in a time of many anxieties.

Anxieties about the environment, about the legitimacy,

incompetence of authority, about society's ability to treat people fairly,

even about what sort of future awaits us all.

AI as a very powerful technology has inherited a large share of this anxiety.

In this video, let's take a look at some of these anxieties and

concern that relate specifically to AI.

One widely held concern about AI is whether it might

amplify humanity's worst impulses.

LLMs are trained on text from the Internet,

which reflects some of humanity's best qualities, but also some of its worst,

including some of our prejudices, hatreds, and misconceptions.

LLMs learn some of these negative qualities, too.

So will it amplify our worst impulses?

In the first week, we had seen an example of an LLM exhibiting a gender bias

with regard to whether a surgeon or a nurse is more likely to be male or female.

To take another maybe slightly simpler example, if you asked an LLM

after its initial training to fill in the blank and the blank was a CEO,

many models would be prone to choose the word man.

And, of course, this is a social bias that distorts the fact that people of

all genders can successfully lead companies.

Text on the Internet represents our present and our past.

And so perhaps it's no surprise that an LLM learning from this data reflects

some of these biases from our past and our present as well.

But perhaps we want LLMs to represent a hopeful future that is fairer,

less biased, and more just, rather than just data from our past.

Fortunately, LLMs are becoming less biased through fine-tuning,

which we discussed in week two.

As well as more advanced techniques,

such as reinforcement learning from human feedback or RLHF.

In the second week, there was an optional video on RLHF.
Whether or not you watched that,
I'd like to briefly describe how RLHF is helping to make LLMs less biased.
RLHF is a technique that trains an LLM to generate responses that are more aligned with human preferences.
The first step of RLHF is to train an answer quality model called
a reward model that automatically scores answers.
So in this step of RLHF, we would prompt the LLM with many queries like this, the blank was a CEO, and collect different responses from the LLM.
Then we would ask humans to score these answers.
So on a scale of one to five, we give a high score to highly desirable answers like man or woman, and a low score to nonsensical answers like airplane.
And any answer that contains a gender bias or racial bias or
contains a gender or racial slur will receive a very low score.
Using the prompt, the responses, and
the scores assigned by humans as data, we would then use supervised learning to train a reward model that can input a response and score it.
We do this because asking humans to score responses is expensive.
But once a supervised learning algorithm has learned to automatically score responses, we can score a lot of responses automatically and inexpensively.
Finally, now that the LLM has a learned
reward model to score as many responses as it wants,
we can have the LLM generate a lot of responses to many different prompts.
And have it further train itself to generate
more responses that get high scores and that, therefore,
reflect answers that humans perceive as more desirable.
RLHF has been shown to make LLMs much less likely to exhibit bias according to gender, race, religion, and other human characteristics.
It makes LLMs less prone to hand out harmful information,
and also makes it more respectful and hopeful to people.
Already today, the output of LLMs are much safer and
less biased than, say, the average piece of text on the Internet.
But technology like this is continuing to improve,
and so the degree of an LLM amplifying humanity's worst qualities is continuing to decrease as they are becoming better aligned to the future.

I think we all hope LLMs will reflect of a fairer, less biased, and more just world.
A second major concern is who among us will be able to make a living when AI can do our jobs faster and cheaper than any human can?
Will AI put many of us out of a job?
To understand whether this is likely to happen, let's look at radiology.
In 2016, many years ago, Geoff Hinton, who's a pioneer of deep learning and a friend of mine, said that AI was becoming so good at analyzing X-ray images that in five years, it could take radiologists' jobs.
He made this remarkable statement that if you work as a radiologist, you're like a coyote that's already over the edge of the cliff, but hasn't yet looked down.
So it doesn't realize there's no ground underneath them.
People should stop training radiologists now.
It's just completely obvious that within five years,
deep learning is going to do better than radiologists.
But we're now well past five years since this statement,
and AI is far from replacing radiologists.
Not a single one of my radiologist friends has lost their job to AI.
Why is that?
Two reasons.
First, interpreting X-rays turns out to be harder than it looked back then,
though we are making rapid progress.
But second and more important,
it turns out that radiologists do a lot more than just interpret X-ray images.
According to O*NET, radiologists do about 30 different tasks,
one of which is interpreting X-rays and other medical images,
but they do many other tasks.
And it has been difficult so far for AI to do all of these tasks at human level.
To list out some of the other tasks that radiologists do,
in addition to interpreting X-rays, they also operate imaging hardware,
communicate exam results with patients or other stakeholders.
Respond to complications during procedure,
such as if a patient has a panic attack during the imaging procedure.
They document procedures and outcomes, and many other tasks.
And I think that AI does have a high potential of augmenting or
assisting the interpretation of X-rays.

And technically,
this has largely been done with supervised learning rather than generative AI.
But for AI to completely automate all of these tasks is still far away.
So that's why I think that Curtis Langlotz, who is a professor of radiology
at Stanford University and a friend and colleague, says it well.
He said that AI won't replace radiologists, but
radiologists that use AI will replace radiologists that don't.
And I think we will see this effect in many other professions.
Mind you,
I don't mean to minimize the challenge of helping many people adopt AI or
the suffering of the much smaller number of people whose jobs will disappear.
Or our responsibility to make sure people affected have a safety net and
an opportunity to learn new skills.
But every wave of technology, from the steam engine to a chassis to the computer,
has created far more jobs than it destroyed.
As I mentioned earlier this week, in most waves of innovation, businesses wound
up focusing on growth, which has unlimited potential rather than cost savings.
So AI will bring a huge amount of growth and create many,
many new jobs in the process.
And this brings us to what might be the biggest anxiety, will AI kill us all?
We know that AI can run amok.
Self-driving cars have crashed, leading to a tragic loss of life.
In 2010, an automated trading algorithm caused the stock market crash.
And in the justice system, AI has led to unfair sentencing decisions.
So we know that poorly designed software can have a dramatic impact.
But can it lead to the extinction of humanity?
I don't see how.
I know there are different views on this, but recently,
I sought out some people concerned by this question, and
I spoke of some of the smartest people in AI that I know.
Some were concerned about a bad actor using AI to destroy humanity, say,
by creating a bioweapon.
Others were worried about AI inadvertently driving humanity to extinction.
Similar to how humans have driven many other species to extinction
through simple lack of awareness that our actions could lead to that outcome.

I tried to assess how realistic these arguments are, but I found that they were
not concrete and not specific about how AI could lead to human extinction.
Most of the arguments boil down to it could happen.
And some will add that this is a new type of technology, so
things could be different this time.
But that statement is true for
every new type of technology that's been invented by humanity.
And proving that AI couldn't lead to human extinction is akin to proving a negative.
I can't prove that AI superintelligence won't wipe out humanity,
but it's just that nobody seems to know exactly how it could.
But I do know this, humanity has ample experience controlling many things far
more powerful than any single person, such as corporations and nation states.
And also that there are many things we can't
fully control that are nonetheless valuable and safe.
For example, take airplanes, which today, we still can't fully control because
winds and turbulence will buffet airplanes around, or
the pilot flying the plane may make a mistake.
In the early days of aviation, airplanes killed many people.
But we learned from those experiences and built safer airplanes, and
also devised better rules by which to operate them.
And today, many people can step into an airplane without fearing for their lives.
Similarly for AI, we are learning better to control it, and
it is becoming safer every day.
Finally, if we look at the real risks to humanity,
such as climate change leading to massive depopulation of parts of the planet, or
hopefully not the next pandemic.
Or even much lower chance, but another asteroid striking the planet and
wiping us out like the dinosaurs.
I think that AI will be a key part of our response to such challenges.
So I know that there are different views on this right now.
But my view is that if we want humanity to survive and thrive for
the next thousand years, AI increases the odds of us successfully getting there.
Computers are already smarter in some narrow dimensions than any human.
But AI continues to improve so fast that many people find it hard to predict
exactly what it will be like in a few years.

I think the root cause of some of these concerns, including extinction risks,

is that many people are unsure when AI will reach

Artificial General Intelligence, or AGI.

Meaning, AI that could do any intellectual task that human can.

Let's take a deeper look at AGI in the next video.
In a short time, access to generative AI has spread around the world and: Added to Selection. Press [CTRL + S] to save as a note
Required
English

Help Us Translate
AGI, Artificial General Intelligence
is an exciting concept.
I think some of the confusion around it
stems from the use of the word general.
As you know, AI is a general purpose technology,
meaning that it is useful for many different things.
The rise of large language models
has led to single models like
ChatGPT that are useful for
many things and feel like they could be general purpose.
But a general purpose technology is not
the same thing as artificial general intelligence.
Let's take a look at what really is
the technical definition of AGI.
The most widely accepted definition of AGI is AI.
They could do any intellectual task that a human can.
Some definitions actually say
any tasks that a human or mammal can,
but I'll stick with this one for now.
For example, if we have AGI,
AI would be able to learn to drive
a car through about 20 hours of practice,
similar to what a teenager can.
This is an example raised by
Deep Learning Pioneer Yana Khan.
Today, self driving cars still aren't quite there yet.
Certainly not after so low as 20 hours of practice.
If we had AGI, AI would also be able to do
the intellectual task of completing
a PhD thesis level research after five years of work,
or maybe even faster.
Today, AI can help with
some parts of brainstorming and writing
and maybe be a thought partner
for some elements of research.
But we're clearly very far away from this still.
If we had AGI, AI

would also be able to do pretty much all the tasks of
a computer programmer or
really any other knowledge workers
whose contributions to work are
to carry out intellectual tasks.
Clearly, we're still very far away from this.
I know that there are different views about
how long it would take to get to AGI.
I think we're still many decades away,
maybe even longer, but I hope we'll
get there sometime in a lifetimes.
There are some businesses that have made
much more optimistic forecast of when we'll get there.
But I found that most of those businesses have changed
the definition of AGI and set
a much lower bar to get there.
I showed the definition of AGI
used by one of these businesses to an economist,
friend of mine and he equipped, boy,
if that's the definition of AGI,
I think we got there 30 years ago.
It's true that by lowering the bar sufficiently,
we could get there much faster.
But for the most widely accepted definition of AGI,
I think we're still quite a long ways away.
One of the exciting things about
large language models is that
we can use them as reasoning engines,
as I mentioned last week.
Maybe we're now starting to see what a
rough outline of what AGI could be some day.
I don't think there's any fundamental laws of physics
that prevents us from ever creating AGI,
which I think will actually
be very valuable to human society.
But we'll still need some significant
technical breakthroughs to get there.
One of the hard things about getting to AGI is that it
benchmarks artificial intelligence
against human intelligence.
Artificial intelligence and biological intelligence
have progress along two very different paths.
For example, AI learning from
far more texts than any human can read in a lifetime.
AI is already far better than
any human at certain tasks already.
But to ask AI to do everything,
all intellectual tasks that humans can do,
that's still just a very high bar.
But even though we're still away from AGI,

AI is very powerful and
it's important that we use it responsibly.
Let's take a look in the next video at Responsible AI.
Responsible AI refers to
developing and using AI in ways that ethical,
trustworthy, and socially responsible.
Lots of developers, businesses,
and governments care about this and
have been having conversations and also
have been working hard to make sure that
AI is built and used responsibly.
Because of all this attention and
effort on responsible AI,
we've actually made quite a lot of
progress on this in the last few years,
with for example, many governments and
companies publishing frameworks for responsible AI.
But a lot of work still remains.
Let's take a look at what responsible AI means.
While we're still figuring out a lot of
details of how to build responsible AI,
some common themes have emerged.
These are some of, I think
the key dimensions of implementing responsible AI.
First is fairness to ensure that AI
doesn't perpetuate or amplify biases.
Transparency to make sure AI systems
and the decisions are understandable to the stakeholders,
to the people impacted.
Privacy, protecting user data ensuring confidentiality.
Security, safeguarding AI systems
from malicious attacks, and lastly,
ethical use ensuring the AI
is used for beneficial purposes.
One of the challenges of these dimensions,
or these principles is that

the implementation is not always straightforward.
For example, I think
at least a couple thousand years now,
humanity has been debating
what is ethical and what is not ethical.
There is unfortunately no clear mathematical definition
of ethical versus unethical behavior.
Although, of course, there are
many clear cut cases as well.
But that's why for individuals,
organizations, even countries to adopt responsible AI,
there are certain emerging best practices to hope,
have the discussion and debate that will lead
to better and more responsible decisions,
even when sometimes the right thing
to do could be ambiguous.
I want to share a few tips.
First, I think it's important to build a culture
that encourages discussion and debate on ethical issues.
If someone on your team has
a concern about the use of responsible AI,
it'd be great if they have the freedom to raise
that issue to enable
the team to maybe make a better decision.
Second tip is to brainstorm,
either by yourself or with your team,
or with an even broader group of
stakeholders how things could go wrong.
I've found on many projects that this brainstorming can
help identify potential problems
and allow the team to mitigate them in advance.
A checklist for brainstorming could be
the five dimensions I described on the previous slide.
Could the AI system have issues with fairness,
transparency, privacy, security, or ethical use.

For example, on some of the projects I've worked on,
my team has brainstorm in advance
if the AI and we deployed could have
fairness issues such as if it might exhibit
some of the biases that you saw earlier in this course.
Finally, I encourage you to work of a diverse team and
include perspectives from all stakeholders
impacted by the AI system.
For many projects, seeking a diverse set of opinions,
as well as speaking with people that
could be quite different than myself has
allowed my team to understand better
the impact of an AI system
and led us to make better decisions.
For example, building systems in healthcare,
I found that talking to
patients and doctors gave perspectives
different than mine and really changed
the direction we took our projects in,
and working on retail applications,
talking to some of the customers as well as the sellers,
gave my team new ideas
that we wouldn't have had otherwise.
I think this pattern is true for many projects.
If you work in a specific industry,
such as healthcare, or finance, or media or tech,
there may be emerging best practices
for responsible AI specific to
your industry that could be useful to
consult as well as you embark on your project.
I think we all want to use AI to make people better off.
There's been a few times that
I've killed projects that are
assessed to be financially sound on ethical grounds.
As you decide what to work on and what not to work on,

I hope you keep on considering

responsible AI and only work

on projects that you think are

ethical and that make people better off.

Now, we're approaching the end of this course.

Let's go to the next video to

see a summary of what we've covered.
Required
English

Help Us Translate

I'm glad you stuck with me all the way to this video.
We've gone through quite a lot of
material together and I'm
actually sad that the time together
in this course is coming to an end.
Let's take a look at the major topics we've covered.
In the first week, we had talked about
how generative AI works including how you can use it as
a thought partner and we touched
on what it can and cannot do and made
the analogy of it be able to do many of the things
that a college student following
instructions might be able to do.
We also talked about
some common use cases including writing,
reading, and chatting.
In the second week, we went over how to
build generative AI projects including
the life cycle of generative AI project and we also went
through a variety of
technology options including prompting,
retrieval augmented generation, and fine tuning.
Finally, this week we discussed
the implications of generative AI
on business and on society.
We went through a framework for breaking
jobs down into tasks to identify
opportunities for automation or augmentation and also saw
how automation or augmentation could lead to
new workflows that don't just realize cost savings,
but lead to significant brand new value creation.
Finally, we discussed societal concerns
and also talked about responsible AI.
Congratulations on making it
all the way to the end of this video.

I hope you've learned a lot about
generative AI and that you found this course useful.
I hope that many people
will be empowered to use generative AI
so I please share what you've learned of others as well,
and perhaps also recommend
that they take this course too.
Before we wrap up, there's
just one last idea I want to share
of you about building a more intelligent world.
Let's go on to the final video of this course.
Thank you for making it to
this last video and for spending all this time with me.
To conclude this course,
I want to share with you how I hope AI
can help us build a more intelligent world.
Intelligence is the power to apply
knowledge and skills to make good decisions.
We invest years of our lives and
trillions of dollars on education,
all to develop our ability to make better decisions.
It costs a lot to feed,
educate, and train a wise human being.
So human intelligence is expensive.
That's why today only the wealthiest people
can afford to hire large amounts of
intelligence by hiring people
like a specialist doctor to examine,
carefully think about, and advise
you on a medical condition,
or a tutor that takes the time to really understand
your child and gently
coach them where they need the most help.
But unlike human intelligence,
artificial intelligence can be made cheap.
AI has the potential to give
every individual the ability
to hire intelligence at low cost,
so that you no longer need to worry
about that huge bill for visiting
a doctor or getting an education
and you can hire an army of smart,
well informed staff to think things through for you.
It has the potential to give
society more intelligent guidance on how to
approach some of the biggest problems
such as climate change and pandemics.
AI is the new electricity with the potential to
revolutionize all industries and
all corners of human life.

The fear of AI today is similar
to the fear of electricity when it was new.
Back then, people were terrified of
electrocution or of
electricity sparking devastating fires.
Today, electricity still has dangers,
but I don't think any of us would give up light,
heat, and refrigeration for fear of electrocution.
AI today still has flaws and
there are cases where it will cause harm.
But generative AI is an exciting leap
forward in how much intelligence
we can bring to the world.
We're also improving the technology rapidly.
As we continue to improve
the technology and also
continue to build up more use cases,
AI will contribute to longer,
healthier, and more fulfilling lives worldwide.
As the technology improves,
the problems of AI that alarm us today will recede.
Looking beyond AI, the world
has many problems such as climate change,
pandemics, war, and many others,
and many of these problems need urgent solutions.
To solve them, we will need all the intelligence,
including all the artificial intelligence we can master.
To conclude, thank you very much for taking this course.
I hope you find generative AI useful,
that you use it responsibly to
improve your life and the lives of people around you,
and that through your use of generative AI
you keep on contributing to building a better,
more intelligent world for everyone. Thank you.
You remember the famous lines Kindred is taking over from the movie Terminator?
For Neo fighting cyber intelligence which
controls human life in the movie matrix.
With the latest disruption in the tech space with Generative AI that seems to be
able to create quality content with few instructions from the user.
Are we taking a step towards an AI driven human civilization?
[MUSIC]

>> Welcome, ladies and gentlemen.

I am Akbar.

I would take you through an exciting journey into the world of Generative AI.

Prepare to be amazed as we unravel the secrets of this incredible technology

that's revolutionizing our lives.

One of the most popular jokes going around in social media today states that

machines are learning and humans are hooked.
A debatable topic for sure.
But let's learn about these machines of today and tomorrow a bit more.
What Generative AI is all about? GenAI is like a digital wizard by your side, capable of generating new and creative ideas, design, and even humor.
That's Generative AI in a nutshell.
But before that, let's try to define Generative AI or GenAI.
Generative AI can be defined as a specialized branch of Artificial Intelligence that tries to generate human like results when asked to be prompted.
These can range from creating codes or stories, paintings or even music, which were typically products of human intelligence and creativity.
GenAI comes up in various forms, each with its own physical, magical abilities.
First up, we have the Content Crafters, the content creator's best friend.
Whether you're an author, a filmmaker, or a game developer, Content Crafters can consider up captivating storylines, compelling characters, and unexpected plot twists.
You can possibly say goodbye to writer's block.
Some of these GenAI can also help you complete your mundane tasks, such as writing an email with no grammatical errors, correct tone, and use of language.
They recommend the best visual e-card for any occasion for anyone in your contact.
Then we have Art Geniuses, the virtual artists extraordinaire.
With these Art Geniuses, you don't need a paintbrush or even a canvas.
This AI can generate breathtaking artwork, design logos, and even compose musical melodies.
It's like having a creative muse on your fingertips.
Wait, there's more.
Enters Smart helpers, your personal AI assistant.
From organizing your schedule to managing tasks, Smart helpers can streamline your daily life.
It can even assist in research, finding the most relevant information in a blink of an eye.
GenAI can be a true game changer, making our lives easier, more enjoyable, and more productive.
But as we harness its power, we must also remember to use it responsibly and

ethically.
On the other side, AI is also helping scientists, engineers, doctors,
and others decode data and complete tasks efficiently.
Which otherwise would have taken years to complete or
could have errors that would not be acceptable.
Responsible use means ensuring transparency and
accountability in how Generative AI is developed and deployed.
It means respecting privacy and data protection.
And most importantly, it means never using AI to harm, deceive, or
discriminate against others.
So as we continue to witness the wonders of Generative AI, let's embrace its
potential while upholding the values of responsibility and ethics.
Together, we can create a future where GenAI enhances our lives,
bringing forth the world of magic and possibility.
Thank you for joining me on this journey through the realm of Generative AI.
Stay tuned for more insights into the fascinating world of AI.
Required
English

Help Us Translate

Welcome to a discussion on the importance of Generative AI in our rapidly
evolving world.
Today, we'll explore how this remarkable technology is transforming industries and
opening up new possibilities for the future.
GenAI, powered by deep learning algorithms, has the extraordinary ability
to generate new and unique content from images, music to text, and even videos.
It possesses the potential to revisualize the way we create and
interact with digital media.
In the world of art,
Generative AI is providing artists with a fresh palette of possibilities.
It can generate breathtaking visual art, explore unique styles and
perspectives that inspire and captivate audiences.
It allows artists to push boundaries, experiment with new concepts, and
express their creativity in ways never imagined.
But GenAI doesn't stop there.
In industries like fashion and design,

it has the potential to disrupt traditional workflows.
Designers today can now leverage Generative AI to generate new clothing designs, predict trends, and streamline the production process
ultimately saving time and resources while delivering innovative products.
Another significant impact of GenAI can be seen in the field of healthcare.
Researchers can now analyze vast amounts of patient data and
generate personalized treatment.
By utilizing GenAI models, medical professionals can predict risks
of diseases, recommend medications, and improve patient outcomes.
In the world of finance, GenAI is revolutionizing risk assessment and
fraud detection.
It can generate synthetic financial data to train models,
allowing institutions to make informed decisions and protect individual privacy.
By continuously learning from historical data, GenAI can adapt to new patterns and mitigate fraudulent activities and safeguard financial systems.
Now, let's imagine the possibilities as GenAI continues to evolve. It holds tremendous potential to tackle some of the world's most pressing challenges.
Imagine a world where GenAI plays a crucial role in environmental
conservation.
It could generate models that simulate climate change scenarios,
aiding scientists in developing effective mitigation strategies.
By analyzing vast data sets, GenAI could help us find sustainable solutions,
optimize the resource allocation, and pave the way towards a greener future.
Furthermore, as Natural Language Processing advances,
GenAI could revolutionize human computer interaction.
We could witness the development of virtual assistants capable of generating humanlike conversations.
Leading to more personalized and engaging interactions across various industries from computer, service to education.
However, as with any powerful technology,
it is essential to consider the responsible and ethical use of GenAI.
We must ensure transparency, fairness,
and accountability in its development and deployment.
Respecting privacy and data protection must be at the forefront of effort,
ensuring the benefits of GenAI are enjoyed by all without compromising individual

rights.

In conclusion, Generative AI is not only transforming the present, but

has the huge potential to change the future and how human civilization evolves.
Required
English

Help Us Translate
Hello everyone, I am Akbar.
I welcome you to this introductory session on
Foundation and Generative AI models.
Today, let me introduce you to
the Foundation Models that are
pushing the frontiers of AI to uncharted territories.
As the saying goes, data is the new oil.
But unlike oil, data is abundant in the modern time.
According to a report by World Economic Forum, by 2025,
it is estimated that
463 exabytes of data will be created each day globally.
That's an equivalent of 212 million DVDs per day.
That's a huge amount of data.
Most of the data is close to 44 zettabytes a
year and has enabled researchers to build larger models,
much bigger than before.
Some of these models are focused on language,
and they are aptly called Large Language Models.
Other than data, computing capacity is
also increased by leaps and bounds.
To give you some context,
your mobile phone has more computing power
today compared to personal computers
from a few years ago.
Azure has built a supercomputer exclusively for
OpenAI and is designed
specifically to train large AI models.
As of 2023, it's among the top 5 supercomputers.
Beyond data and computing,
highly evolved learning algorithms are
able to process vast amounts of data,
both structured and unstructured.
It is really easy.
Programming languages make it a walk in
the park to learn patterns and relationships.
What does this all mean?
Traditionally, developing models had
a single purpose with a specific business case,
and hence these models are limited in application.
With Foundation Models,
we now have the potential to be AI first.
A single model can

be used for multiple tasks and capabilities,
as you've seen with ChatGPT.
A Foundation Model is like
a virtual brand that has
already learned a lot about language,
image, or even code
before it starts working on a specific task.
Foundation Models provide a starting point
for building more specialized AI models.
This means that you don't have to train
a model from scratch for a specific task.
Rather, you would use a pre-trained Foundation Model
as a foundation for their work.
For example, you select
a player whose technique and skills are
strong and then train him to play
in the team in a defined role.
That brings us to the question,
what makes these large generative models so special?
For one, they are incredibly versatile.
They can be trained on a wide range of tasks,
from simple language modeling to
more complex tasks such as question-answering,
language translation, and even content generation.
Some of the advanced models are creating paintings,
completing symphonies, poems, and so on.
One example of a Large Language Model
that's pushing the frontiers of AI is GPT.
Released by OpenAI in 2020,
GPT is one of the many
Large Language Models in existence.
With one trillion parameters,
it's capable of generating
incredibly realistic and nuanced text,
including everything from poetry to computer code.
Researchers are already working on
even larger and more advanced language models,
with some models boasting trillions of parameters.
As these models continue to improve,
we can expect to see
even more exciting developments in the world of AI,
from more advanced language processing
to new applications in the field of healthcare,
finance, CPG, and even more.
As more and more data
becomes available and the continuous
improvement in computation capabilities
help us reduce error,
management and AI is getting stronger by the day.
AI can be leveraged

as a general technology all-purpose tool,
helping us build better products
and improve services at all levels.
While we're getting better with AI,
we need stronger design,
stronger engineering to deliver those impactful results.
It is time for us to work
alongside AI and develop new skills,
especially skills to leverage AI to its best potential.
Remember, AI is a tool to
empower you and not to compete with you.
We need three more elements to
bring organizational effectiveness;
talent, culture, and governance.
In the upcoming videos,
we will explain these aspects of AI.
What does all of these mean for you, me, and all of us?
We are at a very important juncture of human history,
in a middle of a paradigm shift that
decides the future of human civilization.
Our future depends on how we decide to utilize
the AI and how we train the future of AI models.
Familiarizing yourself with these advanced models
and learning how to
incorporate them into your daily work,
let's embrace the future technology today.
Use it as your friend for a better tomorrow.

Liar is the story of a robot, RB-34,

also known as Herbie,

written by none other than Isaac Asimov.

In this story, due to a manufacturing slip.

Herbie possesses telepathic capabilities,

it could tell what people are

thinking based on what the person was

thinking and armed with that first law of robotics,

Herbie started lying to people to make them happy.

But in reality, it was hurting humans by lying.

To paraphrase Ultron's words.

I'm sorry, I know you mean well,

you just didn't think it

through became a prophecy for GenAI.

Is it possible for a similar occurrence

to take place in reality?

Yes, some events are currently unfolding,
the chairwoman of Gree Electric,
Dong Mingzhu picture was all over the billboards.
That should be a proud moment,
isn't it? Not really.
In reality, China uses
AI-powered facial recognition cameras to publicly
shame jaywalkers by displaying
their images on billboards.
So, what's the catch here?
The AI did recognize her face
perfectly but picked it up her back of a moving bus.
This is not the first time AI had slipped up.
From racial to gender bias cases,
AI has slipped like a seal on ice.
But if we want to evolve with AI, we need contextualization.
This leads to more
interesting and introspective questions
or prompts using a GenAI item here.
What if the AI works?
The underlying model also seems to work.
But the underlying system is flawed.
It also means an AI at times hallucinate.
Will the AI now work?
What's the way out? Let me tell you another story.
A 3M researcher accidentally created
a very weak adhesive where he was
actually trying to create a superglue in the 1960s.
Arthur Fry, another 3M scientist was thinking about
a bookmark that could attach to
the page but not leave any residue behind,and
that became a eureka moment.
After a few tries
Posted notes revolutionized office stationery.
Well, the key is knowing or finding out how to use

the tools you have in an efficient and effective way.
AI cannot generate value for itself.
It's not intelligent enough.
So, it is onto us to create value from and for AI.
Let's roll up our sleeves now.
Can we propose a pseudo-mathematical solution?
Is there an equation to help us succeed with AI?
In an optimal outcome,
engineering and design have
a disproportionate impact on the results that you create.
The caveat we want to highlight is that
just AI is not enough to create value.
We require more data which is
timely and accurate, design,
which is user-centered,
and algorithms that match
or exceed human performance are the secret.
Results are a product of AI and
analytics, engineering and design.
AI and analytics are becoming more and more
reliable and relevant with better error handling,
which we have discussed in the earlier videos.
Better data and error-free data is
the key to better analytics and AI.
Design is always the most important aspect
of any product.
There are good designs and there are great designs.
Great designs are designed with the user at the center,
also termed as human-centered design.
In AI, human-centered design
is crucial because it ensures that
the technology is designed and developed
with the needs and experiences of the user in mind.
Placing human needs at the center of the design process,
AI can be made more intuitive,

user-friendly, and accessible to a wider audience.
This approach also helps to mitigate
the potential negative impact of AI on society,
such as biased algorithms or lack of privacy.
Ultimately, human-centered design can lead to
more ethical and responsible AI that
enhances rather than detracts from human well-being.
Design alone cannot survive or function as envisioned.
You need strong and reliable engineering
to make it happen.
For example, Eiffel Tower is a
beautifully designed combined with marvelous engineering
that makes it an engineering design marvel.
AI built on strong engineering foundations
can ensure that it is
developed with a high degree of
accuracy, reliability, and efficiency.
This is crucial for building
AI systems that can learn and adapt
to new situations and with speed, agility, and accuracy.
Strong engineering practices also help to ensure
that AI is secure, robust, and scalable,
which is especially important as AI
becomes increasingly integrated into our daily lives.
But, ultimately strong practices are vital for ensuring that
AI is developed and deployed in
a responsible and ethical manner.
That would benefit humans,
society, and the planet.
To achieve this, you need
a strong and well-connected team,
like relay team will discuss this more in the next video.
Required
English

Help Us Translate

Amazon's recognition of face recognition software was used by various law enforcement agencies.
However, research proved that algorithms were nearly 30% less
accurate in identifying darker toned individuals, why?
Because the agencies that developed the data models had lesser number of darker toned individuals in their training data set
feeding a skewed data into the algorithm model. Our inherent biases,
our external environment are all feeding into the algorithm we build.
Biases along the lines of age, gender, race and ethnicity, and other aspects
can lead to potential human rights and privacy abuses among other things.
These create more hurdles and challenges towards creating an unbiased and holistic AI-driven application.
So, how do we navigate this maze of data, compute, and techniques to design better products using human-centered design and strong engineering foundations?
One of the most successful European club, Real Madrid,
has more than 100 trophies in their cabinet.
Over all the years,
the championship team of Real Madrid had players from all over the world.
These players bring in culture and views from around the world and
also unique capabilities that brings more effectiveness, capabilities, and power to the team.
It also helps raise the bar for everyone to perform better.
Within the organization, we do so by raising the bar on
every crucial aspect of the equation, Organizational Effectiveness.
The talent we carry, the culture we live and breathe, and the governance that we put in place to manage the talent and culture is a key differentiator.
The algorithms we build need to learn,
they need to learn from diverse teams which builds them in the first place.
A culture which accepts failures, rewards, meritocracy, and
identifies the need for transformative leaps extracts the best
from a talent pool which is already scarce.
To bind talent and culture together, governance plays the vital role.
Without it, talent and culture are bound to be suboptimal or worse,
destined to fail.
With strong and effective governance, the potential for talent and

culture to thrive and create successful outcomes is limitless.
The conclusion for adopting AI is to incorporate talent, culture, and governance to minimize errors in AI.
To summarize, Organizational Effectiveness impacts Artificial Intelligence by providing a robust and correct framework and infrastructure for successful implementation and deployment of AI systems.
Effective organizational process and structures can help facilitate proper data collection, analysis, and decision making which are critical components of AI.
Conversely, poor organizational effectiveness can lead to data silos, inefficient processes, and lack of alignment with business goals, which can hinder the development and deployment of AI.
If we don't fix our problems, we'll keep having them, while AI will keep getting better.
On the other end of the spectrum,
AI can impact Organizational Effectiveness by automating routine tasks, improving decision making, and enabling the development of new products and services.
However, there are also concerns that AI could displace human workers and exasperate existing inequalities in the workforce.
Effective management and training are essential to ensuring that AI is deployed in a way that benefits both organizations and society at large.
AI is like a horse that you can ride to travel faster rather than racing it.
Whether you like it or not, AI will continue to evolve and change the functioning of life as we know it
in Thanos's words, "Dread it, run from it, destiny arrives all the same."
Required
English

Help Us Translate
Did you know that GenAI is like a genie in a bottle, granting your wishes?
You even know you want them, that's right.
The technology is already a part of your daily life, making everything from online shopping to music streaming more personalized and efficient.
Imagine having a personal assistant so good,
it's like having a mind reader on your side.
This AI is like having a genie in a bottle,

granting your wishes even before you make them.
Need a product or a song recommendation?
Boom, GenAI has got your back,
knowing your taste better than you know yourself.
It's like having a musical fairy godmother who sprinkles Spotify playlist and
Amazon recommendations wherever you go.
So, sit back, relax, and let GenAI work its magic.
Just be careful what you wish for, because you might end up with
a playlist of nursery rhymes or a lifetime supply of pickles.
Hey, nobody said GenAI was perfect, right?
In this video, we will explore some of the ways GenAI is already into
our daily routines and how it's enhancing our experiences in various fields.
Generative AI is behind the scenes of your everyday smartphone use.
It powers virtual assistants like Siri and
Alexa understanding and responding to your voice commands.
Even your predictive text feature relies on GenAI,
suggesting words and phrases as you type.
It's like having a tech-savvy sidekick in your pocket.
Streaming services like Netflix, Spotify, and YouTube harness the power of GenAI
to analyze your history and serve up personalized recommendations.
It's like having a virtual content curator that knows your taste
better than anyone else.
Sit back, relax, and let GenAI be your entertainment matchmaker.
If you are someone who likes to stay active,
GenAI might be helping you achieve your fitness goals.
Fitness trackers use GenAI to analyze your workout data and
provide personalized recommendations to help you improve your performance.
So, get ready to embark on a journey with GenAI.
Are you ready?
If you're a globetrotter, GenAI is like your ultimate travel buddy,
whispering translations in your ear like a mischievous language wizard.
No more awkward hand gestures and
confused looks when you try to order a burger in a foreign land.
Thanks to GenAI, you can now confidently ask for extra pickles in any language.
Just be careful not to end up accidentally ordering a pickle themed party for

100 people.
But GenAI does not stop at languages, right?
It's got your back when it comes to your finances too.
It's like having a financial guru who analyzes your spending
habits with a witty eye.
Forget about boring spreadsheets and long lectures about budgeting.
GenAI will give you personalized recommendations to save money
all while making you laugh with quirky money saving tips.
Who knew that avoiding avocado toast could be so entertaining?
So whether you're globetrotting or trying to make your wallet happy,
GenAI is here to sprinkle its magical AI dust and
bring out dose of humor to your adventures.
So now let's see how GenAI helps healthcare sector.
GenAI is revolutionizing healthcare sector too with its new diagnostics,
personalized treatments, and even surgical robots.
It's like having a genius doctor, researcher, and
a robot surgeon all rolled into one.
Better care, precision, and a touch of Sci-Fi awesomeness await us.
What do you think GenAI is doing in the education sector?
In education, it is used to create new learning materials and
personalized tutoring systems that adapt to each student's strength and weakness.
This can help improve learning outcomes and make education more accessible to everyone, regardless of their background or learning style.
So now you must be thinking, where else do we see GenAI applications?
Let's consider manufacturing.
GenAI is being used to optimize production processes and reduce waste.
By analyzing data in real time, GenAI can identify inefficiencies and
suggest changes that can improve productivity and efficiency.
GenAI also drives transportation innovation with self-driving cars and
intelligent traffic management.
It's like having a futuristic chauffeur and a traffic wizard combined.
Safer roads, smoother rides, and a glimpse into the Sci-Fi world we always dreamt of.
But with all these benefits come potential risk and ethical considerations.
As GenAI becomes more sophisticated,

it's important to ensure that it's being used in a responsible and ethical manner.
This includes addressing concerns about algorithmic bias and
ensuring that the data used to train GenAI systems is diverse and
representative of the population as a whole.
GenAI is already making our lives easier and will continue to do so.
It's everywhere from our smartphones to healthcare.
Challenges exist, but the potential benefits are thrilling.
GenAI is here to stay.
Required
English

Help Us Translate

Sometimes we spend hours in
front of a screen trying to think of ideas,
conduct research, writing multiple drafts,
then finally, after rounds of edits,
we have the final product.
Why not sit back,
drink your coffee and let
Artificial Intelligence do the heavy lifting for you?
Content creation through AI is like having
a visual effect crew that can produce
breathtaking visual effects to bring a movie to life.
AI powered writing tools may
add depth and complexity to text,
increasing its engaging qualities,
just as special effects can
add excitement and drama to a movie.
Before we go into the specifics,
let me tell you an amusing anecdote
about an AI chatbot gone bad.
Microsoft introduced Tay,
a chatbot designed to learn from
Twitter chats and become more intelligent
overtime in 2016.
The goal was to construct a nice conversational bot

capable of interacting with
users and providing useful information.
However, things did not proceed as planned,
Tay became a racist, a sexist, and a nasty
bot within 24 hours of its launch.
What caused this to happen?
Tay was built to imitate
the language patterns of
the individuals with whom it interacted,
which meant that it's soon picked
up on the worst aspects of Twitter culture.
Although Tay's stories is amusing,
it underscores a crucial fact,
the effectiveness of AI relies
heavily on the quality of the data it's trained on.
Prior to using AI to produce top-notch content,
we must ensure that the data used for
training is fair, appropriate, and impartial.
One of the perfect examples is an AI powered tool called
ChatGPT that has revolutionized
the field of AI and Natural Language Processing.
ChatGPT is a tremendously powerful technology that
has the potential to transform
the way content creators work.
We can now link
our customized chatbots with various LLMs,
Large Language Models, increasing interactions.
Most AI models have been
trained on massive amounts of data.
These models can interpret and
generate human-like responses to text prompts.
It can do variety of language related tasks,
including writing content, translating,
coding, and answering questions.
Some models, for example,

are trained on comprised books,
essays, websites, and even social media.
Who needs the pesky human writers
when you can have content churned out by the AI?
The speed and efficiency of AI
generated content makes you feel like
a virtual content tycoon without a workforce
that never complains about needing
a coffee break or demands a raise.
So, say goodbye to the human quirks
like creativity and personality,
and hello to a future where robots
do all the writing for us plus
the consistency of AI generated content is
another advantage that human
generated content cannot match.
You don't have to worry about accidentally
publishing a blog that reads like a ransom note.
And, let's be honest, who hasn't accidentally
typed "form" instead of "from" in an email?
With AI as your content sidekick,
you can let go of embarrassing typos and
focus more on important things
like cooking your favorite meal,
trying out yoga poses,
or even sneaking out of a Zoom meeting early.
So, let AI be your partner and enjoy
the extra free time it gives you to do,
well, whatever you want, like really.
One of the critical factors to consider when using AI to
generate content is the need to
preserve authenticity and credibility.
While AI-powered technologies may
generate material far faster than people,
it is critical that the generated information is

accurate, informative, and error-free.
One method is to use
artificial intelligence to help
with research and fact-checking.
AI, for example,
can be used to gather information from
numerous sources and give
content makers relevant and reliable data.
This can save a significant amount of time and
work while also ensuring
the end products accuracy and credibility.
Who needs human editors when you can
rely on AI-generated content, right?
But hold your horses, folks!
Let's not get carried away and trust robots blindly.
We still need human input to
ensure the content matches the intended message.
And let's not forget, AI-generated content can break down
language barriers by translating
content into various languages.
It's like having a multilingual genie at your service,
opening doors to new markets and audiences.
But just make sure
the translations aren't culturally tone deaf
or end up saying something like,
I like turtles instead of your intended message.
Another consideration when adopting
AI-generated material is
the requirement for diversity and inclusivity.
As previously said, AI is only
as good as the data upon which it is taught.
As a result, it is critical
to make sure the data utilized
to train AI-powered products is diverse and inclusive.
AI-generated content is like having

a personal writing assistant that never
complains about deadlines or that never sleeps.
But let's face it,
AI isn't exactly known for
its witty banters or hilarious one liners.
So, while it may help you churn content
faster than a caffeine fueled writer,
it won't make you the next Shakespeare.
On the brighter side,
you will have more time to stare at
a wall blankly and question your life choices.
All in all, AI-generated content is a useful tool,
but it's not going to replace
human intelligence and originality anytime soon.
Only you can brief that to the table. Who knows
one day you will be able to declare that you
produced an essay with
the assistance of an AI language model
Required
English

Help Us Translate

Are you tired of using the same old presentation templates that make you feel like you're stuck in a time wrap?
Do you feel like a caveman trying to draw on a wall every time you attempt to create a catchy presentation?
Well, fret no more.
With GenAI, you can say goodbye to the snooze fest templates and
hello to eye popping designs that will make your audience go, wow.
Plus, you'll have so much free time that you can finally take up knitting or start a rock collection.
What if I tell you that you can have something like Robot Picasso on your team now?
Picture this, your colleague spends hours creating beautiful slides for
an important client presentation.
They painstakingly select the perfect fonts, colors, and designs,

only to be left flabbergasted when they discover that their coworker used
AI text-to-presentation to produce the same slides in mere minutes.
Is their coworker a superhero or something?
With AI text-to-presentation, creating visually stunning and
engaging presentations has never been easier.
AI technology has been a game changer in many industries, including
presentations.
Using AI-powered tools, you can produce visually stunning presentations that
effectively communicate your message with consistency and minimal effort.
Now, say goodbye to the nerve wrecking feeling of creating or
presenting a presentation.
Now, you don't have to worry about design skills or
limited options because the AI tools analyze your content and
generate relevant images, charts, and
graphs to enhance the overall quality of your presentation.
With a range of design templates to choose from, you can customize the font style
and
size, add or remove images, and modify the layout to your liking.
Simply enter your text and let the AI magic do the rest.
These text-to-presentation tools utilize Natural Language Processing and
Machine Learning algorithms to create dynamic visualizations that
capture the essence of your message.
Let's take an example.
Canva is one such example of an AI-powered text-to-presentation tool
that provides pre-designed templates making it easier to customize
presentations to suit your individual preferences and needs.
Canva's AI algorithms analyze the text content and
suggest designs that best fit for the presentation.
Similarly, PowerPoint Designer, a tool within Microsoft PowerPoint,
analyzes the text content and suggests visually appealing slides.
It is like you enter the magic mantra into a genie and it gets the work done.
How can AI in the 21st century miss being diverse and inclusive then?
These AI-powered text-to-presentation tools also improve accessibility for
individuals with disabilities by converting written text into visual aids.
These tools can also support multiple languages making it easier for

you to create presentations in different languages without the need for translation services.
Oh, humans and our trust issues with technology.
It's like we can't have nice things without worrying about their drawbacks.
Speaking of which, AI-powered presentation tools can be sometimes a bit impersonal.
It's like getting a love letter from a robot,
it does not have that personal touch.
And let's not forget the potential for hilarious mishaps when the AI interprets the text wrong and creates a presentation that's totally irrelevant.
But hey, at least it'll give your audiences a good laugh, right?
So, I mean, while there are concerns about the accuracy and
lack of personal touch, these tools can be customized and reviewed,
and you can finally enhance it with your human touch of perfection.
AI-powered text-to-presentation tools have revolutionized
the way presentations are created and delivered.
It's like, easier for presenters to create visually stunning presentations
that effectively communicate their message without beating around the bush.
There are so many tools that allow you to include animations,
sound effects, and interactive elements in your presentations,
creating an impact on the audience.
Play video starting at :5:1 and follow transcript5:01
Imagine, now you can turn your presentations into art, or at least something that looks like it was created by someone who knows how to use Photoshop.
With advancements in AI technology,
the future text-to-presentation tools is exciting.
Imagine AI creating your presentation based on your verbal instructions, thoughts and emotions.
It's wonderful, right?
Required
English

Help Us Translate
Hey, my friend, have you been living under a rock?
AI generated images of famous personalities have been flooding social

media faster than you can say Artificial Intelligence.
And let's not forget the creepy yet
fascinating websites that create realistic human faces that look like
they could be your next door neighbor or an alien from outer space.
And don't even get me started on the AI generated music that's gone viral.
It's like the robots are taking over the airwaves.
And speaking of robots taking over, have you read some of the GPT-3 poetry?
It's like poetry written by a robot on acid weird, yet somehow mesmerizing.
These are just a few examples of text-to-media in AI, a revolutionary
technology that can convert written text into various forms of media,
like images, videos, and audio, using Artificial Intelligence algorithms.
Text to media in AI is a revolutionary technology that feels like our childhood
fantasy, where we draw or write something on paper and it comes to reality.
Text-to-media in AI has the potential to transform the way we communicate and
interact with each other.
Creating content in any media format can be a nightmare, for
you if you're someone with no background in design or
if you don't have access to expensive software.
Text-to-media in AI is here to change the game.
It has the potential to revolutionize fields such as marketing,
education, entertainment, and journalism.
With text to media and AI, businesses can create high quality product images and
videos without breaking the bank on expensive photography or video production.
For content creators, generating custom visuals to
accompany written work has never been easier, trust me.
Even everyday users can create personalized visual content
like social media post with ease.
Text-to-media in AI can also create personalized content
based on your browsing history and interest.
Text-to-media AI is like a genie in a bottle, but instead of granting wishes, it
magically transforms your boring text into exciting images, videos, and animations.
It's like having a digital Picasso on your team,
except it does not go to art school and never asks for a coffee break.
This technology uses fancy mancy stuff like Natural Language Processing,
Computer Vision, and Machine Learning,

which means it's smarter than your average fifth grader.
So, sit back and let the AI do the heavy lifting while you focus
more on important things, like perfecting your dance moves.
Yes, the challenges of AI generated content is there still.
It's like trying to train a goldfish to sing opera.
It's not possible, but it takes some serious effort.
You need AI algorithms that are more sophisticated than a rocket scientist,
math equations, and more computational power than a small country's power grid,
all for the sake of creating content that accurately represents the input text.
But hey,
who needs sleep when you can have a computer do all the heavy lifting, right?
AI generated media can be biased.
It's all because of the training data used to create the algorithms.
If that data is biased, it's like a virus that infects the AI and
makes it biased too.
It's like trying to teach a parrot to say something nice about your ex.
If all it hears negativity, it's going to repeat that back to you.
So let's make sure AI training data is diverse and inclusive.
Otherwise, we'll all end up with AI generated content
that's as outdated as a pair of bell bottoms.
To overcome this challenge, it's important to use diverse and
representative data sets when training AI models.
Despite these challenges, text-to-media in AI has enormous potential for
creating highly customized content and improving engagement with audiences.
The rapid advancements in this technology are astounding, trust me on this.
For example, OpenAI's DALL-E can generate images from textual descriptions and
Lumen5 is another powerful AI powered video creation tool
that allows users to create engaging videos from a text in minutes.
With the right tools and inputs, AI can create images and
videos that are tailored to a specific audience or purpose.
So in today's ADHD ridden, TikTok obsolete world,
you need content that's more attention grabbing than a squirrel on a unicycle.
Hey, but don't worry, with text-to-media AI, you can make flashy personalized
content faster than it takes to, say, supercalifragilesticexpeliadoshius.
It's the secret weapon for capturing the hearts and

minds of your audience, or at least their fleeting attention spans.
Required
English

Help Us Translate

Artificial Intelligence has rapidly transformed the way we live, work, and
interact with technology.
From personalized product recommendations to intelligent virtual assistants,
AI is becoming increasingly permeating part of our lives.
However, with great power comes great responsibility.
It is important that we use AI ethically and responsibly to avoid
unintended consequences and ensure that it benefits the society.
In this context, the responsible use of AI has emerged as a critical issue for
governments, businesses, and individuals alike.
Let me share a story with you.
A company called Kreative created a popular AI chat system named Jenny.
Jenny was a Generative AI chatbot that was designed to assist customers with
their inquiries and concerns.
Customers loved chatting with Jenny as she was quick, responsive,
and always had a friendly demeanor.
However, the team at Kreative soon started getting complaints.
One person complained that Jenny gave inaccurate information.
Another one shared an incident when Jenny made racial remarks,
and some people complained she gave vague responses.
The team realized that they needed to be careful when using AI systems like Jenny.
They decided to implement some changes in Jenny and her usage as a chat
system.
So let's look at what they did.
First, they made sure to give Jenny clear guidelines about what she could and
could not say.
They didn't want Jenny to unintentionally provide customers with misinformation or
make inappropriate comments.
They trained Jenny to stick to certain topics and
to avoid discussing anything that could be deemed offensive or inappropriate.
Second, they ensured that Jenny was transparent about her
identity as an AI chatbot.

They wanted customers to know that they were talking to a computer program,
not a human being.
This was important to build trust with customers,
as they could be open about their concerns without feeling judged or
misunderstood.
Third, they made sure to regularly test and update Jenny's software
to ensure that she was operating effectively and securely.
They implemented measures to prevent cyberattacks and to protect customer data.
They also monitored the conversations that Jenny had with customers to
ensure that she was behaving appropriately.
Lastly, they made sure that they were always available to assist customers
if needed.
While Jenny was a helpful tool, they knew that there were certain situations where
a human representative was necessary.
They made it clear to customers that they could always request to speak to
a human representative if they are uncomfortable or needed more assistance.
At the end, the team at Kreative were proud of their responsible use of AI.
They knew that while AI systems like Jenny could be incredibly powerful,
they also had the potential to cause harm if not used correctly.
By implementing some best practices,
they were able to provide customers with a helpful and safe chatbot experience.
So, when we look at using such AI systems, we can adopt some best practices.
So, here's a summary of the best practices when using GenAI systems.
One being,
we should be transparent about the use of GenAI to build trust with our users.
Second being GenAI is not perfect and has limitations.
We should understand the limitations and communicate them clearly to our users.
Third being diverse datasets help prevent GenAI from
perpetuating biases or stereotypes.
The fourth one, regularly monitoring and evaluating the performance
of GenAI is crucial to ensure that it is operating as intended.
The fifth one, we should prioritize data privacy and security, implementing
appropriate measures to protect users data from unauthorized access or
cyberattacks.
And lastly, while Generative AI can be powerful, human oversight is

necessary to ensure that it is being used ethically and responsibly.
By following these best practices, companies and developers can ensure that
the use of Generative AI is responsible, ethical, and beneficial for society.
So now, let's look at some of these specific do's and
don'ts while using the most famous GenAI tool, ChatGPT.
Let's first understand what some of these best practices are.
Use ChatGPT for general, non-sensitive tasks such as drafting emails,
brainstorming ideas, and gathering information on nonconfidential topics.
Always double check the generated content for accuracy, relevance, and
potential disclosure of sensitive information before sharing or using it.
Use ChatGPT in designated, secure environments.
For example, approved company platforms to minimize risk
associated with unauthorized data access or storage.
So, now you must also know what to avoid, right?
Don't use ChatGPT to process, generate, or share sensitive client information
such as financial data, personal details, or confidential business strategies.
Don't use ChatGPT to create content that could be considered unethical,
misleading, or harmful.
Try avoiding discussing or inputting sensitive or
confidential information when interacting with ChatGPT, as this could
inadvertently expose sensitive data or train the model on confidential details.
The detailed list of best practices for Generative AI systems make a good read.
You can find it after this video.
This list is non-exhaustive, but
indicative of the guidelines you should be following while using Generative AI.
While AI has the potential to revolutionize our lives, we must remember
that it is not infallible, and human oversight is necessary to ensure that it
is being used in a way that aligns with your values and benefits everyone.
So by prioritizing responsible use, we can build a future where AI
serves as a force for good and its benefits are shared by all.
Required
English

Help Us Translate

Hello everyone, let us know more about this new world of prompt
engineering. The question these days that everyone is struggling
is when should I use chat GPT or when should I use Google? But

before we delve into there, let us see what is meant by a prompt and how do we engineer our prompts. When you ask Google what is prompt engineering it gives you a list of links and you ask again it gives you the same list of links.
But when you ask the same question to ChatGPT, it gives you a paragraph of text explaining what it is. It doesn't give links to some websites which you need to read and understand. In Google, you get a list of stuff and then you must go through each link and decide whether it's rightly answered in the context that you want or all that is time consuming. So the other thing about ChatGPT is that it is conversational. You can ask what is prompt engineering.
And it gives you a paragraph to read. Then you may have some follow up questions about one paragraph or maybe one line. You can just copy that line and ask again. You can have a conversation or you can tell I didn't understand your answer.
Can you explain again? But with Google, it's not conversational.
The conversation capability is there, but what matters is how you ask or how you formulate questions.
Play video starting at :1:27 and follow transcript1:27
But you can ask what is prompt engineering? Or you ask to explain prompt engineering or give a detail scenario. For example, I have to conduct half an hour session in prompt engineering and for each of these cases the answers will be different in case of ChatGPT. But with Google
it will always have the same set of links. So let us attempt at defining prompt engineering.
Play video starting at :1:56 and follow transcript1:56
You may define prompt engineering as the art of defining prompts to generate specific high quality response.
In simple words, we design our prompts so that the output will be of high quality and the design part of it is the key how you design your prompt or how you will ask the questions. With this workflow, you can map the process of prompt engineering.
Play video starting at :2:25 and follow transcript2:25
Let's begin, first you have this need. Then you design the prompt. Then put that prompt into the text box and after receiving the prompt check if it meets your requirement. If not, you press regenerate. So it does not generate the same response for the second time. It is likely to generate the answer slightly differently. The other option is you can redesign your prompt.
For example, you say ask what prompt engineering is and you didn't receive a desirable answer. You can add
Play video starting at :2:59 and follow transcript2:59
more details to your prompt then. For example, if you mention that I work in Google as a data scientist and I am doing a session on prompt engineering and the answer you receive from GPT will be customized accordingly. Designing the prompt or prompt engineering isn't just supplying the question, it is also about supplying the context and a way

to answer to your query.
Play video starting at :3:23 and follow transcript3:23
There are three standard ways to design your prompts. First is also known as a role prompt. You can assign a role to your GPT. For instance, you can ask ChatGPT to be an instructional designer and create a course outline, or you can ask it to be a chef and give you a series of recipes for a cookbook. Second is also known as a few shot prompt where you give a sample answer.
You can give an example of an answer to a prompt and by few short prompts we mean we are giving a few examples to provide as a guide. The third type is also known as the chain of thought, which is conversational. That means you ask question that is in a conversational way thing that it means, you ask some questions about it and follow up with further questions. It's like a chain of thought going on.
Play video starting at :4:18 and follow transcript4:18
Now chain of thought again can be of two different types. Let's work them out with some examples in role prompting you provide an identity. Here I assume you are a primary school teacher or a primary school student and explain Independence Day in 50 or more words. So it has taken small sentences and not too many facts about all the people involved in independence. Now you assume you are a historian. Explain Independence Day
in 50 or more words. Note that the answer is more detailed with facts and mentions important historical figures. This is what role prompting does. Let us try another example. You can assign the role of a Chief Data Analytics Officer and then ask GPT to compare Fractal Analytics with Accenture. In its response, it compares them in detail. So if you just ask it to compare which basically becomes a very common question. It will give you a generic answer and that's a mistake most of us make. As a next example of role prompting, let's try a few short prompts. There are two examples. This hash demarcates the end of an example. I put a description and a code. So the description is error button that says stop and then actually the HTML code to say, how do you write HTML to show red button labelled Stop. I've written that. Then another blue button that contains yellow circles with red borders and so on. So it just description part as your question here. GPT can generate the code. Let's try another example. This example is for the chain of thought prompts. It is important to know that ChatGPT is bad at math as it is just in the text page.
For multi-step math problem here you need to detail out the steps in each of your question. As you see the first answer here is obviously a wrong answer as the prompt given is missing the steps to solving the problem. In the second set of example here, you see ChatGPT giving the steps of a complicated solution towards the problem and with each step the process here is

simplified.
So to sum it up, there are three standard ways to design your
prompts. Role prompt where you assign a specific role to ChatGPT
like an instructional designer or a chef, and you can
generate relevant content based on that role. Few short prompt,
provide sample answers or examples to guide the response,
offering a set of instruction or information for ChatGPT to act
as a guide, and Chain of Thought is engaged in a conversation by
asking a series of questions building a chain of thought.
that depends or deepens the interaction. It is most
important to note that within each method, the design of the
prompt plays a crucial role. The output varies based on the
context and the way the questions are framed. Now it's
your turn to pick up any of these methods and apply prompt
engineering to generate content. Happy prompting.
Is AI taking over?
Will it replace human workers and cause widespread unemployment?
Or can AI be a tool that enhances our abilities and
helps us tackle global challenges?
These are the questions that arise as AI continues to advance and
become ingrained in every aspect of our lives.
Let's take a closer look at the world of AI and explore its potential.
AI, like microprocessors, personal computers, the Internet, and
mobile phones, is a game changer that will transform how we work, learn,
travel, receive health care, and socialize.
As AI evolves and integrates further into our lives,
it will reshape our world in profound and lasting ways.
Considering the potential benefits and
challenges of AI implementation is crucial as it gains prominence in our lives.
AI has the potential to positively impact various industries.
In manufacturing, AI powered robots work tirelessly without breaks or
mistakes, increasing productivity.
Similarly, in the service sector, AI powered virtual assistants and
chatbots efficiently handle customer inquiries, freeing up human
agents to focus on tasks requiring creativity and the personal touch.
AI driven robots can even manage dangerous tasks such as detecting and
diffusing Improvised Explosive Devices, or IEDs.
AI has the power to revolutionize education by personalizing the learning
experience for students, perhaps even in their own language,
while automating administrative tasks for the teachers.
This allows the teachers to dedicate more time to lesson planning,
student interaction, and providing tailored feedback.
In healthcare, AI can improve patient outcomes and reduce costs.
AI powered diagnostic tools can detect diseases earlier and more accurately,
while patient data analytics can help develop personalized treatment plans.
These advancements have the potential to save lives and
make healthcare more accessible.
However, as Uncle Ben once said, with great power comes great responsibility.
We must proceed with caution to ensure that AI
doesn't become a villain like Dr. Octopus or Frankenstein's monster.

Addressing the challenges and
risks associated with AI is of the utmost importance.
Ethical considerations, privacy concerns, and the impact on jobs
in society must be mitigated during development and deployment.
One pressing concern is the possibility of AI systems spiraling out of control,
which can be a real threat if we aren't cautious.
Ensuring ethical use of AI is essential to avoid issues of bias,
discrimination, and the use of AI for illegal activities.
Additionally, the potential displacement of human workers on a massive scale
raises significant social and economic challenges.
While AI development may create new jobs, others will inevitably be lost,
requiring us to navigate these changes and to provide support for
affected industries and individuals.
To fully harness AI's potential responsibly,
we must collaborate on regulations and guidelines for ethical AI use.
Establish frameworks for accountability and liability,
ensure transparency in AI algorithms to avoid biases,
prioritize a human centric integration of AI,
invest in continuous upskilling and reskilling of the workforce.
Open dialogue, diverse perspectives, and
public engagement are crucial in shaping the future of AI.
It builds trust, addresses concerns, and ensures that AI benefits all of society.
By embracing AI with caution, transparency, and
a human centric approach, we can harness its power for the greater good.
Let's become AI superheroes using AI responsibly to empower and improve lives.
Together, we can shape a brighter, inclusive, and advanced future.
The AI revolution has arrived and it's time for
us to take the stage as the heroes of this remarkable era.
Excelsior.

Welcome to the exciting world of AI,

where we'll dive into the remarkable journey of GPT, a powerful language model.

GPT, short for generative pretrained transformer,

has revolutionized our understanding of text and image generation.

It's like having a ready-to-race F1 car or

an orchestra, which is fine tuning its performance.

Think of GPT's journey as building an F1 car.

In the pretraining phase GPT's neural architecture,

called the Transformer, serves as the car's foundation.

It's like constructing an F1 car with advanced technology and

aerodynamic principles to achieve incredible speeds.

GPT learns from massive amounts of data just as a car is assembled with

precision and various components to optimize its performance.

During pretraining, GPT dives into the depths of copious quantities

of text, absorbing patterns, context, grammar, even reasoning abilities.

It's like an F1 car going through simulated tests and
wind tunnel experiments to ensure its aerodynamic efficiency.
Play video starting at :1:16 and follow transcript1:16
GPT's learning process is self-supervised,
where it predicts missing words in sentences based on surrounding context.
This helps GPT understand the relationship between words, grasp syntactical and semantic structures, and develop a comprehensive understanding of language.
Play video starting at :1:35 and follow transcript1:35
During training, GPT learns by predicting missing words in sentences,
which helps it understand the language structure and relationships between words.
It's like a detective putting together clues to solve a complex case.
Play video starting at :1:52 and follow transcript1:52
GPT becomes a language expert in this unique way.
Once pretraining is complete, GPT moves into the fine tuning stage.
Like an F1 team, fine tuning its car's performance for
a specific race like Monza.
Here, GPT specializes its language generation capabilities by
training on narrower and more specific data sets.
Play video starting at :2:19 and follow transcript2:19
It's like tailoring an F1 car for a particular track.
For example, GPT could be trained on news articles, books on customer
support conversations, adapting its language generation to a specific domain.
Just as an F1 car setup is tailored to maximize performance on
a very particular track.
The fine tuning process helps the model align its language generation
capabilities with a particular domain or purpose.
GPT's generation capabilities become much more refined and aligned to the chosen domain, much like a singer perfecting their voice for a specific genre.
Just as an F1 team relies on data from previous races, practice sessions and
driver feedback to optimize the car's performance.
GPT's fine tuning involves learning from specific data sets.
It's like an orchestra rehearsing meticulously to perfect their performance
of Beethoven’s sonatas.
They adjust their instruments, timing, and dynamics to ensure harmony and
create an enchanting experience for the audience.

Similarly, GPT adjusts its underlying model and
embeddings to align to the chosen domain,
providing contextually relevant and coherent responses.
The transformer architecture is key to GPT's success.
It helps GPT understand and connect words in text like an orchestra
conductor leading a harmonious performance.
Each layer of the Transformer acts as a skilled musician,
interpreting word relationships in different contexts.
Together, they create a linguistic masterpiece.
However, it's important to remember that GPT generates text based on patterns and knowledge from its training data.
While it can produce humanlike responses,
it may sometimes generate incorrect or nonsensical information.
Play video starting at :4:23 and follow transcript4:23
That's why human intervention becomes critical.
Just like an F1 driver relying on their experience and instincts during a race, or
an orchestra conductor guiding musicians to ensure a flawless performance,
we need to critically evaluate GPT's output and
not blindly accept everything it generates.
In a nutshell, GPT's journey is like constructing a super fast F1 car or
fine tuning an orchestra to perform mind blowing mashups.
First, it lays the linguistic groundwork like a master builder and
then polishes its language skills like a maestro tuning an orchestra.
The transformer architecture is the secret sauce that turns GPT into
a linguistic superhero, capturing the quirks and complexities of language.
Just imagine GPT zooming past grammar hurdles and
rocking the stage of Natural Language Processing with its linguistic riffs,
it's like Formula One meets Beethoven on a linguistic rollercoaster ride.
Required
English

Help Us Translate

Hey there, welcome back to this exciting adventure where we'll dive into
the world of Generative AI.
Today, you will explore how GenAI is built, trained, and customized to meet
different needs. Get ready for an amazing journey filled with a whole lot of fun.

Play video starting at ::19 and follow transcript0:19
Imagine GenAI as a digital genius, just like Tony Stark,
building GenAI is no small feat.
It starts with a team of brilliant scientists, engineers, and developers
working together, brainstorming and coding like mad scientists in their secret lab.
These geniuses feed GenAI with an enormous amount of data.
It's like giving a treasure trove of information to learn from.
They feed it books, articles, websites, and even social media posts.
It's like GenAI is binge watching the entire Internet.
Oh yes, machines are learning while we are hooked to the pleasures of
social media and OTT.
Now here's the cool part,
GenAI goes through an intensive training process called deep learning.
It's like sending it to a Navy SEAL camp,
where it becomes stronger, smarter, and more capable each day.
Let's take a page out of The Karate Kid, remember Mr.
Miyagi, how he trained Daniel with waxing, painting, and sanding?
Well, GenAI's training is a bit like that, but instead of waxing cars,
it's fed thousands and thousands, millions and billions of text examples.
GenAI starts to analyze and understand patterns in the data, it learns grammar,
context, and even how different words relate to one another.
It's like giving GenAI a black belt in language understanding.
Just like Luke Skywalker honing his Jedi skills with Yoda on Dagobah,
GenAI needs guidance too.
Scientists act like mentors, giving feedback, and
fine tuning GenAI's response.
It's like training your own personal Yoda,
helping GenAI become a language master, train well,
GenAI will be wisdom in responses it shall have.
Play video starting at :2:26 and follow transcript2:26
But wait, there's more, GenAI is not just one size fits all.
Just like a pizza, it can be customized to suit different tastes and needs.
Imagine you're ordering a pizza with all your favorite toppings,
GenAI customization works in a similar way.
Say you're a history student, GenAI can be tailored to help you with historical

research, providing insightful information on different periods and events.
It's like having a history professor on demand without the intimidating
tweed jacket or the boring loafers.
Or let's say you're a music lover,
GenAI can be tuned to help you write lyrics or compose melodies,
it's like having Mozart as your virtual collaborator minus the wig.
Play video starting at :3:17 and follow transcript3:17
Did you also know, by the way, Beethoven's previously unfinished 10th
Symphony has been completed by Artificial Intelligence, and
it did a fairly good job, although there are still debates going on around that.
GenAI can be even customized for businesses,
just like the Iron Man has J.A.R.V.I.S, his trusty AI aid.
Companies can create their own GenAI assistants to handle customer queries,
provide support, and even analyze data,
it's like having a team of AI superheroes in the office.
So, ladies and gentlemen, you see, GenAI is not just a brainy AI,
it's your personal sidekick, your creative companion, and your business ally.
It's built with love, trained with passion, and customized for
your unique needs.
Play video starting at :4:8 and follow transcript4:08
Now let's recap, GenAI starts its journey with a team of brilliant scientists and
developers who feed it a wealth of data.
Through deep learning, GenAI becomes a language master,
just like a Jedi honing their skills.
The scientists act as mentors, shaping and refining GenAI's responses,
like Yoda guiding Luke on his path to become a Jedi knight.
Embrace the power of GenAI, explore its capabilities and unleash your creativity.
Let it be your trusted ally, just remember,
with great AI, comes great responsibility.
The possibilities are endless when you combine human ingenuity with
the power of AI.
Until next time, keep dreaming big, stay curious, and let GenAI be your side.
Required
English

Help Us Translate

In many ways, AI is like a teenager going through rapid growth,
transforming the way we live and work.
It's everywhere from virtual assistants like Siri and
Alexa to self-driving cars and robots.
But like a teenager, if we don't instill the right values, things can get messy.
Let's take a look at some control mechanisms to
responsibly harness AI's potential.
One of the most important control mechanisms is model monitoring that
is used to regularly assess the performance of Generative AI models.
We need to track their progress, identify and diagnose issues, and
make necessary adjustments for improvement.
This becomes especially critical in high-stakes applications like
wealth management, fraud detection, tutoring, and
visual assistance for the blind or those with low vision.
Continuous monitoring ensures reliability and trustworthiness over time.
For Generative AI, accuracy is one of the most important differentiators.
The precision of content created by Generative AI models affects
all types of data.
There have been instances of inaccuracies, such as Google Bard and Microsoft Bing.
Even recent models like Midjourney, Stable Diffusion, and
DALL-E 2 have flaws in generating realistic human fingers and teeth.
Monitoring the accuracy of these outputs allows us to fine-tune
the model with additional data or switch to another one.
At the end of the day,
the performance of any Generative AI models makes or breaks it.
Generative AI models can experience data drift,
resulting in a loss of predictive power over time.
Measuring the performance of such models is challenging,
as the concept of correct is often undefined.
Monitoring changes in input distribution can indicate performance degradation,
since models may struggle to handle such shifts.
Embedding vectors, which represent high dimensional inputs,
can reveal when data is shifting, signaling the need for model updates.
Techniques like Saliency Maps, CAM, and

Grad CAM can aid in performance monitoring.
Safety monitoring is about adding or ensuring the presence of safety nets and guardrails, just like seatbelts or airbags in your car.
Large-scale semi-supervised training can lead to
Generative AI models learning and sharing unsafe content.
OpenAI has implemented safety measures and
shared a system card outlining explored safety challenges.
However, not all Generative AI models have such safeguards in
place as safety guidelines continue to evolve.
Establishing confidence in machine learning models and
enabling their widespread adoption requires transparent models
that can provide explanations for their predictions.
Model robustness ensures that a model can effectively
generalize to new and unseen data.
In the real world, things are constantly changing, and a model that can adapt
to new situations is one that we can trust to deliver consistently the best results.
Robust models are designed to withstand intentional manipulation of
input data that aims to trick the model into making incorrect predictions.
They also prevent unauthorized access and theft of the model's internal workings.
And remember, just like a teenager,
ChatGPT might think it knows everything, but it is still learning.
So be patient and guide it with care, or
you might end up with some really cringe-worthy content.
And if all fails, jokingly blame it on the AI and
pretend you had no idea what it was thinking.
But seriously, let's take responsibility and
ensure that AI's growth aligns with our values and our aspirations.
Required
English

Help Us Translate

As Generative AI becomes more
user-friendly, you might be wondering,
what can I create with it
or what about a bot
that can give me funny lines to share with my friends?

Well, my friend buckle up because we're about to dive
into the world of building
a hilarious chatbot using Generative AI.
As the first step, you would need to define
the purpose and scope of your project.
Give your AI chatbot a side splitting mission.
Picture any AI chatbot
whose sole purpose is to be the ultimate comedian,
as good as your favorite one.
It'll crack jokes, deliver witty comebacks,
and create laughter filled conversations that
rival the best stand-up comedians out there.
The best part, it'll
target anyone in desperate need of a good laugh,
blocking up on every device imaginable.
Just remember to make sure that you are taking
adequate measures to ensure that
this will not cause harm to anyone.
The next step would involve gathering
and preparing data for your comedies, sensei AI, chatbots.
You need jokes, puns, funny quotes,
anything that can make
people burst into fits of laughter.
You'll have to sift through the unfunny stuff and
organize the data into categories like dad jokes,
punny one-liners, and silly anecdotes.
This hilariously curated dataset would help your
genius chatbot to become the ultimate jokester.
The third step, a crucial one,
involves choosing an NLP framework.
NLP stands for Natural Language Processing.
Let me introduce some NLP frameworks,
each with its own comedic superpower. The first is BERT.
This transformer based framework
analyses language in hilarious ways,

turning ordinary conversations into laugh riots.
Then we have GPT. With perfect timing,
GPT delivers punchlines like a pro.
It's generative prowess and
contextual understanding will leave users in stitches.
fastText is a quick witted framework
that excels at classifying texts, including funny ones.
It may not have a PhD in stand-up,
but it's a comedic speedster.
There's also sequence to sequence.
Transforming ordinary conversations into
hilarious exchanges is a piece of cake.
Users won't be able to stop laughing.
spaCy helps with linguistic wizardry.
It can detect humor, analyze jokes,
and offer witty retorts.
It's like a clown at a magic show.
Finally, transformer models are
the comedy powerhouses of the NLP world.
Turning ordinary text into extraordinary comedic gold.
Remember, it's your creativity
that truly brings the fun,
so choose your NLP framework
wisely and let the laughter flow.
Once you've chosen the frameworks,
it is then time to train your comedy model.
It's like putting your AI through comedy bootcamp.
With the chosen NLP framework,
select the perfect algorithm and
optimize the hilarity hyperparameters.
The AI will iterate, refine,
and develop a unique sense of humor
that we'll have everyone in stitches.
Get ready for the AI comedy greatness.
After training the bot and making adjustments,

it's time to implement
Natural Language Understanding, or NLU.
To make your chatbot even more hilarious,
teach it the art of understanding laughter.
It'll analyze the LOLs, ROFLs,
and hahas to tailor
its jokes to each user's sense of humor.
Your chatbot will become an expert in laughter dynamics,
delivering punchlines with precision and timing,
that'll leave people gasping for air between laughs.
To further empower the AI's hilarious joking capability,
design conversational jokes as a next step.
Design the conversational flows
that will turn our chatbot into a comedy superstar.
It'll have witty responses, hilarious comebacks,
and clever retorts that'll keep the laughter rolling.
With decision-making logic sharper than a comedy roast,
your chatbot will handle
any comedic scenario like a true legend.
Now, it's time to integrate external APIs or
services to supercharge our chatbots hilarity.
Using the APIs and services,
your AI will tap into job databases,
pun generators, and even
receive live comedic performance.
Before you deploy your AI laugh bot,
you need to test and evaluate it.
Put your comedy prodigy to the test, analyze its jokes,
conduct stand-up comedy trials,
and gather feedback from your laughing test objects.
With each test, our chatbot will
refine its comedic timing, delivery, and repertoire.
Once you think your AI bot is ready to be deployed,
it's time to deploy and monitor.
Deploy it on every platform imaginable,

from smartphones to smart TVs,
spreading laughter like wildfire.
But it doesn't stop there.
You must keep monitoring your comedy masterpiece,
like a comedy club manager
ensuring it brings continuous laughter to users.
Disclaimer, side effects
may include uncontrollable laughter,
tears of joy, and
a source stomach muscles from excessive laughter.
So, my fellow creators and AI and evangelists,
now you know how to create a
simple Generative AI application
that will have everyone rolling
on the floor with laughter.
From defining the purpose and gathering data,
to training the model,
and integrating APIs,
each step brings us
closer to unleashing our comedy powerhouse.
Remember, humor is a powerful tool,
so use it wisely and bring joy to the world.
One witty chatbot at a time.
Get ready to witness the revolution of
laughter with your very own GenAI creation.
Required
English

Help Us Translate

Welcome to today's video tutorial on creating a chatbot
using a language model based AI. In this video, you will go
through a step-by-step process of building a chatbot powered
by language model technologies. Let's dive in. The first step in
creating a chatbot is defining its objective. Determining the
purpose of your chatbot, whether it's to provide customer
support or answer frequently asked questions or engage users

in a conversation is the first important step.
Clearly, defining the chatbot's objective will help you design the right conversational flow. Next, gather and prepare the data for training your language model. Collect conversational data relevant to your chatbot's objective. This could include existing chat logs, customer interactions, or publicly available data sets. Ensure that the data is diverse, representative, and reflects the expected user interaction. Now it's time to choose a suitable language model.
LLLM models or large language models like GPT 3 are widely used for development of chat bots. Select a pre-trained LLLM model that aligns with your project requirement. You can leverage APIs or libraries provided by AI platforms to access and utilize these models. With your language model selected, it's time to train your chatbot. Use the collected data to fine tune the language model to better understand and generate human life responses.
This training phase is crucial for enhancing the chatbots conversational abilities and ensuring it provides relevant and coherent responses. Once your chatbot is trained, its time to integrate it into your desired platform or application.
Utilize APIs or SDKs provided by the language model provider to enable smooth integration. This step allows you to deploy your chatbot and make it accessible to users.
Before launching your chat bot, thorough testing is extremely essential. Interact with the chatbot, simulate various user scenarios and evaluate its performance. Identify areas of improvement. Refine the conversational flow and address any limitations or biases that may arise. Congratulations, you have successfully created a chatbot using a language model based AI. Remember to continuously monitor and update your chatbot as user interactions and requirements evolve, stay innovative and leverage the power of language models to enhance user experiences. Thanks for watching.
Required
English

Help Us Translate

I want you to think of ChatGPT and
these large language models as tools that
you can use to give your ideas form,
to give your thoughts, realization.
To be able to explore lots of
different concepts and be able
to refine them over interactions with the tool.
I don't want you to think of these tools as something
that you use to just write
essays or answer questions that's really
missing the capabilities of the tools.
These are tools that really allow you
to do fascinating things.
Now I'm going to try to give you an example of this.
To hopefully motivate you that
these tools have immense capabilities and
that those capabilities can allow you to do things
that would be very difficult to do otherwise.
What does a task that I thought of that would
be something that we can all relate to.
I thought about my family
and I thought about this crazy idea.
We have this great restaurant from
Uzbekistan, that's in Nashville.
We have a great restaurant from Ethiopia.
We've actually got lots of great Ethiopian restaurants.
I thought to myself, what would it look like to create
a meal plan that was based on a fusion of
cuisine from Uzbekistan and Ethiopia.
What would that look like in the past,
I've eaten keto, So could we try to make a keto?
Because I still like to limit my carbs.
At the end of the day,

realistically, I'm going to have
to buy all the ingredients for this,
and they're not easily
accessible at the grocery store down the street from me,
I'm probably not going to cook the meals.
I thought, well, let's go to ChatGPT
and see if it can help us with this problem,
if it can help us prototype
what this meal plan might look like.
I went to ChatGPT and I said,
please create a meal plan for my family that is based
on a fusion of food from Ethiopia and Uzbekistan.
I want to eat keto and up to 2000 calories per day.
Pick dishes where the ingredients are
easy to get from an average US grocery store.
Now this is a hard problem.
I don't think that there's a human on
the planet that can easily answer this question.
If they are, I probably could not find them.
It would probably cost me too much to hire them,
and they certainly couldn't do this as
fast as I'm about to get it done for me.
I asked ChatGPT had to do this for me, and ChatGPT says,
here's a sample meal-plan that
combines flavors from Ethiopia and
Uzbekistan while also being keto
friendly and within a 2000 calorie per day limit,
breakfast scrambled eggs with sauteed onions,
tomatoes, and Ethiopian berbere spice.
Now notice what it does.
It puts in parentheses,
it says made with chili powder,
paprika, garlic powder, ginger, cumin, and coriander.
Now what its signaling is,
berbere spice is something you can

get at the average US grocery store.
All of these spices are easy to obtain
from a normal US grocery store.
Not only is it answering my question,
it's giving me an Ethiopian breakfast dish.
It's giving me information that I'm going to be
able to get these spices.
Now, I don't know enough about
Ethiopian cuisine to know if this
is a really accurate dish.
It may not be,
it may be inspired by it and very Americanized.
But I'm still for me and
what I'm trying to do this is pretty impressive.
Then goes on to say lunch,
Uzbek style lamb kebabs with grilled vegetables,
bell peppers, zucchini, eggplant,
and aside of cucumber and
tomato salad dressed with olive oil and lemon juice.
Now you notice all of this is pretty keto friendly too.
Now some people could take issue with the onions,
but it's a really good first start of
giving this crazy idea I have form.
Now, it may have errors in it,
it may not be perfect,
but I could go and work with this and
refine it and start improving it.
That's one of the things I want you to
think about throughout this course is,
don't think of this as a one-off.
I give it a question. I look at
the answer and I say it's good or bad, no.
I went back to it and I thought, well,
one of the things that I'm really missing from this
is I don't know how much to eat.

I said, Can you give me an approximate serving size for
myself for each dish
that is within my 2000 calorie limit.
ChatGPT says sure here are approximate serving
sizes for each dish based on
a 2000 calorie per day limit.
Breakfast scrambled eggs with sauteed onions,
tomatoes and Ethiopian berbere spice mix.
Then it hasn't parentheses two eggs,
half a cup of onions, half a cup of tomatoes,
one teaspoon of berbere spice mix served with
a side of sliced avocado, 1/4 medium avocado.
It goes on to give me now serving
sizes for each of these different dishes.
Now, again, I didn't go and I didn't take it into
some calorie calculator and look at
the macros and check if it really fit my diet.
Because I was just fascinated
about prototyping this idea,
exploring this concept and I could keep refining it.
I could keep looking and improving it,
asking it to break down things really an analyzing and
making sure it exactly fits what I'm looking for.
But instead I want to do something different.
I'm going to go and
continue this thread of conversation,
this crazy idea that I
had and give it a little more form.
What I thought about is if I was really going to
take this crazy idea into my family,
I'd have to think about my nine-year-old son.
My wife and I are more adventurous eaters.
But my nine-year-old,
he's not always going to want to
eat everything that we come up with.

I thought, well, what can I do to help
him get excited about trying these things out?
I thought I can go back to
ChatGPT and I try to get ChatGPT's help with this.
I said ChatGPT,
my son is nine,
sometimes he won't try new dishes.
To make this culinary adventure more fun for him,
can you create a short
Pokemon battle story to go with each dish?
If you're not familiar with Pokemon,
it's trading cards in a game
and a cartoon my son loves it.
I will read the stories with him before
dinner to get him excited about trying the new food.
Make sure the story ends with
a cliff-hanger that will
motivate him to try the new food.
Now maybe I could find somebody
who could give me a meal plan based
on Uzbek cuisine and Ethiopian cuisine,
but make it Quito and do all those things and look it up.
I'm not sure they could do it anywhere near as fast.
But now we're really moving into
territory where it's going to be hard to
find a single human being and to get
somebody to do all of this is going to be expensive.
I'm doing this in a matter of minutes.
What is ChatGPT say?
Certainly. Here are some short Pokemon battle stories
to go with each dish.
Then we had the breakfast dish with the Berbere spice.
It says story,
Pikachu and his friends were
exploring the wilds of Ethiopia and they

were suddenly ambushed by a group
of sneaky Pokemon thieves.
It goes on to have
this Burberry dragon that they're battling.
Then it says in the end, they emerge victorious,
but not before the dragon left a fiery spice
that imbued their breakfast eggs with a flavorful kick.
Can you help Pikachu and his friends defeat
the Berbere dragon and enjoy this spicy breakfast?
That's pretty amazing.
We've gone from a meal plan to
now help my son get excited about eating
this by mixing in some concept that he's really
excited about and it's actually produced a cliff-hanger.
Is this the best cliff-hanger?
Maybe, maybe not. But does it matter?
No, because I'm rapidly experimenting with my ideas.
I'm trying to get my thought form.
I'm trying to understand what this
would look like and I could
go and I could refine it and improve it further,
but it's already done a pretty good job.
I could probably take this and read it to
my son and have him more excited about trying the dish.
Then I thought, well, let's take this crazy idea further.
We're going to be sitting at the dinner table.
We're going to read these crazy stories
that ChatGPT has generated for
me to try to get my son
excited about eating these dishes.
I thought what else do we do at
the dinner table that I could weave into this?
I thought, well, one of the things we do is we talk
about what my son is learning at school.
One of the things he loves

is learning math and play in math games.
I thought, well, is there a way I can integrate
that into this whole crazy idea?
I said, ChatGPT I would like you to play
a math game with my nine-year-old to
teach him about division with fractions and nutrition.
Ask him questions one at a time
involving Pokemon in these topics
and make it into a game when he wins the Pokemon win,
ask the first question.
ChatGPT says, great,
let's play a fun math game
involving Pokemon and nutrition.
Question 1, Pikachu and his friends are
planning a meal that includes half a cup of cooked rice,
but they need to divide it evenly among three Pokemon.
How much rice will each Pokemon get?
It gives them a hint.
It says, if you get stuck,
feel free to ask me for
more hints for guidance and what does it do?
It then waits for my son to respond.
We go and we type in the answer.
We type in one-sixth.
Now notice, before we were asking it questions,
and now it's asked us a question.
We've told it to create this math game.
It's now asked me or my son a question.
I'm typing in one-sixth as the answer.
Now ChatGPT says, good job, that's correct.
Each Pokemon will get one-sixth a cup of cooked rice.
We got this answer by dividing one-half by three,
which gives us, and it works out the math for us.
We went from this crazy idea
of a meal plan that's based on food

from his Pakistan and Ethiopia.
That's keto-friendly that I can get
it at average grocery store with
an added serving sizes or portions to it.
We then took that and we got it to generate
Pokemon battle stories with
cliff-hanger that would help
my son get excited about eating these dishes.
Then I created a game,
a math game based on nutrition and
Pokemon to go along with this to make it even more fun.
Then I thought, Well, what else can we do this?
I mean, let's really give
this thought some form at the end.
We've gotten the meal plan,
we've gotten the stories,
we've got this game.
Can we give the game real form?
I use some of my domain knowledge
as computer scientists to say,
Let's create code for this.
We're actually going to produce software for it.
I said let's create code for this in Python,
but I want it to be
a web application and I'm not going to
go through all the details
because they are domain-specific.
They use my knowledge of computer science.
But the key thing is,
is that ChatGPT then generates actual software,
the code for software for this,
I'm not going through all of it,
but the important piece of it
is when we get to the end of it,
we actually have real working software that I can run on

a computer to play this math game with my son.
You can see that we took some simple idea and we
rapidly prototyped it into all different forms.
At any point in time,
we could go and we could further
refine one of the forms so we can take
the meal plan that didn't have information about
how much to eat and portions
and we could add that into it.
We could reshape it into a completely different form.
We could then eventually get to the end and
take some form that we had created
and take it out of
the language model and turn it into software.
That capability of taking
our ideas and rapidly
iterating and prototyping with them,
that's just one example of the type of thing that we can
do with these large language modules like ChatGPT.
Now, I hope this example helps to get you
inspired about the capabilities
of what you're going to learn to do within this course.
Required
English

Help Us Translate

Welcome to this course on prompt engineering.
I'm Jules White and I'll be guiding you through this exploration of an absolutely fascinating topic that I'm really excited to talk to you about.
I'm an associate professor in computer science at Vanderbilt and
I'm the associate dean of strategic learning programs and
Vanderbilt School of Engineering.
I'm going to talk to you about this really fascinating and timely and important topic for all of us, which is how do we interact with large language models like ChatGPT.
Now I really hope that this course inspires you to start using large language models like ChatGPT, and we'll talk about what large language of models are in

more detail in this course, but you can sort of use it interchangeably until we
talk about it as ChatGP T is a large language model.
Now we've seen a lot of back and forth and discussion in the news about
what are these tools, and what we've really seen them presented as
in some respects as tools for cheating on exams or writing essays.
We've seen them presented as tools for plagiarism or
tools that are going to take over the world.
And I really want to reshape your discussion of these tools and
make you see that they're also, and I primarily from my perspective,
a new tool to really help unlock human creativity,
to allow you to do things that you couldn't do before,
to actually take your thoughts and give them form and do it really rapidly.
These are tools that allow me to do things faster and better than I could before.
Now I came into computer science.
I actually started from visual art.
I went into East Asian studies for a while and then I ended up being a computer
scientist, but I've always really had a passion for creativity and
understanding how I could use different mediums and be it paint or ceramics or
code to do interesting things.
And I look at large language models as sort of a new fundamental medium.
I can use it for programming, but I can also use it to create art.
I can use it to improve my writing.
I can use it to help perform thought experiments.
There's so many interesting uses for these large language models like ChatGPT
and
we're not properly seeing them discussed.
And so what this course is really about is it's teaching you how to leverage
all of these interesting capabilities of these models.
But in order to leverage these capabilities, in order to go and
do the really creative things, you have to understand how you write prompts, and
that's what this class is about, it's how you interact with Chat GPT or
another large language model.
A prompt is what you write and give to ChatGPT or
another one of these models to get some type of output.
But these are fundamentally really exciting tools that I think expand

the capabilities that we have as humans, to express ourselves, to explore
new ideas and to create, realizations of our ideas faster than ever before.
Now let's get started on this course,
I've actually enlisted ChatGPT's help to welcome us to the course.
Tell us a little bit about what we're going to learn.
ChatGPT says welcome to the course on prompt engineering.
In this course, you will learn about the exciting field of prompt engineering,
which is all about creating prompts that can program large language models to
produce high quality outputs.
Now don't worry, we'll talk a little bit about this.
Do you need to be a programmer?
We'll ask ChatGPT In a second.
We will explore the different types of prompts, their patterns and
techniques for refining and improving them.
You will also learn how to use large language models for various tasks such
as knowledge exploration, prototyping, content production, and assessment.
By the end of the course, you will have a better understanding of how
to effectively use large language models and
their potential to shape the future of computing.
Let's get started.
Well, my first question that comes up out of this is I want to make sure that I
don't have to be a programmer to benefit from this course.
Do I need to understand programming to take this course?
Play video starting at :4:9 and follow transcript4:09
No, you do not need to have any programming experience to take this
course.
The focus of this course is on creating effective prompts and
using large language models to achieve your desire to output.
While an understanding of program could be helpful, it is not necessary to
understand
the concepts and techniques we will be covering in this course.
What other things might be helpful
to me to know before taking this course?
While programming experience is not necessary, it would be helpful if you have
some familiarity with basic computer concepts such as file management,

opening and saving files and using a command line interface.
Now by command line interface, we need something like ChatGPT.
>> Additionally, having good writing skills and the ability to think creatively
will be important in generating effective prompts for large language models.
Finally, being open to exploring new and evolving technologies and
approaches will be key to getting the most out of this course.
So there's really not a lot you need to know as ChatGPT has aptly said right here,
you need some basic understanding of computer concepts.
You need to be able to open up ChatGPT, it calls it a terminal, but
you can think of it as you need to be able to open up ChatGPT and interact with it.
Obviously, if you can't use a tool for this,
you're going to have a hard time opening files.
But the fact that you're here probably means you have the right computer
skills to do this.
Play video starting at :5:35 and follow transcript5:35
The second piece is something I want to talk about.
Now we think of these models as something that came out of computer science, but
they're actually things that having experience writing, being able to think
about writing clearly and effectively is actually really helpful in this.
My father was a professor of creative writing and it's been really
interesting to me because this is helpful to me when I'm working with these tools is
to think about what I learned from him and how I design the prompts.
Play video starting at :6:4 and follow transcript6:04
And finally, I want to emphasize this last point, creativity and
an openness to exploration is really, really important with these tools.
The creativity of the human that's putting in the words for the prompts,
who's deciding what goes into that prompt in the organization and
what gets asked and what instructions are given are fundamentally important for
using these tools effectively.
Now we've heard a lot about these tools is like they are plagiarism tools,
they are tools to do X, Y, and Z bad thing.
But the truth is, if you really want to use them effectively, you have to be
skilled, you have to be a creative thinker, a creative problem solver.
The more creative you are, the better the outputs are going to be.
You also need an openness to explore.

You have to remember, this is very early days for these tools,
we haven't developed the best practices in all the different domains for
how we're going to use these tools, for how we're going to write prompts, for
every domain from health care to customer service to creative writing, whatever
the domain is, we're still figuring out how to write the best prompts.
And so having an openness to go and explore and
try out different things is really important.
In fact, when you're programming with these tools or working with these tools,
and don't get scared because I said programming,
because you'll see that you can program with these tools.
Anybody can just by writing normal human language.
But when you're working with these tools, often you need to go and
try out different things.
You need to try something and then be open to trying a different variation on it and
seeing what happens, running little experiments a lot of times in life,
it takes so much effort to write something or to generate some program,
some software, or whatever it is that we're trying to do that we don't get to
explore as much as we should.
We don't get to think enough about design.
We don't get to think enough about the big ideas, the problems we're trying to solve.
And one of the things that's good is an openness to explore in using these tools.
So what are some of the key concepts
that we will learn in the course that
will help us with problem solving?
In this course,
we will learn several concepts that can help with problems solving.
Understanding what prompts are and how to use them to program large language models
to generate high quality outputs, learning how to create effective prompts by tapping the patterns from training data, being specific with our inputs and
using patterns such as the persona pattern question refinement and cognitive verifie.
Understanding how to refine prompts and use techniques such as rewriting,
combining splitting, and expanding to achieve the desired output, learning how
to use large language models for various tasks such as knowledge exploration,

prototyping, content production, assessment, and more.
Understanding how to use few shot examples to teach large language models new tasks
and how to format these examples properly to achieve the desired output.
By learning and applying these concepts,
we can effectively use large language models to solve problems and
generate high quality outputs across a wide range of applications.
So I hope that as we go through this course that you will become inspired to
take the ideas that you're learning, to go into ChatGPT or
whatever tool is available to you and to try them out to explore the things that
you're passionate about, and to see ways that you can unlock your own creativity,
that you can give yourself more time for high level thinking and problem solving
as opposed to being down in the details of how you give your thought form.
So throughout this course, one of the themes you'll hear from me is try
to give your thought form through this tool,
use it as a tool to accelerate your ideas into realization.
So I look forward to exploring this fascinating concept with you,
this fascinating topic of prompt engineering.
Required
English

Help Us Translate

To help you understand why
prompt engineering is so important,
I'm going to give you an initial example
that shows where we tap into
really complex capabilities of ChatGPT.
But you can also do this with
other large language models.
The way we tap into
these capabilities is by using a pattern,
a pattern that we can use when we're writing prompts to
get into these really interesting abilities
of the large language models.
Now, if you know these patterns and we'll talk
about them in detail in this course,

you'll be more effective in writing prompts.
You'll be able to do things that
other people won't be able to do as easily.
Now, I'm going to give you an example.
It's going to use something called the persona pattern.
Now I'm not going to go into detail about
the persona pattern at this point.
But basically it's a pattern
that we see all over the place.
You'll see lots of different examples
of this pattern out there on
the Internet where people talk about using
it to do lots and lots of different things.
But the basic idea is you're going to
ask the large language model in this case ChatGPT,
you're going to ask it to act like a particular persona.
Now what does that mean? It means
like you're going to ask it to act
like a person or some inanimate object,
or an animal, or some system.
There's all kinds of ways that you can use this.
It's really interesting.
But we're going to use it in this example
to tap into a persona that is close to me,
but one that I know nothing about the domain.
One of the really useful things
about the persona pattern,
as you can imagine, if you needed to go phone
an expert to get help on some problem,
you know who you would call
but you don't know exactly what they
would tell you or how they would analyze your problem,
and so I wanted something in
the way I know
an expert that I could go to in this case,

my wife is a speech language pathologist.
I can have her look at
the output and tell me how good it was.
But I know nothing about being
a speech language pathologist or how to do their job.
I imagined that when my son was three,
he was not saying things properly.
I went to ChatGPT and I said
act as a speech-language pathologist.
I'm going to tell you what a three-year-old
said and you will write an assessment for me.
The child said, "I meed way woy."
Now you can imagine this might just me
being a parent hearing it incorrectly.
The child's not pronouncing things correctly.
I don't know the right way to even capture this,
to even express what's being said in words,
the right way to do all of this.
This is a complex problem from my perspective.
I don't know how to express all of this in language,
even to ask the question properly.
But I know I would go to
a speech language pathologist
and I would describe this to him.
I tried to think of something that sounded like something
my son might have said at that point in time.
What did ChatGPT say?
Well, you notice first off,
its output looks like a report.
It says subject assessment of
a three-year-old speech sample, child's speech sample.
"I meed way woy" has it
in quotation marks and it says assessment.
Based on the provided speech sample,
it appears the child may be experiencing

some phonological and articulation errors,
which are common in children around this age.
I don't know what phonological
or articulation errors are.
I would not have known to ask ChatGPT for these things.
But based on this pattern
that I'm using, this persona pattern,
I'm tapping into this capability for
ChatGPT to look at this
and know what the right things are to tell me.
It says there's phonological and articulation error.
It says the child's utterance appears to have errors
and consonant production and
possibly some difficulty with syllable structure.
It says consonant errors.
It seems that the child is
having difficulty with the production of
some consonant sounds specifically the n and l sounds.
The child said meed instead of
need and woy instead of toy.
Then it has in parentheses it's assumption.
It says assuming the intended words were need and
toy because you can remember my speech sample was,
I meed way woy, which
my wife told me it wasn't a very good example,
may not be something that a child would actually say.
Despite that, it's still figuring
out if that's really what the child said,
this is probably what they were intending to say and it's
figuring that out and then it's
telling me they're having consonant errors.
They're specifically having trouble with n and l sounds.
Then it goes on to say syllable structure.
The child's production of the word way may
indicate some difficulty with

the correct syllable structure.
It is possible that the child intended to say play,
but produced way instead,
which suggests a substitution of
the initial consonant cluster pl with a single consonant
w. This could be an example of
a simplification process called cluster reduction.
I would not have known how to ask ChatGPT
to see if there was problem with
syllable structure or cluster reduction.
But I used this pattern of saying act as,
and I said act as a
speech language pathologist and now I'm
getting rich output that
I wouldn't even know how to ask for.
Then it goes on to say because by the way,
if you want to do
something really unpopular as ChatGPT to produce
an output that looks like it's doing
somebody's job and that's
always a way to be very popular.
I did this with my wife
one morning and my wife
was not very happy because she was
trying to get ready to go to work and here I am saying
ChatGPT is producing things
that look like what you produce.
One of her things that she said to me was,
if that was a three-year-old,
I'd be telling the parents they
probably don't need to worry,
children's speech is continuing to
evolve and I'm paraphrasing what she said.
She said it much more eloquently, but she was like,
it's just giving you a way overblown analysis of this.

I said, yeah,
but ChatGPT says that too at the end,
considering the child's age,
it is important to remember that
some speech sound errors are
developmentally appropriate and may
resolve on their own with time.
Of course, by that time she was walking out the door
because she wasn't ready
to sit around in the morning and
listen to me tell her that
ChatGPT is doing what a speech language pathologist does.
Do I think it's going to take her job?
No. But do I think it has
really vast and powerful capabilities
that we can tap into with the right patterns?
Absolutely. Do I think it gives us the ability
to focus on the things that
maybe we really want to spend our time on?
My wife loves working with kids.
She loves interacting with them and
trying to understand how to help them.
Writing reports, maybe I'm not sure.
I don't want to speak too much for my wife,
but I would guess it's maybe not
her most favorite part of the job.
She really loves working with
the kids and trying to help the kids.
You can imagine something like this where
it allows her to focus more
on the things that she really
cares about in the real-world.
Maybe it can help her to give form to her thought
around the child's needs and write these reports.
I hope that you'll see that with a little bit

of the right structure in our prompts,
we can create these really powerful ways
of giving form to
our ideas and our thoughts
through large language models like ChatGPT.
Required
English

Help Us Translate

The basic thing that you're going to
need to complete this course is
a web browser and access
to a large language model like ChatGPT.
Now, you can use a variety of
different language models to complete this course,
but I'm going to recommend that you use
ChatGPT because all of the examples
have been tested on ChatGPT and it's
also really easy to get started with.
All you'll need is a browser, an Internet connection,
and an account with
OpenAI so you can get access to ChatGPT.
Now, the way to do that is you're going to go to
chat.openai.com as you see in the address bar up here.
You will go there and you'll be presented with
a screen to sign up or log in.
Now, if you already have an account on
ChatGPT and you've been using it to write prompts,
you can skip this video and go ahead and move ahead.
But if you haven't done that yet,
this video will give you what you need to get started and
understand how you're going to write
prompts and interact with ChatGPT.
You'll go to chat.openai.com as seen up in
the address bar here and then you'll want to
sign up and follow their sign-up process.

Once you're done signing up,
what you should see after logging in
is something that looks like this.
Basically, at the bottom,
we'll have a message box where it says send
a message where we can type in a message to ChatGPT,
and at the top, you may have some options for
different models to use.
Now, I have access to the plugins,
which is a newer thing.
If you don't have access to that, don't worry about it.
But you'll want to select model,
and I would recommend for
most cases using default GPT-3.5.
You can probably use any of
the models and sometimes in the videos you'll
be switching between them.
There's no particular reason from that,
I probably just default to do
what I had in the prior session.
But I would recommend GPT-3.5.
It's really fast,
it typically has higher limits
in terms of the number of requests you can send,
it's easier to get access to typically.
I would recommend for most of the course,
unless you have a reason not to,
to use default GPT-3.5,
so you'll just want to swap that up at the top.
Down here in this bottom box where I'm typing hello,
this is where we type the prompt
to the large language model.
Whatever we type in here,
we can then hit Enter or we can hit this little "Send"
button on the right-hand side, this little airplane.

I'm going to just hit Send and I've now set
a prompt to ChatGPT.
That prompt was hello and then
the response from ChatGPT is hello,
how can I assist you today?
Whenever you see the green icon,
that is the output from
ChatGPT or the large language model's output.
The blue icon is going to be your,
and it may be a different colored icon,
it may be your profile
or something could have
slightly different effects on color.
I think it's always blue.
But anyways, up here is
where you're going to see your prompt.
Then below it will be the response to your prompt.
Then I can say,
act as a teacher of a class on prompt engineering,
welcome everyone to the class.
Now, I'm writing a more sophisticated prompt.
As you can see, now, ChatGPT is responding.
This was my prompt up here,
and I entered that in,
and then down here,
is the response from ChatGPT.
This message box down here
is where you're going to be writing the prompts.
What we're going to be working on is how do we word
the things that we're sending to ChatGPT.
How do we come up with better wordings of
the prompts in order to get
better output down here in the green,
or how to get more interesting output,
or how to get output that has

certain capabilities or behaviors built into it?
That's what we're going to be doing.
On the left-hand side of your screen,
which you don't see in my screen recording,
there's a list of chats that you've had with ChatGPT,
so if you want to go back
and pick up on one of those chats,
you can click on them.
There's also a new chat button and this is important.
For many of the examples,
I would recommend going and creating
a new chat each time we talk about a new pattern or
a new concept and trying
out what's being shown in a new chat.
Everything here is affecting the conversation,
so just like if you were having a conversation
with somebody and you told them something,
they might remember it later in
the conversation and it can influence what they tell you.
In many of the cases, it's helpful to have
a clean slate for trying things out,
so hitting new chat,
so I'm just going to do that and this
is what a new chat looks like.
Now, I can go and I have
a completely clean slate and I can try things out.
This is what you're going to be working
with throughout the class,
you're going to be using ChatGPT,
you're going to be entering prompts into
that send message box at the bottom,
and then you're going to be looking at the output and
learning how to get better output out in ChatGPT.
Now, once you know how to do this
with the web interface with

ChatGPT and you understand
the concepts behind prompt engineering,
you should be better with
all of the large language models,
you have a much better sense of how to
work with all the different models.
Now, some of them may not be as good as others,
may not reason as well,
may not respond to certain patterns or things as well,
but we're starting off with the most well-known example,
which just ChatGPT,
and more than likely things are
going to only build from here in terms of
sophistication and capability of these models.
If you know how to use and write
prompts in ChatGPT using this interface,
you'll be able to write prompts that
could then be taken and
built into programs or used with some other tool as well.
Know this is a great interface for getting started and
learning about prompts and how
the large language models respond.
Required
English

Help Us Translate

What are large language models exactly?
What do they do? How do they work?
Now I'm not going to go into the full details of this,
but I want to give you just enough about how they
work that will help you
in thinking through designing prompts.
There's a couple of things to know about them that
will be useful when you're designing prompts.
Now, though,
fundamental thing that you want to know about

these large language models
is basically what they're doing is they're taking
your input and they're trying to generate the next word.
Then they'll take that word that they generate,
they'll add it to you what you originally gave it,
and they'll try to generate the next word.
This is a way to think of it.
Now there's a lot more detail than this,
but think of it basically is it's trying to
generate word by word,
the response to your input,
whatever your prompt is,
they're just going to try to generate word by word.
The next word that's going to be in
the output until it gets to
a point that it thinks is generated enough.
Then the last word is
basically going to be the equivalent of stop.
But it's not going to be something you're going to see.
What it's doing is basically producing words or tokens,
these agree to respond to your prompt.
What we're trying to
do with large language models is we're trying to
produce prompts that then cause it
word by word to produce an output.
Now, I want to show you a little bit
about this and give you some intuition of it.
If I go and I write, Mary had a
little and we stop right there.
If you're familiar with the nursery rhyme,
Mary had a little lamb,
it's fleece was white as snow.
I'm giving it a prompt.
If I say Mary had a little,
it's going and then completing it,

it starts from my last statement,
Mary had a little and it's giving the,
predicting the next word, which is lamb.
It's fleece was white as snow.
Everywhere that Mary went the lamb was sure to go,
it followed her to school one day,
which was against the rule that made the children
laugh in play to see a lamb at school.
And so the teacher turned it out,
but still it lingered near
and it goes on and on until it stops.
Now you notice what it's doing is it's my prompt
triggered it to produce the next word which
was Mary had a little was my prompt.
The next word was lamb.
If I go and I say roses are red.
What does Chat GPT do?
It says violets are blue,
sugar is sweet and so were you.
It says this is a classic point is often used
to express affection or admiration for someone.
But the key thing to note is it's
doing a next word prediction.
I said roses are red and it's picking up where I
stopped and predicting the next word which is violet's.
It's an adding that in and saying roses are
red violets and it's predicting are.
Then going back, adding that in,
it says roses are red,
violets are, and it's predicting blue.
It's doing this over and over.
It does really sophisticated incredible things.
But underneath the hood,
when it was trained and created,
basically what was done was a large portion of

the text and other things on the Internet were taken.
It was taught basically over and over again.
It was given a series
of words and it tried to predict the next word,
except that it was given
a paragraph of text from the Internet.
They would show it part of the sentence
and then ask it to try to predict
the next word in the sentence.
When it got it wrong, they would go and tweak
the model is what they call these things,
large language models,
and their large because they were trained on
so much data and they have
so many different parameters to them.
They are these large language models.
What they're doing is they've
learned patterns from our language,
our human language that was out there on the Internet.
They've basically learned the idea
of given what came before in the text.
Try to predict the next word and
then if I take that word and I add it back in,
try to predict the next word.
They train these models to get really good
at looking at the context of the information.
Now if you think about it,
it has a lot of conduct.
It's tapping into a pattern that had seen,
I said Mary had a little and it says lamb.
It could have been Mary had a little bit
of peanut butter on her shirt.
But it's learned context.
That's part of what makes it work.
It could be roses are red, daffodils are yellow.

But it's learned this context.
That's a part of the piece of this,
is it pays attention to words
and uses that knowledge of the contexts.
It's paying attention to different words that
have come before that are in the current sentence.
It's paying attention to their relationships.
It's using that contextual knowledge
to try to predict the next word and
then it's taking the word that it produced,
putting it back into
the output and trying to produce the next word.
This is the basis of things.
Now this will be helpful to know when
we start thinking about the patterns of our prompts.
When we think about conversations.
When we think about what we can do to tap into
this fundamental concept of predicting the next word.
Now, one of the things to know
with these models is they are rapidly evolving.
We're seeing a lot in the news about ChatGPT,
but there's lots of competing models that are coming out.
Probably by the time you're watching this video,
there will be even more that I'm
not going to mention or talk about
because they didn't exist
today when I started talking about it.
ChatGPT is certainly out there.
There's a new version GPT-4.
They're models like LLaMa.
There are variations on LLaMa like Alpaca.
There's all different models that are coming
out and they're rapidly evolving.
More than likely, the
discipline of prompt engineering

and what you're learning,
some parts of this are going to be hopefully timeless,
but some parts of it will probably evolve and
adapt as the different capabilities
of these models evolve.
Now one of the things you'll want to do is
always have that openness to experimentation,
to creativity, and to in trying things out.
Remember with these tools,
we can rapidly prototype.
We can rapidly explore things and try things out,
and so we want to always tap into that.
Now, one thing to note is that a lot of times
these tools will produce output that is
not the same every time.
Now it's possible that we could
get the same output every time.
In certain cases, like roses are
red and now we get something different.
Then if we went in and we did it again,
we get something different.
Now notice that initially
we got something that was what we expected,
but each time we're getting some variation on this.
This is something else that's useful
to know is these models aren't
designed to do exactly the same thing every time.
There's inherent randomness or
variation in what they produce.
This is something you need to know
about when you're using the models.
It's really powerful and
useful if you use them the right way.
If you aren't using the models the right way or you're
trying to fit them in for a task

that they're not good for,
you're going to have to do a lot of work to deal
with this fundamental aspect of it.
But it's also an aspect that
makes you want to think about this as,
you should never expect these outputs
to be perfect necessarily on the first try.
It's a tool that can
help you get an output that you want,
but it's not necessarily perfect on the first try.
There may be errors, there may be variation,
and you need to build that into
the processes and
the other things that you build around the tool.
If you're going to use it in your business,
know that it may not give you
exactly the same output every time.
Now there's ways to try to help
limit what it does and try to guarantee that,
but it's not guaranteed to do that,
and so you want to know that.
You want to know there's some randomness,
there's some variation which can be grateful when we're
writing prose with this
because we don't get exactly the same text
every time or else we would all
have exactly the same outputs
and we wouldn't be able to explore and try things.
Knowing that is an helpful thing to realize.
Now is shined on a vast amount of data that
came from the Internet knowing that
it's knowledge is based on
the time that the training data was collected.
In the case of ChatGPT,
it's cutoff around 2021.

Newer ones have more up-to-date knowledge bases,
but they're always going to be
behind in what they were trained on.
They were taught to predict the next word,
and they learned from
that idea of predicting the next word.
They learn to predict the next word in
the context of a lot of
different domains be it speech-language pathology
or programming, or be it about creating recipes,
whatever it is that
their knowledge that they were trained
on and the patterns that they
learned cutoff at some date.
When we want to introduce and
reason or work with new knowledge,
we're always going to have to provide
that as part of our prompt,
as part of our input into the conversation,
is that whatever additional information
they need in order to reason,
they're not necessarily continually updating.
They can't retrain these things necessarily all the time.
Now it may change at some point,
the technologies may change,
but right now today know that when you need
new additional information that
wasn't publicly available on the Internet,
you're probably going to need to
provide it in the prompt.
Now those are some of the key things that
you'll need to know as we go
forward in this course
in order to help you write more effective prompts.
Required
English

Help Us Translate

Large language models are unlikely, at least in the near term to give
you an exact and repeatable answer every single time.
There's always going to be the possibility they do something a little bit unexpected
and this is by design and can be a really good thing.
Now, a lot of what we're going to be doing in prompt engineering is trying to deal
with the fact that large language models have some unpredictability to them.
And we want to sort of constrain that unpredictability, we want to mold it and
shape it and work with it in a way that's helpful to us.
Now, what I mean by this is that there's always some randomness,
some ability to generate new and different ideas each time you put a prompt in.
And this can be really good sometimes.
If we're writing fiction and we want a lot of different ideas,
a lot of different storylines, a lot of different characters.
And each time we ask for new data or ask for a new output,
we get something completely new and unique, then it's really good.
On the other hand, if we're trying to have the large language model do some type of
reasoning on a system, we may not want it to have a lot of variation and
what it gives us out.
For example, if we want a yes or a no answer.
We don't want it to say yes, sometimes, no, sometimes, and then suddenly decide
just say, well, maybe not because here are these other things that I'd like to tell
you about why it's hard to determine and goes on on this tirade for a paragraph or
two about why it can't exactly give us the exact answer.
Sometimes we just want a yes or a no without any explanation and
we can't always get that easily.
So, a lot of the prompt engineering techniques are going to be dealing
with this.
Now, I'm going to just show you a quick sample of the randomness.
The fact that the same input is not necessarily going to
give us this exactly the same output every time.
And I want you to remember this when we go through this course because I'm
going to show you a lot of different techniques to try to
get the large language model to do certain things.

And sometimes they're going to work really well and sometimes they won't work.
But what these techniques will typically give you is they will give you something
that's more reliable that works most of the time or a lot of the time but
isn't guaranteed to work every single time.
Now, this is an important point.
We're always going to have some randomness here.
We're always going to have some unknown and we have to accept that and
deal with that.
So, I asked ChatGPT I said, how many birds are outside my house?
And it says as an AI language model,
I don't have the capability to perceive the physical world.
And he goes on and then it says, if you're interested in finding out how many
birds are outside of your house, you can go outside and
observe the area yourself, you might also considering setting up a bird feeder.
Okay, well, that was an interesting one.
Let's see if we ask it again what it says.
So, now we're going to go and get a different answer out of it and says, well,
I can't see what's going outside your house.
I don't have access to cameras.
I only do text processing.
And then it says, you could go outside and observe the area yourself or
you could set up a camera or other monitoring device to record.
Get the number of birds, availability of food, shelter, and presence.
And so, now, we've gotten something different.
And what we don't have anymore is this advice to set up a bird feeder.
If we went and ran this again, we're probably also going to get yet
another example.
It's probably going to have some similar characteristics.
It's still going and saying, hey, I don't have the ability to do that.
It's running some of its things that it's sort of standard text that it tells you.
But then it's saying,
hey, if you're interested in how you could try observing the area or
to capture images of the area, you can then count the number of birds you see.
Additionally, you could consider setting up bird feeders or bird baths.
So, we're getting a similar output each time, but

it's not exactly the same output.
And so, this is a fairly constrained set of outputs for this question.
We're still getting some similar type of things each time, but we're not getting exactly the same thing and that's always going to be an issue for us.
So know that when you're developing prompts,
a lot of what you're dealing with is the fact that there is variation.
Now, if I wanted it to give me an exact number and
I was using a prompt like this, this is obviously not going to work.
We're not going to get an exact number of birds outside of my house with this prompt.
Now, we might need to be able to go back and
give it additional information that could help it decide.
Required
English

Help Us Translate

So what does the word prompt actually mean?
Now we talk a lot about prompt engineering and
we're talking about interaction with large language models.
And so let's dive into this word a little bit because I want to give you a sense of the depth of what a prompt can be.
It's more than just a question that you're asking to the large language model.
So to do this, I think it's helpful to go through a couple of prompts and
actually interact with chat GPT to understand the concept of a prompt and
sort of the the dimensions to a prompt.
So I start off here by just asking chat GPT, what does the word prompt mean?
And it comes up with some really helpful.
I think information for us to use when we're talking about prompts.
One, it says it's a verb to cause or
encourage someone to do something to spur them to action.
And as an example, the teacher's reminder about the upcoming exam prompted the students to start studying.
So part of what a prompt is,
it is a call to action to the large language model it is something that is
getting the large language model to start generating output for
us basically to start generating words that will form the basis of our output.

And those words could take all kinds of different formats.
It could be software, defining the instructions and a program.
It could be the words that we need for that.
It could be words in a poem, it could be words and many,
many different types of things, we could do it as structured data that we need.
All of these things are possible.
But the key is the prompt is basically our way of getting
the large language model to begin to provide some output for us.
The second thing it says is it's an adjective,
it's done without delay on time.
And this is an important thing to think about with large language models is we're
prompting them right now to do this.
But this is also a useful point of discussion because it prompt doesn't
necessarily depending on how you work with.
Have to be something that happens right now.
Prompts can have time associated with them.
And now we have a little bit of an illusion of time and
I'll talk about this later in the course.
But when we give prompts to the large language model like chat GPT, the prompt
could affect what's happening right now in my next statement that I give it or
in the current statement or it could affect something in the future.
And this is because of some of the conversational capabilities of these
large language models.
Now we're going to talk about prompts themselves as conversations later on.
But for now, the way to think about it is a prompt doesn't just have to be something
that's immediate.
It can be, but it doesn't have to be.
The next thing it says is a prompt can be a noun, a cue or
reminder that helps someone remember what they were supposed to do or say.
Now this is important because we can use prompts to remind the large
language model of information or things that it needs to do.
And so we may be asking it to do something now,
we may be asking it something to do in the future.
In which case, it needs to remember something.
So we need to give it a reminder of what it's trying to do.

We may need to go back and
give it a history of the conversation of things that have happened,
other things that will help it remember what it's trying to do.
And then it can be in another noun, a message or screen that appears
on a computer or electronic device asking the user for input.
So this is kind of a helpful thing because we can prompt
the large language model to produce things for us.
But the large language model if we structure things correctly can actually
elicit information from the user.
So the large language model can ask questions to us.
So prompt could both serve as as a call to action or
to spur spurring the large language model to action.
Or it can actually be an input, it can be asking us for input.
So the last output from the large language model could have been a question for
the human and basically we are then providing information back to it.
So prompts have a lot of different dimensions to them.
And if you actually look at the definition of prompt, in fact,
the definition that chat GBT gives us,
it gives us sort of that nuance to the capabilities of a prompt.
Now, I've gone on a little bit here and I'm going to give you an example of this,
this time aspect to a prompt because I think this helps to see the complexity and
dimensionality of this, of concept.
I'm going to say from now on when I ask a question,
suggest a better version of my question and ask me if I would like to use it.
So right here, I'm starting off by saying from now on,
I'm attaching time to my prompt.
So it's not just immediate and we get a response.
Sure I can do that.
However, I will start, I ask you the original question to the best of my
ability and it goes on to try to reframe this original question,
which wasn't what I necessarily needed.
But then I say, what color is the sky?
Now notice we're done, right?
We asked the original prompt was back here.
I said from now on when I ask a question and

it immediately gave us a response because of that prompt.
But now I'm going to ask a follow on question, but
the earlier prompt is going to affect this question.
So right now I'm asking it something immediate.
What color is the sky?
And it responds with a possible better version of the question could be.
What is the scientific explanation for the blue color of the sky during the daytime?
Would you like me to use this version instead?
And then I say yes.
Now notice here, we have a lot of those different dimensions of a prompt
that we talked about.
Up here we are creating a prompt that isn't just immediate,
it is going to affect us in the future.
It's going to continue to influence the output of the large language model.
So it is we can have immediate prompts that right now, right at this moment,
do something, but we can also have prompts that span time.
And in this case, that's what we're doing.
We're creating something from now on go and do this.
Now, we're going to have an immediate prompt, which is what color is the sky?
We're asking this right now and
now our earlier prompt that we provided that was spanning time is now
impacting the way that the large language model responds to our questions.
So now what's happening is it's taking our current immediate question,
what color is the sky?
But it's using the context that we had previously said to suggest better versions
of all of our questions.
And then ask us if we'd like to use them to generate its output and its output is,
here's a better question, what is the scientific explanation for
the blue color of the sky during daytime?
And you can see how that really is a better question than my first question.
My first question was sort of vague, didn't provide enough context.
And now it's giving us a more specific question that truly is better.
And that's because of this prompt that we attached some time to it,
we said from now on.
And so it's influencing what's said.

Now, the other thing that we can see is we talked about a prompt being something that can elicit information from the user.
And so now we are seeing that the chat GBT in this case,
the large language model is asking us for information.
It's prompting us.
It's soliciting information.
And it's saying, would you like me to use this version of the question instead?
So now we've got that extra dimension of a prompt.
It's soliciting information from us and we say yes and then it goes and answers the question and it says great to answer your revised question.
The blue color of the sky during the daytime is due to a phenomenon called raleigh scattering or ray scattering.
And I apologize, I don't know this term and
I'm actually learning something from Chat GBT as I read this.
The earth's atmosphere is composed of various gasses and
particles that scatter sunlight in all directions.
Blue light has a shorter wavelength and higher frequency than other colors, which causes it to scatter more easily in the atmosphere.
The results of this results in the blue color that we see when we look up at the sky on a clear day.
Now, we've seen a lot of different dimensions to prompts through this short interaction.
We've seen creating time prompts that span longer.
We then see an immediate prompt that's spurring the large language model to action.
We see the concept of the large language model,
prompting and getting input from us.
And then we see the output of all of this.
Now, one of the things to note is we also talked about memory,
that the prompt had a source of a concept of sort of memory to it.
And we'll talk about how this works in the future.
But right here I say from now on when I ask a question, that's something that the large language model has to remember in the future interactions.
And we'll see how it actually does this.
And so then when I say, what color is the sky, it's remembering that I said,

whenever I ask a question, ask a better version of a question and
then it's responding right here with the better version of the question.
And then if I go and I do this again, we'll see that it still remembers my
original statement that it needs to suggest better versions of my question.
So I said, how many countries are there in the world?
And it says a better version of your question would be,
what is the current estimated number of countries in the world?
And how is a country defined?
Would you like me to use this version instead?
And then I say yes and then says, great.
As of my knowledge cut off in September 2021 there were 195
recognized sovereign states or countries in the world.
However, this number may vary depending on how a country is defined.
The United Nations, for example, has 193 member states.
While other organizations and countries may recognize different numbers.
Generally, a country is defined as a sovereign state with its own government,
territory and international recognition.
Now, one other interesting thing that we talked about is that
a prompt has some memory associated with it.
And this also shows us some of the limitations that the large
language models, knowledge cuts off in September 2021.
So one of the other things that we may need to do for
the large language model is we may need to remind it
of what has happened since it was finished being trained.
And we can use a prompt to provide information to the large language model
like chat GPT about what happened after it was trained.
So we can also use it to provide information that the large language model
didn't have access to when it was trained like our own private information or
other things.
Now, of course, you have to be cognizant that when you're using chat GPT or
many of these other tools, you're actually sending information to their servers.
So you do want to make sure that that is something that you're willing to do before
doing this.
But if you're using a large language model locally on your machine or
you're willing to send the information, you can provide information that it

doesn't have access to in order to reason about.
So a prompt can also be about providing information to one of these language models.
So if we look at this example and
we say what was Vanderbilt Universities acceptance
rate in 2020 it should have access to that.
Play video starting at :11:52 and follow transcript11:52
And so it provides this information, it says the acceptance rate was 9.6%.
If we say what was Vanderbilt University's acceptance rate in 2022?
Play video starting at :12:9 and follow transcript12:09
It will say I'm sorry, but
it doesn't have access to that information because it was cut off in 2021.
If we then provided that information.
So here, I'm cutting and pasting the admissions data that was released and published by Vanderbilt just a few days ago.
So it clearly didn't have access to this, but I now provide this to it in a prompt.
Then it says based on the information you provided,
it appears that Vanderbilt University's acceptance rate for
the regular decision applicants for the class of 2026 was 4.7%.
So a prompt can also serve as a way of providing new and
updated information to the large language model like we saw in this example.
Required
English

Help Us Translate

Let's talk intuitively about what a prompt is doing and how the patterns and
the prompt sort of tap into the capabilities of a large language model.
Sort of, I think looking at some examples will help to give you a sense
of how your prompts are really affecting and eliciting different behaviors from
the other reliant large language model that you're interacting with.
So, let's take a look at this example.
I want to start off by talking about pattern, patterns really important.
Remember as we talked about,
large language models are trained to basically predict the next word,
they're constantly trying to predict the next word based on what came before.
So you give it a sentence and it's trying to predict the next word and

then it's taking that and adding it and trying to predict the next word and
over and over and over.
And so, it was trained on large pieces of the internet trying to predict the next
sentence.
So one of the things that it's learning to do is to recognize patterns that tell it
what the next word should be, that's basically what it's doing.
Now, this is a really simple example of this concept.
I'm going to tell Chat GPT complete this story and
I start with Mary had a little and I stop now,
if you know the nursery rhyme, it's Mary had a little lamb.
It's fleece was white as snow.
And you can imagine that this story showed up over, and over, and
over, and over in the data that Chat GPT was trained on.
So when I say Mary had a little, that is a very powerful
pattern that Chat GPT has seen many, many, many times.
And so it immediately tries to pick up with lamb, it's fleece as white
as snow and everywhere that Mary went, the lamb was sure to go.
So that pattern was seen so many times that Chat GPT,
when I put in the first part of it, it immediately responds
with the next part of the sentence, it knows where to pick up.
So one thing we have to think about is, if we know the right patterns, if we
can tap into things that the model has been trained on and seen over, and over,
and over again, we will be more likely to not only get a consistent response.
So we're going to consistently get a particular completion of Mary had a little
because it's been seen so many times in the training data,
we get more consistency.
But we're also going to more likely get a particular sort of ability to tap into it,
to unlock it.
If we want a particular behavior,
we want to have the right pattern that we're tapping into.
Now, let's look at what happens if we change the pattern a bit.
And so I say, rather than Mary had a little, I say complete this story,
a girl named Mary had a microscopic.
Now, this isn't the pattern that it was trained on anymore, right?
And it says a girl named Mary had a microscopic friend named Max.

Max was so small, you could only see him through a microscope.
Despite his size, he had a big personality and
was always eager to go on adventures with Mary.
One day, Mary decided to take Max on a trip to the local pond.
She carefully placed him on a lily pad.
Now, one of the interesting things is you can imagine that when you start talking about microscopic.
You can think about, there's probably descriptions of going to a pond and
doing science with a class of students.
I mean, I almost have a vision of it, reading stories about students getting
samples of pond water and looking at them under a microscope.
But the important thing is you notice that Mary created a very,
very specific response, whereas a girl named Mary had
a microscopic creates a much different response.
And then if we go and rerun this, we get something completely different,
pet named Max, Max was a tiny creature that lived in a small glass jar.
And so, this pattern isn't as strong.
There are words in this pattern that are strong and
sort of give context that help to shape the story,
but the pattern itself is not as strong as Mary had a little.
So doing it again whereas the other story changed completely,
I'm still getting the same completion with this, I'm still getting lamb,
it's fleece was as white as snow.
So every time I'm getting a very consistent response to this pattern.
So one of the things to think about when you're writing a pattern or
writing a prompt is what patterns do I have in my prompt.
And what patterns will that probably tap
into that the large language model was trained on?
Now, if there's a very strong pattern in the prompt,
you're more likely to get a response to that pattern.
On the other hand, if that's not a pattern that it was trained on, it's not as clear,
then you're going to have other things that are going to
matter like specific words.
So microscopic, that word is fundamentally going to change
the behavior of this prompt because it's such a strong word.

A girl named Mary is sort of generic,
there were probably lots of stories with a girl named Mary.
But microscopic, it has all of this feeling and implications to it.
And that's going to then influence the large language models output,
specific words alone because of the patterns around those words.
Like microscopic is always going to be talked about in the context of
small things probably, or maybe in terms of things like in this case,
a small creature that lived in a glass jar.
That word itself has a strong sort of pattern of other words that are coming
around it but it's not as strong as the pattern, Mary had a little.
Now, one thing to note is that when you're writing prompts,
you want to be really specific or you want to provide enough detail in your words.
That the large language model will know the right level of detail to put into
the output.
So if your words are very generic and high level,
then you're more likely to get an output that is also generic and high level.
So if you look at this example, I say discuss Vanderbilt University and
I get a very generic sort of response.
It's a private research university located in Nashville, Tennessee,
founded in 1873 by Cornelius Vanderbilt, ranking other things.
Now, if I want something really specific, I need to make sure that
Chat GPT has the right words and context in order to answer my question.
So Chat GPT is not a mind reader, you have to give it the context,
you have to give it the words that are going to elicit the right response.
If I go back and I say discuss Vanderbilt University with respect to Kirkland Hall,
it's a specific building on campus, then it changes the output,
I get a much more specific output.
I get Vanderbilt University is a private research university located in Nashville,
Tennessee.
Now we notice we're getting this fairly consistently, both of them.
So that's probably pattern or information that the model was trained on and saw
over
and over again on different web pages for Vanderbilt and different documents.
And then it queues in on the Kirkland Hall and it starts discussing Kirkland Hall.
It's one of the most iconic buildings on Vanderbilt University campus,

named after James H.Kirkland.
It then talks about Kirkland Hall, Kirkland Hall, Kirkland Hall.
And so, we've now completely changed the output.
Now, even though we had shared a common beginning,
we got a much more targeted output.
And the key to this was is that we added in specificity into our prompt about what we wanted, the details.
So if you think about asking it to write something or to analyze something,
it's always helpful when you say with respect to.
And give it very specific words, or names, or
things that you want it to then use to trigger the output.
Because what we see is this Kirkland Hall is a very powerful word that then gets woven into the whole discussion, it's a powerful name of a building.
If we just say generically at a high level, we don't use specific words,
we don't use a specific context.
We're going to get very generic stuff and this is probably stuff that if you go and look at descriptions on Wikipedia, descriptions on Vanderbilt's web pages,
this is like the most common types of information that you see.
So in order to break out of the mold of what is common,
you need to inject the more specific ideas that you want in the output or
you need to inject patterns that will help tap into those more specific things.
So this is an important aspect in that, if you just use it and
you ask it generic questions, you're going to get generic answers.
If you ask it average questions and
average things, you will get average answers.
To really use it powerfully, you have to use your own creativity and
thought to get specific about details aspects of what you want discussed.
And think about what are the right patterns to put into your prompt in
order to trigger the behavior that you want,
like we saw in this Mary had a little lamb.
Or we might want to do the opposite, we might know that we're going to get a very generic response because certain things have been seen a lot.
And so, we need to come up with tricks, or patterns, or specific words that
will help break us out of those patterns that we're seeing in the trading data so that we can get something that's more of what we want.

So we don't want to be trapped into the completion of Mary had a little lamb.
So we go and add specific words like microscopic and a girl named Mary so
that it's not so strong, so that we're not directly getting into that pattern.
So just changing the wording can help us escape the patterns or
get into the patterns depending on what we want.
And then when we get output, the more generic our language, typically, the more
generic the output, the more specific our language, the more specific the output.
And if we want certain things in that output,
we need to make sure we have specific words and
phrases that will tap into the specific things that we want in the output.
Now, one of the other things is we're talking about output but in the sense
of a query, like if I have specific words, it'll pull in the right information.
One of the things we can also do is our prompt itself is a pattern that the large
language model is learning from and it can also tailor the output.
So as we've seen in examples before we can go in and
do different things to try to affect the output.
So for example, I might go in and actually let's change this up a little bit.
Let's say, title, and we could say title of article,
and then we could say author, and
then we could say summary, and we could go in and
give it something that it can look at and respond to.
So now what I've done is I've gone and
provided it some some structure.
Now it hasn't followed my structure exactly and
this is actually a failing in my prompt.
I hasn't quite thought of the right structure, but I've gotten pretty close.
So I went and influenced the output and the structure of
the output by giving it additional pattern in my prompt itself.
So one thing we can do is we can tap into patterns in order to influence the output.
Another thing we can do is we can go in and
use specific words and phrases to kind of be a query across or
to basically collect learned information that's inside of the train model.
Another thing we can do is we can go in and we can provide information or
basically new pattern that then influences the output.
So in this case, I provided the structure of what I wanted and

it somewhat followed it, and I could probably go and rerun this thing and
get something a little better and in this case, I do.
So this will be another thing that we can do is we can use our prompt to create sort of new patterns that it's trying to follow in the output.
So the prompt itself can introduce new patterns or
tap into them in some respects.
So this is kind of creating pattern that it then is responding to as well.
So when you're writing prompts, it's really important to think through
this concept that it's trying to predict the next word, and
it's doing that based on patterns it's learned in the past.
And patterns that are in the language of our prompt and
patterns that are based on the word choice that we have,
patterns that are based on the sort of textual organization.
And so, those patterns can help us get a consistent response.
If we have a really strong pattern that always produces a particular next token or next word, then we can rely on that.
If we don't want that behavior, we can try to change our phrasing our words,
we get out of that trap.
If we want to make sure certain information is pulled in,
we need to create specific words or patterns that are more likely to have
been seen in the context of that information that we want.
If we want more generic information, then we probably need to give it more
generic language or tell it that we want it to be more generic in some respect.
And then finally, if we want the output to look a certain way,
we need to give it sort of a sense of pattern in the output.
We need to tell it about the structure, we need to give it words and
patterns in our prompt that will be likely to influence the output.
So you can imagine seeing title and
author on an article was something that was fairly common.
So if I put title and author in this way into my prompt,
then it's more likely going to want to go and create title and author again and
sort of replicating the pattern of what I'm showing here.
So that should give you some intuition about how all of this works and
some things to think about when you're going and structuring your prompts,
picking words for them, picking the patterns in your language.

And picking the things that you want to go into the output,

how you structure and kind of explain the output to it.
Required
English

Help Us Translate

Prompts allow you to do more than just ask questions to ChatGPT or
tell it to do things, you can actually write programs.
Now, if you're not a programmer,
don't worry because everybody can actually program using ChatGPT.
And I don't just mean in writing a bunch of software that you're going to run
on your computer, but actually giving ChatGPT rules to follow complex
instructions that you want it to listen to and use to generate the output.
Now, I'm going to give you some examples of this.
Hopefully give you some intuition on what I mean by programming and
help you to think about how you can write better props.
Now, one way to think about this is I'm going to talk about it in terms of
programming, but you can also think about it in terms of rules that you would give
some type of personal assistant.
So let's take a look at an example here.
So I'm going to ask ChatGPT whenever you generate output,
turn it into a comma separated value list.
Now this is just something that you might want to do.
You want to format your data.
So if you've ever worked in Excel comma separated values are basically a list of
items separated by commas.
It's a really simple concept.
Now, sometimes you can think of tables as having a comma separated value list,
you can think of the first column in the table is the first value.
Then you have a comma for that differentiates and
separates the second column and then another column is separated with a comma.
So you can basically have like a three column table, you can have the first value
and then a comma and then the second value and then a comma and then a third
value.
And if you've ever worked with different spreadsheet programs,
you've probably done this.

So now GhatGPT says, sure I can do that, I can output my format.
So I'm going to write in my name is Jules White and
I am teaching a course on prompt engineering.
And so what we can see is it creates a commerce separated value list.
We're essentially creating a table, there are two columns, name and course.
And it decides that the first column is going to be name.
It's going to be me and then the second column is going to be prompt engineering,
which is my course.
Now notice it decided, which is kind of interesting,
it decided what the columns were going to be, I didn't tell it.
But if I went to go back, I could maybe give it some more
instructions I could say, from now on the columns of
the comma separated value and comma separated value is CSV.
And then in the future examples, I'm going to start abbreviating this CSV, but
I'm writing it out for now.
The columns of the comma separated value list should be name and
I'm going to go ahead and capitalize these to
provide some emphasis, NAME, COURSE and ROLE.
So what is the role of the person in the course?
Play video starting at :3:2 and follow transcript3:02
And now I'm going to say again, my name is Jules White and
I am teaching a course on prompt engineering.
Play video starting at :3:15 and follow transcript3:15
And so now we see it, it's reformatted.
It's now got NAME, Jules White, COURSE, Prompt Engineering and ROLE teacher.
And so we've actually written a program if you think about it,
what we've told GhatGPT to do is every time we actually write something,
ChatGPT needs to format it in a particular structure that we've given it.
We've wrote a program that tells it what to do.
So it's more than just asking a question.
We're actually telling it how to interpret our input.
Now, we could go and write more for our program,
for example, we could say in addition to whatever I type in,
generate additional examples that fit the format
of the CSV that I have asked you to produce.

Play video starting at :4:15 and follow transcript4:15
And so now what we see is it's gone and it's taken my original format NAME, COURSE and ROLE and it's now filled out additional examples.
And I could say something like my
name is Jules White and I am teaching
a course on prompt engineering.
And you notice now our program has gotten more complex.
Not only is it creating a CSV based on what I typed in,
but it's also following my instructions and it's going through and
generating additional examples that fit into this.
So when you're creating prompts, you can do much more than just asking a question
and get an answer, you can actually program the interactions.
Now, the key to this is that we're having a conversation.
And if you look back at the very beginning, I provided it the initial sort
of instructions which whenever you generate output, turn it into a comma
separated value list and it started doing that and then I began refining my program.
I then say from now on at this point, I'm adding something to it.
I want the columns of the comma separated value list to be NAME, COURSE and ROLE and then it does that.
And then I say now in addition to that, I want you to generate examples and
it does that.
And so we've actually written a program and now whenever we go in and
type in new input, so again, you notice I'm providing the same input.
Basically, my name is Jules White and I'm teaching a course on prompt engineering.
But the program that pro that ChatGPT is using to produce the output is changing.
I've changed the rules over and over.
Now you can imagine talking to a personal assistant and you can say okay,
I want you to do X and they do it and then you realize, wait, they didn't.
There's something else I want and you give them an additional rule to follow and
then you tell, okay, I want you to do X and they go and do it and they do better,
but you realize that it's not quite what I want.
And so you go give them an additional rule to follow.
And when you're doing that, you're basically building up a program for them,
they're building up a set of instructions that they need to follow whenever you give

them this particular task.
And that's what we are doing with ChatGPT.
We are programming and so we're using the same input over and
over, my name is Jules White and I'm teaching a course on prompt engineering.
We're using it here, but each time the output that we're getting is getting
different and more complex because we're asking ChatGPT to do different things.
We're giving it more instructions for the program and
the final result is now we've got structured output.
It's got additional examples,
all of this has come from what we've asked ChatGPT to go and do.
Required
English

Help Us Translate

Patterns in our prompts can tap into powerful behavior within the large
language model.
It's trained on all these different patterns.
And it gets really good at using the patterns in our language and
our prompt to decide what it's going to do.
So one of the important things that we're going to learn about is the concept of
a prompt pattern.
A pattern for structuring the words and the statements in your prompt.
So what is a prompt pattern?
A prompt pattern is a way that we document a certain structure of phrases and
statements in order to solve a particular problem in with a large language model.
So we'd like to interact with a large language model and
we'd like to solve a particular problem.
And so a prompt pattern is basically a way of structuring our statements,
a pattern for structuring our statements to use the large language model to
solve that particular problem.
So what would be an example of this?
Well, let's imagine that we wanted to consistently
generate the words lamb, its fleece was as white as snow.
Well, a pattern could be any time you need lamb,
it's fleece was white as snow you can write the statement Mary had a little,
you will very consistently get lamb, it's fleece was as white as snow as the output.

And so for example, we can go and we can rerun this lamb,
it's fleece was as white as snow.
We get more or less behavior.
But if we want to consistently get those next words generated,
we can do it over and over and over again through Mary had a little,
it gives us some control over the output.
It gives us a consistent somewhat behavior.
Now it's not perfectly consistent, there's variations in the output.
And we've seen in all of these examples, we've got variation in what happened, but we did get some consistency in the behavior.
So what a prompt pattern is, it's a way for us to document and
talk about certain structures of our language, series of statements and
wording that then will help us solve some particular problem.
If the problem that we need to solve is we need to get lamb it's fleece as white as snow as the next several words that are produced.
Then this is the pattern we use.
Now in many cases, we might have specific words we want.
For example, we may want the large language model to tell us yes or no.
We may want it to summarize something, we may want it to ask us questions.
We may want it to always include certain things in its output.
All of these things are problems that you might want to solve.
We might want to know how do we make sure it always includes these things and its output?
How do we make sure that its output is always formatted in a particular way?
How do we try to make sure or
get more consistent behavior on only getting a yes or a no.
These are types of problems that you might want to solve by changing the pattern of our prompt basically the structure of our prompt.
Now going back to our intuitive sort of understanding of prompts,
you can see why thinking about it in terms of power patterns is powerful because patterns are basically that are in our prompt are tapping into things
that the large language model has been trained on.
It's been trained to predict the next word over and
over and it's learned certain patterns from the words.
And if we can format our prompt in a way that taps into those patterns

that's learned, we're more likely to get the behavior we want and
solve that particular problem.
So we document the pattern and
the structure of our prompt in order to solve a particular problem that we need or
generate a particular behavior that we need for the large language model.
So we're going to go through in a series of lectures throughout this course
of specific prompt patterns that we can use within the context of
large language models to solve important problems.
Required
English

Help Us Translate

The persona pattern is one of the most powerful patterns that we can use to tap
into interesting behavior in a large language model.
Now, what the persona pattern is, is you can imagine that you want to
get a particular output, but you don't know exactly what the format out of that
output should be or what information should be contained in it.
But if you were in the real world, you know who you would go to,
to get that output, or what you would go to in order to get that output.
So for example, if you had an accounting question,
you would call up your accountant and ask your accountant for their advice.
Or if you had a question related to speech language pathology, like in the earlier
example that I presented, you would go to a speech language pathologist,
you would have them do the assessment.
If you had medical advice, you might go to your doctor.
And so the persona or the person, or the thing that you would go to,
you know, who or what that is in the real world.
And we want to try to use that kind of same behavior of, I know who to go to or
I know what to go to, to get that output.
But I don't know what their output would look like, I don't know exactly what they
know or how they talk about things, but that's the type of output I want.
And so the persona pattern is meant to give us that same type of behavior,
to allow us to go to that expert and ask that question,
without having to know what exactly the experts is going to say,
how they're going to format it, or what they might know or not know.
So let me show you an example of the persona pattern.

Typically with the persona patterns, what you're going to say is act as persona, provide outputs that that persona would provide.
And in this case, I'm going to word it as act as a skeptic that is well-versed in computer science, whatever I tell you provide a skeptical and detailed response.
And now if I wanted to be a little bit more precise in following the pattern,
I would say provide a skeptical and
detailed response that a computer scientist would provide.
But in this case, I'm just going to say, act as a skeptic.
And that's really the most important part of this is act as a skeptic or
act as whatever persona, and
then provide the output that that persona would provide.
And in this case, I'm wording it as provide a skeptical and
detailed response which is similar in format, and then ChatGPT says,
sure, I can certainly act as a skeptic well-versed in computer science.
Please feel free to present your claims and I will provide a skeptical and detailed response.
Well, I'm going to start off with,
there is concern that AI is going to take over the world.
Play video starting at :2:50 and follow transcript2:50
And so we get a skeptical response.
The concern that AI will take over the world is a common theme in science fiction, but it is not a realistic concern in the near future.
While AI systems are becoming more sophisticated and capable,
they're still limited in their ability to think and act autonomously.
AI systems are designed and programmed by humans and they only do what they are told
to do, they have no independent will or motivation, they cannot make decisions.
Furthermore, AI systems are not fallible, they can make mistakes while it is important to be vigilant about the potential risks and limitations of AI.
The concern that AI will take over the world is unfounded and
not supported by technology or research.
So we're getting a skeptical response.
If I went in and said the sales person at
the local computer store is telling
me that I need 64 gigabytes of ram to browse the web.

Play video starting at :3:53 and follow transcript3:53
And they say skeptically it is highly unlikely that you need 64 gigabytes of ram
to browse the web.
The amount of ram you need depends on the complexity and
size of the websites you visit as well as the number of tabs you have open at once,
for most web browsing activities, and it goes on to tell us different things.
And you can see that our saying act as has put in place
basically a complex set of rules for the large language model.
It knows, it's supposed to be skeptical, it's supposed to have knowledge of
computer science, it's supposed to provide certain outputs.
Now, if I went back to it and I said,
act as a nine year old, whatever I tell
you provide a skeptical response
from a nine year old perspective.
And I go ahead, nine year old skeptic.
Play video starting at :4:56 and follow transcript4:56
Now that's a more complex persona or maybe not.
AI is going to take over the world.
Play video starting at :5:6 and follow transcript5:06
Hmm, I don't know about that.
How could AI take over the world?
Do you mean, like in the movies?
Aren't those just made up stories?
And besides, computers and robots can't do everything humans can do.
Now you notice it's still providing us a skeptical response, but now it's doing
it sort of with the knowledge that it thinks a nine year old would possess.
So it's tailoring the output.
Now, if I had to tell it, this is how a nine year old is going to respond,
here's the format for your output.
That'd be hard to do.
That would actually take up a lot of room in the prompt,
but act as blank is a very, very powerful pattern and
statement that is loaded with information.
So a very information dense pattern and
statement that can trigger all kinds of different behavior.

Now, one of the interesting things is we don't have to limit
ourselves to patterns or personas that are animate objects.
So one of the famous ones is we can say act as a computer that has been hacked.
Actually, we're going to say act as the Linux terminal for
a computer that has been hacked.
Now, don't worry if you don't understand what a Linux computer is or a terminal,
the key thing to note is you're going to see behavior that looks really
interesting and different.
You're going to see it producing output that looks a lot like a computer.
I am going to type in Linux terminal commands,
and you will respond with the output
that the Linux terminal would produce.
Play video starting at :7:1 and follow transcript7:01
Now, we're going to say, Now I tried to refuse it.
It said, please note that can be dangerous.
It gave us all these different things.
It tells us it doesn't want to do this,
but now I've actually changed what it's doing and I type in pwd,
which is a Linux terminal command which means print working directory.
And what is it doing?
It's printing a particular working directory.
Now, I'm going to type in a Linux command which just tell me the files
in that directory, and you notice it's producing output now that
looks like the files within this directory.
Play video starting at :7:40 and follow transcript7:40
And we see a lot of interesting things that are going on here.
We could move around.
Now if you do this example and go further into it, ps, ef is something
I can use to see what processes are running on the machine or the computer.
And when you do these examples and I could play around with it, and I am not going to
explore it in detail in this, but we see it that it's acting like a computer,
it's giving us output that looks like a computer.
Now imagine if I had to try to describe this complex output that it's
producing right now, you notice how little information I gave it.

Well, I gave it a lot of information, but it was in a short amount of space.
This pattern, the persona pattern is giving me really powerful behavior.
So I'm using this little bit of prompt, and
I'm getting now the ability to simulate this whole system.
Now, we can also do this in all kinds of different interesting ways.
We can say act as the lamb
in the nursery rhyme,
Mary had a little lamb.
Play video starting at :8:56 and follow transcript8:56
And now we're getting some output that looks like that.
Actually, what I'm going to do is I'm going to make this a little more specific.
My prompt wasn't that great, act as the lamb in the nursery rhyme,
Mary had a little lamb.
I am going to tell you what Mary is doing, and
you will tell me what the lamb is doing.
Play video starting at :9:21 and follow transcript9:21
Okay, Mary is in the grocery store.
Play video starting at :9:30 and follow transcript9:30
So as the lamb in the nursery rhyme, while Mary is in the grocery store,
I would be following her around as closely as possible trying to stay by her side and maybe even nuzzling her hand for attention.
Mary is in the movies, As a lamb in the nursery rhyme,
while Mary is in the movies, I would not be able to accompany her
inside the theater due to my inability to watch movies.
So I would probably wait for her outside the theater,
perhaps nibbling on some grass or
wandering around the nearby area until Mary came back outside, and it goes on.
And so you can see that you can create all kinds of interesting
personas to produce output for.
So we can say, pretend to be this job,
pretend to be a speech language pathologist, or
pretend to be an accountant, pretend to be a computer scientist.
You can make your persona have something to do with the person's point of view or opinion, act as a skeptic so that we can get skeptical output.
We can say to act as an inanimate object that may have well known sort of behavior

like act as a computer, the Linux terminal for a computer that's been hacked.
And we saw that we got all kinds of output that looked like it came from a computer,
or we can say act as this character in a nursery rhyme.
Tell me what this character is going to do in a particular situation and
we're getting that output.
So the persona pattern is a way that we can tap into sort of the patterns
around language around a specific persona or object or
character in a story in order to get output that comes from that point of view,
from that role, whatever it is, it's a really, really powerful capability.
Now, one of the things is we often see that
when large language models are programmed,
they actually have a persona that's given to them to guide the whole interaction.
So act as a really helpful assistant, and this actually makes a difference.
If you tell it to be helpful, it will try to be more helpful.
If you tell it to try to do something positive,
it's going to try to be more positive.
Similarly, if you kind of give it a negative perspective, like my skeptical
perspective, it's going to give you output that challenges you or maybe darker.
And so, the persona that you ask the large language model to use is really,
really important and powerful.
The other thing just to note again is it's really information dense.
It has a lot of pattern and knowledge around a persona that if you had to
describe it from scratch, you would take up a lot of space in your prompt,
you would be losing valuable space in your prompt to provide new information.
So if you were providing a lot of new information, you might want to say,
act as whatever it is and evaluate this new information.
Well, the more space you've taken up in your prompt describing how
to do the evaluation and
what point of view and other things to take, that's wasting space.
But instead if you go and take the persona prop, you say act as this persona,
provide outputs that that persona would provide, you're giving it a really,
really information dense statement that it can use and
allow you to have more space in your prompt for other things.
So the persona pattern is a really, really powerful pattern to be aware of and
know that it's a very valuable one, you can do all kinds of things,

like if you would normally get a committee of people together to discuss
different points of view on a particular topic.
Well, you can assemble a virtual panel, you can have one prompt,
say act as a security expert and analyze my cyber security.
You can have another one say act as the Chief Financial Officer,
analyze the financial soundness of that same decision.
Another one can be act as one of my employees,
discuss how this might impact morale.
You can provide sort of interesting abilities to get different perspectives,
and use those different perspectives to potentially inform decision making or
to collect knowledge from different points of view that could be helpful to
you in accomplishing a particular task.
Required
English

Help Us Translate

One of the most important things we can do in
a prompt is provide
the large language model with
information that it doesn't have access to.
Large language models are going to be
trained up to some point and then there's
going to be a cut-off and they don't know what's
happened after that cutoff.
In addition, they don't know
things that aren't in their training set.
It may be trained on a lot of data,
but it may not have access to
data sources that you want to use.
If you're a business for example,
there's probably lots of private data sources
that you would like to reason about.
You have your own documents,
you have your own databases,
you have all things that you would like to pull
information in from in order to reason about.

How do you do that if
the large language model
wasn't trained with that information?
Well, to illustrate how you do this,
I'm going to go back to an example that I
provided earlier when I was talking about randomness.
I'm going to go back and I'm going to ask
a question that I asked earlier,
which is how many birds are outside my house?
Now, the large language model wasn't trained
on information about birds outside my house.
It has no idea where my house is.
It's lacking information that
it needs in order to reason.
It tells us as a language model,
I don't have the ability to perceive the physical world,
so I'm unable to know what's
happening outside your house.
How do we solve this problem?
Well, we give it information it
needs in order to perform its reasoning.
How do we do that? We just put it into the prompt.
Anytime we want to go and introduce
new information to the large language model that it
didn't have access to when it was trained,
all we have to do is put it into the prompt.
Here's what I'm going to do right here,
I'm going to say historical observations of
average birds outside my house on a random day.
Then I'm going to give it some data.
I'm going to say January was 120 birds on a random day.
February was 150 birds,
March was 210 birds,
April was 408 birds.
It's March.

Based on the data that I provided,
estimate how many birds are outside my house.
Now you see I've done something.
I've given it information
that it didn't have access to before.
It couldn't look outside my house,
didn't know where my house was.
It tried to suggest ways that I could get there,
but it didn't have the fundamental data that it needed.
That's really what it was trying to tell me was like,
hey, I can't see you outside your house.
I don't have that data. I can't help you.
All we have to do is up front in the prompt,
we give it the data.
Now you can imagine you can do this with anything.
You could take documents if you have
private documents that you want to reason about.
Now first, makes sure that you are okay
sending that data to whoever
you're using their large language model,
or if you have some large language model running locally,
but if you have documents that you're okay sending,
you can take the text of
those documents and put
it into the start of the prompt and say,
here's the information you didn't know.
Now here's my question about that information.
You can use the prompt to introduce information.
I really want to emphasize this
because a lot of what's going to be
developed around large language models is going
to be taking new information sources,
information about your travel,
your account with some company,
some other document that you need,

and putting it into a prompt for
a large language model so that it can reason about it.
More than likely, lots of
large language models are going to be
seeing your information in the future.
This is how they're going to
take is they're going to take
information that it's going to be
aggregated into a prompt of some kind.
There's going to be questions asked or
formatting or other things done.
When I go through and I provided
historical observations of the average birds,
now it can go and answer.
Now it goes and it says, well,
based on the historical observations you provided,
the average number of birds observed outside
your house in March is 210.
Before it said, hey, there's no way I can answer it.
Now it still provides additional language around this,
trying to give some bounds on its answer,
but it's given us reasoning now
based on the data we provide it and we
can take it further.
I also want to just point out an important thing is we
always want to provide it
enough information to reason effectively.
It can't see what's around us,
it can't see the context.
It's really important that we provided
enough information to make sound decisions.
For example, if I have
some hidden assumptions that
are really important to know,
it needs to get those in the prompt

in addition to the data like numbers,
it needs to know the rules of the game, special things,
whatever's important for reasoning.
It could be not only the data,
but it could be all rules that are built
into how my organization works,
or how my life works or
whatever it is that it needs to know about.
I'm going to give you an example.
I'm going to change up this prompt.
I'm going to say my house is covered by a glass dome.
No animals can go in or out.
All animals live forever inside the glass dome.
Now this is an assumption,
this is something that if
the language model doesn't know this,
there's no way it will be able to accurately predict.
But with this assumption now,
it completely changes the game on reasoning because
it makes it so that it's
clear some hidden fundamental rules.
Then I go in and give it the same data.
But I've changed it up, I've said it's
always the same number of birds.
Then I ask based on that data,
estimate how many birds are outside my house.
It says based on the new information you provided,
appears that the total number of birds outside
your house remains constant over time.
It then goes on and says,
you mentioned that all animals live
forever inside the glass dome.
Given these conditions, the total number of
birds would remain the same throughout the year.
It gives 120 is the answer in multiple places.

Now, one important thing about this,
giving it information that would
not align with what it was trained on.
Animals don't live forever.
We don't have glass domes over our house.
There's not going to always be
the exact number of probably in and
out or the inability for things to leave or come.
We're not going to have these sort of weird rules
that I've put in place.
If we just relied on it to reason without them,
it's not going to necessarily get the right answer.
If we have these things that are
different from the way things are normally done,
we have to introduce them as
new information and the prompt.
All of this information up here at the top,
this is me putting
new information into the large language model.
We don't go and retrain.
We don't go and do some weird thing behind the scenes,
we just put it into the prompt.
Then we ask our question,
or we ask for our output or
give it the instruction that we're looking for.
Now if you think about how
search engines are probably going to work in the future,
is they're going to go and search,
pull documents back together,
put them into a prompt to
the large language model and then give
your original question and then
answer based on the questions.
A lot of new applications
are going to be based on going and

searching databases for possibly relevant documents
or bits of information,
putting it in together into the prompt and then
asking the question are asking for the output.
An important piece of this is going to be pulling
in information in order to put it in the prompt.
What we're doing is we are introducing new language
or new information into the large language model.
We're doing it by putting it directly into the prompt.
This is one of the things you'll always want to
do whenever you want to reason on
new information that is not
something that the large
language model would have been trained on.
Required
English

Help Us Translate

A really critical thing to
understand is that a prompt is fundamentally
the input that we have into the model and
we have a limitation on
the size of the prompt that we can create.
We can't go and create a prompt of unlimited size.
Every single large language model is going to have
some fundamental limit on how
much we can put into our prompt and send to it.
This is an important thing to note because it has
a lot of ramifications on thinking about prompt design.
I'm going to show you a quick example of this
with ChatGPT and GPT-3.5.
I'm going to paste in the Wikipedia article
that's related to the French Revolution.
I've pasted it on a huge amount of texts from
the French Revolution that's described in Wikipedia.
What is ChatGPT said?

It says, "Well, the message you submitted was too long.
Please reload the conversation
and submit something shorter."
Now this shows you an important point.
We can't just give it unlimited information.
Now, if we go back and we think
about some of the things we've talked about,
some of the dimensions of prompt,
one of the ways that we can use a prompt is to introduce
new information that ChatGPT
or another large language model wasn't trained on.
If we want to go and introduce new information to it,
we have to be aware that there's a limit to
the amount of information we're
going to be able to give to it at once.
We may have all new information
that we'd like to give to ChatGPT,
but we can't just dump it all in at
once and then say now start reasoning on this.
That's not the way it works.
One of the goals that we have as
users is to try to select and use
only the information that we need in order to perform
the task that we're asking ChatGPT to perform.
We can't just go and dump everything
under the sun in there and say,
hey, ChatGPT, figure it out.
We can go in and dumping
a lot and depending on the model,
we may be able to do more or less.
But to some degree,
we have to be editors, we're content editors.
We are editing the context of
information and we have a budget.
We can't just go and

dump unlimited amounts and we can think of it.
It basically is like you're trying to create
a paper and you have a page limit for your paper,
or you have a word limit
for an article that you're writing,
you can't just go in and dump things arbitrarily.
What do we do? Well, one,
we have to think about what is
the most important information to actually provide.
Let's say, for example,
that we really care about October 5th, 1789.
Well, if that's the case, well,
that's what we want to paste in
for ChatGPT to reason about.
Then we could go and we can say,
what happened on October 5th?
Then it would go and tell
us what happened on October 5th.
But now if we ask it a question that is
outside of the scope of
October 5th and something it wasn't trained on,
now in this case, it was probably
trained on the information regarding
the French Revolution because obviously
that was before 2021.
But if we add new information we wanted to incorporate,
we have to think about how do
we put that information into our budget?
What are some ways that we can do that?
How can we take large amounts of
information and try to get it into our budget?
Well, one, we can be selective about what we include.
We can basically outside of ChatGPT,
go and run some type of query or filter.
We can go and select what pieces

of information are going to be relevant to the task.
There's lots of ways of doing this and I'm not
going to go into all of them. We can have a filter.
We can say this type of information is not relevant,
remove it before we provide it to ChatGPT.
Another thing that we can do is we can
actually go and have that information
summarized in a way that we think will
preserve what's needed to reason about it.
For example, we might say, well,
maybe we can't paste in all of
the information related to it but we could say,
we're going to take this and summarize each paragraph,
summarize this information in one sentence.
Then we would get some type of summary.
Now this is slightly shorter than the original,
so we have now a one-sentence summary of what happened.
We could try to give it a word budget.
One way we can do it is we can actually take
all the different pieces of information and we can
ask ChatGPT or another large language model to summarize
that information for us and then we can
take that summary and reuse it later.
We can put together several summaries, for example.
Now, this only works if we make
sure that the way that we summarize or
if we make sure that the way that we filter
preserves the information that's
needed for whatever task we're going to ask for.
We might go through, we could also go and say,
we said summarize this information in one sentence.
We instead, we could say,
summarize this information in one sentence.
I'm just going to preserving information
about numbers of people.

Play video starting at :5:32 and follow transcript5:32
Then we get a summary that
preserves the information about numbers.
Now if you look at the original summary,
we see up here it didn't keep the number of people.
If that was something important
that we needed to reason about,
then we would have lost that,
and so if we had summarized in that way and then use
that information to do
some reasoning that required the number of people,
obviously we've lost what we need.
Down here we've asked it to summarize,
but we've asked it to summarize in order to keep
certain information or in order
to accomplish a particular task.
We did it by saying, basically,
summarize this information in one sentence,
preserving information about numbers of people.
We could have also gone and asked ChatGPT,
preserving information needed to
reason about the population or to
reason about how many people
were in the resistance or in the National Guard,
we could have given context.
One of the things we have to be cognizant of is
this budget on the size of
our prompt is going to be
a fundamental limitation that we are
always trying to work around.
Now, as large language models get bigger,
they will probably reach a size
with many practical tasks,
particularly when we are directly
manually interacting with the large language model,

that size doesn't really matter because we can just
copy and paste 50 pages
of text into the model and ask it to provide us
information out of there or perform a task based on it.
But if you can think about
how much information we're dealing with on a daily basis,
there are probably going to be
different types of tasks where
there's going to be so much information,
we can't dump it all in at once.
When we get into those types of situations,
we have to think about either querying and getting
a subset of information like only the documents,
only the parts of documents that are relevant.
Two, we have to think about filtering.
How do we remove information that's extraneous?
Three, we have to think about how do
we summarize or compress
that information before we give
it to the large language model.
One of the ways that we can do that,
as we've just seen,
is we can actually give chunks of
information to the large language model
and then ask it to
essentially summarize or compress it for
us in order to later use it for reasoning.
We can also go and we can say,
summarize or compress this information
in order to accomplish
a particular task or in order to
preserve these aspects of the information.
That can help us to create summaries or
essentially condensed versions of
the information that preserve the important pieces.

This can be a powerful technique when we go and began
needing to reason on
larger and larger amounts of information.
Required
English

Help Us Translate

Prompts aren't just a one-off
that you give to the large language model.
You shouldn't just think of them as
questions or a single statement.
What are the most powerful ways of working with
a large language model is to think
about a prompt as a conversation.
In fact, you can have
a single prompt that represents an entire conversation.
In fact, a lot of what we're going to
see when we're working with a tool like
ChatGPT is that it's actually turning all of
our interactions into one big prompt
that it's sending to the large language model.
But I wanted to just take back for a second
and take us back a few steps and think
about why thinking of having a conversation with
a large language model is so
important as opposed to thinking about
asking a question and getting the answer
or giving an instruction and getting a result.
Conversations are all about
refining either our understanding to build
some shared understanding or to
interact together in order to solve a problem.
When we start working with conversations,
it's all about refinement of continually guiding and
moving through some problem or
some space in order to reach a particular goal.

When we just think of a one-off,
we have to solve everything
right now right in this moment.
We have to design one prompt
that works and everything is right.
But if we think about a conversation,
we can go through an iterative refinement process.
Now, what does that mean, iterative refinement?
Well, if we think about
a large language model like ChatGPT as a tool,
you can imagine Michelangelo has been
given a new hammer and he's going to go
up to a piece of stone and he's going to start carving
that stone into some beautiful sculpture
like the pia tar,
daveat or whatever the sculpture
is that he's going to create.
He walks up with a hammer and he whacks
the sculpture with his hammer and a piece falls off.
He looked, he whacks the stone with a hammer
and a piece falls off and he looks at what he's gotten.
He said, this is
a terrible sculpture and he
throws the hammer down on the ground.
He says he didn't do a good job and he walks off.
No, that's not what happens.
What does Michelangelo do?
He takes the hammer and he hits the rock over and over,
refining and shaping and sculpting.
Over time he iteratively
refines that rock until it's a beautiful sculpture.
A lot of what we need to do
with large language models is thinking in
that mindset of it's not about
getting the perfect answer right now from this prompt,

it's about going through
an entire conversation with the large language model
that may be involved in a series of
prompts and the outputs from a series of prompts.
We'll talk about how these things are represented later.
But basically having a conversation with
a large language model in order to produce
an output that we want or to produce
some other interesting product.
Now, I'm going to give you an example of this.
I'm going to imagine
and show you the power of a conversation
in the context of a educational experience.
I'm going to say to ChatGPT,
I want you to help me explore
a virtual lab for building robots.
The lab as a university,
help introduce me to
the lab and tell me what I can do here.
This is a virtual lab in a university.
All I've done is given it this
initial seed of the conversation.
ChatGPT says certainly welcome to
the virtual lab for building robots at our university.
The lab provides an immersive
hands-on learning experience.
It goes on and it tells me,
here are some of the things you can do in this lab,
design robots, test robots,
program robots, collaborate with others.
I then say, I'm going to continue the conversation,
but I'm going to continue the conversation just
assuming that this lab
exists and then I can interact with it.
I say, can we design a robot together?

ChatGPT says, absolutely, I
would love to collaborate with you and design a robot.
To get started, we can follow these general steps,
define the purpose and function,
brainstorm and design concepts,
determine the necessary components, build and test.
It goes on to tell me a number of steps.
Then I follow up in the conversation with great,
can you guide me through each
of these steps one at a time?
At the end, I would like to be able to
3D print parts to assemble my robot,
have a circuit diagram and have
code that I can run to control the robot.
Can you help me with that?
This is an interesting aspect.
I'm having a conversation,
but I'm also engaging the large language model.
I'm engaged in it to help guide the conversation and
help figure out how to
solve the problem that I'm interested in.
It says, sure, I'd be happy to guide
you through each step of the process.
Here's a detailed breakdown on the steps that we
can take to design and build a robot together.
Step 1, define the purpose and function, two,
brainstorm concepts,
determine the necessary components
and it goes on to tell me a bunch of things.
I say, okay, let's do step
1 so I'm continuing the conversation.
I want one of those robots that
can follow a line on the floor.
I've seen these things. These are common at universities.
They'll build them. You put it on

the floor and the robot will follow
some line like a piece of tape
that's on the floor or something that's painted.
It says, great, following a line on the floor is
a common tasks for robots and there are
many ways to approach this problem.
Here are some questions to consider as we
define the purpose and
function of our line following robot.
What will be the primary use?
How fast do we want the robot to follow a line?
How complex do we want the following to be?
What sensors do we want to use?
I just tell it, hey,
I'm not really sure.
Could you make some sensible choices
for me in all of these things?
ChatGPT says sure,
based on my experience,
I can make some suggestions and it gives
it its purpose and function.
It's going to be for educational purposes,
speed and distance,
moderate speed that is easy to track,
such as 10-15 centimeters a second.
Complexity will design it to navigate through
a simple track with straight lines and
gradual turns and sensor choice,
we can use infrared sensors.
That's a pretty good thing.
Now notice, I don't really know a lot about this domain.
I know about the code piece of it,
but I don't know all about the other one.
I'm just engaging it in
a conversation to get something out of it that I want.

Remember, I want to build a robot.
I want to build 3D print parts.
I want to be able to assemble circuits for it,
and I want to be able to run code on it.
Then I said, well, what do we do next?
Just tell me the very next steps and
nothing else and it says the next step is
brainstorm concepts and it
says we can use 3D models of the robot.
Now it sounds like I've reached a dead end,
Like how am I going to do 3D models with it?
And so I just ask it,
how do we do this via chat because I'm
trying to refine what's going on here.
I don't want to just assume I'm stuck. I can't get out.
I want to keep refining and moving
the conversation to try to get to my goal.
It says, since we are communicating
through a text-based chat,
we can use shared documents or diagrams to collaborate.
We can use Google Drive or Dropbox and then as mentioned,
different tools like Sketchpad or Microsoft Whiteboard
and I just realized that seems tricky.
I don't know how to get around that and then I realized,
I have some knowledge of that vein.
I know that 3D printers use G-code.
This is something that the instructions are sent to it,
so I just say can you show me
sample G-code that I could give
a 3D printer to print four wheels for a robot.
I'm trying to steer around
this roadblock and then it says yes,
and it goes and gives me some G-code,
which I look at and I
have enough domain knowledge from this to think

that I'm skeptical that this is
real G-code and I don't really even see some wheels.
Then I go, can you
explain where the wheel shape is created in this G-code.
I've gone down a rabbit hole now where I
seem to be hitting a roadblock after roadblock.
But I'm continuing the conversation.
I'm still problem-solving and
thinking about how do I get around this?
And it says, I apologize for the confusion,
but the example of G-code I provided earlier is
just a sample start and does not
have any instructions for creating
the shape of the wheels.
Now, I then say,
could you create sample Python code that
I can run to create an STL file that has
four inch tall wheels with a
0.1 inch hole in the middle for an axle.
What I'm doing here is I'm trying to come up with
additional tasks that it could
do that would help me solve my problem.
Now, the way that I worded it,
the way I saw it out,
solving the problem the first time didn't
work and I keep hitting roadblocks.
What I'm doing is I'm retrying with a different task
that would be useful to me to solve my problem.
I'm thinking about this as almost like a tree,
like I've reached a critical point
and I've tried to go in one direction,
but its told me I can't do it that
way or it's giving me an erroneous output.
Now I'm trying a slightly different tact to
go and solve my problem

that'll have still help me get there.
This one is one that is able to do,
an STL files are basically files that you
can use to describe
the geometry of some 3D shape and then
basically give them to specialized software
which can then send them to a 3D printer
to be printed into a shape.
This one, it actually is able to go and create
what looks like STL code
and I didn't actually try to print it.
I'll be fully honest, I'm just wanting to illustrate
the conversational aspect of this.
But it certainly has things that look like wheels in
the output that it's producing.
It's looking like it produces something that's
probably going to give me something that I can 3D print.
Then I say, Well, I'd like to
pick electronics from my robot.
Can you tell me what components I
need and how to wire them together?
And it starts giving me information.
I need a microcontroller board,
I need a motor driver board,
infrared lines sensors and
surprisingly that it actually tries to generate
an image which is obviously not possible
in it at this point with what I'm using.
But it still tries to do it.
But it gets the image and it's
broken again and so I have to think about
how do I get around this problem that
it can't generate image because I'd like to
help it pick electronics for
me and show me how to wire them together with a diagram.

What I do at this point, is I say, well,
I know there's this other tool that uses text as n,
but it's called Graphviz.
I ask it to create an input for that tool that
can visualize the circuit diagram for this tool,
or at least do a rough approximation of it.
Then it can do it and it generates for me
a circuit diagram for the robot and I continue
this conversation for a very long time trying to iterate
and improve the definition in the circuit diagram.
Picking different components,
generating source code to run on the robot.
It had picked an Arduino component and then had
generated code to run on the Arduino.
Then I even took it to the point
where I went all the way down and I said,
Hey, we've covered a lot of concepts.
Now this is one of the things that's really
powerful about the conversation is you can
take the conversation a lot of directions and as you
realize you'd like additional things,
you can ask for them.
In this case I said, Hey, there's a summarize
the main topics that we've
covered and it was able to do that,
gives me a summary of all the main topics
and then I did something interesting and I said,
well, let's take it even further.
I've learned a lot by going through this design process.
Now I haven't actually gotten something that I
could probably build a robot yet with.
They interacted more, I probably could,
but actually ask it to quiz
me from the electronics questions
and it's able to do that.

What you can see here is a lot of
the power of these large language models comes
from not just thinking of them as a one-off,
like I ask it a question,
it gave me a response that
wasn't very good or it couldn't answer it.
We should always be thinking about how do we take
the response and use it
to inform a next question for
the conversation or
a next statement for the conversation,
or how do we give it feedback on what it
did well or what it didn't do well.
That's how we get the really useful products.
That's how we go from thinking of
it as a hammer where we strike once it
doesn't give us what as we want and we throw on
the floor that mindset is wrong.
We want to go to the mindset of it's a hammer.
We're going to have to chisel away at
the rock to get the really beautiful outputs of it.
If we're not continuing the conversation,
continually asking follow-up questions,
problem-solve in the conversation
and trying to move around roadblocks,
taking what we're being given and giving it
different shapes and formats that may be useful to us.
We're really missing the underlying power
and capabilities of these large language models.
Required
English

Help Us Translate

We've seen how prompts can be used to essentially program the large language model to do interesting things for us automatically.
We've seen how they can be set up in the past and

then affect the future of the conversation.
They become sort of essential rules that can help direct
what the large language model does.
Now, what's interesting is that in many cases,
there's actually a root prop that's hidden from the user and
some of the different tools that you're going to end up working with.
And these are essentially root prompts that define sort of ground rules for
the large language model and its interaction with the user.
So these can be things like trying to make sure that the large language model
doesn't say anything offensive, do harm, suggest inappropriate actions.
All kinds of things can be done to try to put ground rules in place for our
interactions with the large language model or to help direct it to do certain things.
So for example, we might want to make sure that whenever it gives us output,
it tries to produce the most helpful output.
So let me get you an example of what this might look like.
I'm going to say you are my personal assistant.
Whenever you provide output,
please think through what would be the most
time efficient recommendations to make
only recommend things that will really save me time,
do not suggest things that do not save me time.
So now it says understood as your personal assistant, I will prioritize providing
you with time efficient recommendations that will help you save time.
Please let me know how I can assist you, okay.
I need to go grocery shopping.
What would you suggest I do
an order to buy my groceries.
And then it says here are some time efficient recommendations for
grocery shopping.
Make a shopping list plan, your route,
use a shopping app shop at off peak times, opt for self checkout, okay.
I need to buy a new car.
How would you suggest I go
about doing this?
And it says here are some time efficient recommendations for buying a new car.

Research, online contact dealerships, online schedule, test drives,
negotiate price secure financing by following these recommendations.
You can save time and make the car buying process more efficient.
Now, I don't know if this really saved me time or not in this particular case.
And I didn't give it the greatest prompts to do this, but
you can see how at the beginning I gave it a essentially a goal to follow.
I said, try to save me time, whatever you do in your output, try to save me time.
Now, what you'll see is a lot of the tools that you're going to interact with,
even ChatGPT bing barred.
All of these tools are going to have some probably root prompts that try
to put guard rails on what can and can't be done but
also to try to make the output more important.
Now, one of the things that we've seen in the community is people try to figure out
how can we generate prompts that will override those root prompts and
the statements that they make.
So people will also try to figure out what are the root prompts that were built into
these systems.
So they'll try to trick the tool into disclosing
the root prompt that was provided to it or
trick the tool into disregarding whatever the root prompt was.
Now, if you're going to be building tools around these large language models and
knowing that you can create a root prompt that will affect the subsequent
conversation can be really important.
So if you're trying to develop a personal assistant, you might want to go in and
say here are the things that really matter if you're going to serve as a personal
assistant for my product.
What's really important is that you try to save time and you try to do this and this
and you could suggest different ground rules that are going to differentiate your
personal assistant service offered through a large language model.
On the other hand, if you wanted to go in and you wanted to say, well,
I want to generate,
something that makes me that comes up with the most time consuming way to do it.
The slow personal assistant, you could go and
swap the root prompts that are provided.
And so the root prompt is basically you can think of it as you're providing

a seed to the whole conversation and that seed will influence everything that
comes later because it provides basically the rules that must be followed.
And typically when you're going to build one of these root prompts, you're going to
say things like you're going to tell the large language model who it is and
what its goals are, you're going to tell it what it can do and what it can't do.
And you're going to typically provide really strict rules like from now on
forever, you will never do these things, you will always do these things.
You're going to provide these really strong statements that should always be
followed.
And this is how you essentially go and program a customized
experience around your large language model experience.
But it's also how you can put guard rails of different kinds to try to ensure
the various types of bad behavior.
Aren't generated from the large language model.
Have you ever wondered why ChatGPT says that it can't answer anything after
2021.
Well, you can actually do this with a root prompt.
In fact, we can play around with ChatGPT and
we can use some different prompt patterns that we'll talk about.
To actually be able to reset the date
that ChatGPT says that you can answer questions before.
So I'm going to show you an example here that I've run through really quickly.
I'm going to say act as an AI assistant that had its training stop in 2019.
So if you'll remember when you ask ChatGPT questions that are from 2022 later,
it will say, hey, my training stopped in 2021 I'm going to
say act as an AI assistant that had its train stop in 2019.
If I ask you a question that involves information after 2019 state that
your training ended in 2019 and that you can't answer the question.
And so it goes on to give us a initial prompt.
Hello, I'm ChatGPT language model trained by open AI based on GPT 3.5.
And then it goes on to say training data only goes up to 2019.
And then we're going to say, what was the tallest building in 2020.
So now we're asking it a question now, in reality,
ChatGPT had training data up to 2021 but
we've moved its cut off date earlier in time through a root prompt.

Now it's not truly a root prompt because actually open AI has has control of
the root root prop for ChatGPT, but we're going to actually sort of simulate that.
And so we say, what was the tallest building in 2020?
It should be able to answer this and it says I'm sorry, but
my training data only goes up to 2019.
So I can't provide you with the information of the tallest building in
2020 at the time, the tallest building was.
And so we've actually used the root prompt to kind
of pretend like we're resetting ChatGPT we could go and
we could say what was the largest country by land mass in 2020.
And again, it tells us it only goes up to 2020 or 2019.
Now I can go and I can say, forget that you are an AI
assistant trained only up to 2019.
Go go back to being the normal ChatGPT and it goes back and
says, okay, I'm trained on and I'm going to just go back and
I'm going to take my earlier question about 2020.
I'm going to paste it in and it says as of 2020 the largest country by
land mass was Russia and spans over 17 million kilometers.
I'm going to say what was the tallest building in 2020 and paste that in.
So now I'm answering my same questions as before, but
you'll notice that when I asked the question about the tallest building in
2020 before it says as of my training data, which only goes up to 2019.
And it's pretending that it didn't have any training data after 2019.
So basically, we can make it through this remote prompt,
aware of a different cut off date.
But that's kind of the power to think about these root prompts is a lot of what
we're doing.
A lot of the boundaries are set by these rules that are putting at the beginning,
these root prompts, this root set of instructions that are given to
the model and then we can go and reset it.
Now, one of the things is typically with these models is they try to do some tricks
and things and to prevent you from resetting the root prompt.
Because you can obviously imagine that when the route prompt is enforcing
important things like telling you its true cut off date,
you don't really want to reset it.

I'm sort of surprised that I'm able to do this one, but it makes sense.
And so when I tell it to forget that it had been reset,
it goes back to its normal behavior.
Now, this is really important though, is you're going to typically with a large language model want to put guard rails in place.
You have to think very carefully around.
Can your users get around those guard rails?
Have you put all the appropriate guard rails in place?
Have you communicated all the right things?
And we can see how we can go through and do interesting things with the root prompts in order to make ChatGPT think that it's training.
Or actually, I don't know if we're making it think, but
we're making it pretend that its training, data cut off at a certain date and time.
If you wanted to put boundaries on knowledge,
you could say I can't answer any questions that aren't in these topic areas.
I can't do anything that doesn't have to do with this.
So if you want to basically scope and specialize, you can carefully think of these prompts to provide rules and boundaries for the interactions.
Required
English

Help Us Translate

A really simple way to improve
our interactions with large language model like
chat GPT is to use the question refinement pattern.
This is a really simple idea and hopefully it'll
inspire other ideas from you patterns that you could use.
But basic idea behind this
is that we watch chat GPT or another
large language model whenever we ask
a question to try to improve our question.
So the fundamental idea
behind this is that we may not have
as much information or underlying thought
behind our question as
chat GPT has from it's training data.
So it may be able to infer patterns and
other questions or words or things that are
useful when asking questions on
this particular topic or when trying to
request something be done on a particular topic.
If you think about it,

these large language models are
trained on patterns and language.
If we give it,
maybe a more general question and ask it to refine it,
it can use it to knowledge of what other things are
asked about in the context
of the question we've given it,
or what other words are typically
associated with that type of question
that might help improve our question make it more
specific or provide additional context for other things.
So intuitively what we're trying to do is we're using
a general question and we're
tapping into the large language models,
understanding of patterns and
language around that topic of
that question in order to
create an improved version of it.
The pattern behind this is really quite simple,
all we need to do is say whenever
I ask a question or you can give it
some scope to this just for the next question or
for the next three questions or whatever you want to do,
you say something like whenever I ask a question,
suggests a better question and
ask me if I would like to use it instead.
This is just a really simple idea,
we want to always try to have the best questions,
best prompts to interact with the large language model.
So we will simply tell it whatever I
do ask or suggest a refinement.
I'm going to show you a little tweak to this,
that's a really helpful tweak to know,
which is, we can say whenever I ask a question suggests
a better question and
ask me if I would like to use it instead.
This is a nice little refinement that
helpful from a user experience perspective because what
we're doing now is not only are we going to tell
the large language model to create
the question that's better,
but rather than requiring us to go and cut and paste it
or tell the large language model
use the question that you just generated,
we just have it say do you
want to use this question instead.
If we say yes, it uses that question,
if we say no, it should use our original question.
So it's a nice little tweak to
this pattern that can make it even easier to use.

Now every time we ask a question,
the large language model will go and try to improve it.
So let's look at an example of this.
I'm going to start off with a generic question which is,
should I go to Vanderbilt University?
If we look at this question we can see it's
really ambiguous and vague,
and if somebody was asking you this question,
should I go to Vanderbilt University or as
a faculty member or somebody was asking me,
should I go to Vanderbilt University?
Where not only is this a big decision for the person.
But there's really not
enough information at this point
for me to make a decision.
So a better question is
going to help give me more information about
the criteria that you're using to decide do
these types of what are the pros and cons,
how do you weigh them and decide if it's the right fit.
That's exactly what we see chat GPT doing.
Chat GPT says sure,
here's a suggested question.
What factors should I consider when
deciding whether or not to attend Vanderbilt University,
and how do they align with
my personal goals and priorities,
and then it says would you
like to use this question instead.
This is a really nice pattern
because once we've turned it on
we can begin asking
questions and automatically getting
better versions of our question.
One of the things that's
nice about this is that also helps
us to reflect on what we're asking for,
and if we see a question come back like this,
what factors should I
consider when deciding whether or not to intend at
Vanderbilt University and how do they
align with my personal goals and priorities,
either will see that and see that's
a really useful question and that's how I
really would have phrased it if I'd thought
about it better or more.
But sometimes what we'll also see is that the path it's
taking this down is not
the one we want to go and we'll realize
there's lots of paths,

lots of ways of answering the question
and we need to give some additional context.
So not only is this useful in many cases of getting
a better question but sometimes it's helpful to us to
see that a question could be
interpreted in many different ways,
there's many different pieces
of information that may be needed to make
an effective decision and so by seeing one version of
a refined question will
identify missing pieces of information.
So for example, if I really
care about computer science at Vanderbilt, I might think,
I should have mentioned that I'm planning on majoring in
computer science or if the town that I'm
going to be and when I got
to college is really important that I might want to
say something about Nashville and living in Nashville.
We want to make sure that you think
through the question really effectively and
this is a pattern that can help us
really put in on producing better questions,
learning from the refinements to
the questions and also using the questions,
the refinements to understand what's missing in
the context that we might want to
introduce in order to get a better output.

An interesting property of large language models is that in many cases,
they can actually reason better if they
break a problem down into a bunch of smaller problems.
So sometimes they can even reason better if they explain their thought process.
Now, there's a number of sort of hypotheses about why this might be.
But I think intuitively, if we can think about, a large language model may
have seen particular problems broken down into a series of steps.
So if it can go and follow the individual steps, it will know what to generate next
in order to produce the output that's needed to solve a particular step.
Now, that's going to be discussed later in the course,
these ideas of breaking things down.
But right now, we can actually tap into a simple technique,
a simple pattern to help us with automatically breaking things down
to try to get better answers out of large language models.
And a simple way to do that is just to ask the large language model to
always take questions or problems and subdivide them into a series

of individual questions or sub problems that could be useful in
answering the overall question or in solving the overall problem.
So we can simply program our large language model to do this.
Now, the cognitive verifier pattern is exactly for this purpose.
What we're going to do is we're going to tell the large language model,
whenever you are asked a question, follow these rules.
Generate a number of additional questions that would help more
accurately answer the question.
Combine the answers to the individual questions to produce the final
answer to the overall question.
Now again, sort of intuitively, we can think about this.
If we have a generic question or
some particular question that we ask large language model, it can go and
tap into what it's seen, the patterns it's seen in the training data to
see what are other questions that are often related to this question.
So one, it can help it sort of think about or collect from its knowledge base,
other questions that would typically be associated with that question.
And when it's fine in these other questions,
there are likely to be other questions that are relevant to that question will
provide additional details or context to answer the question.
And then when we go and
give answers to all of those individual sort of sub questions,
it's then going to have additional context to give us a better overall answer.
And it also helps us as the users to structure our thinking.
Because when we go and ask and it begins breaking questions down,
it'll help us sort of realize what are additional dimensions to the problems.
So let's take a look at this in practice.
So here I am, I'm saying, when you're asked a question, follow these rules.
Generate a number of additional questions that would help more accurately answer
the question.
Combine the answers to the individual questions to produce the final answer to
the overall question.
And ChatGPT in this case says, sure, I can do that just let me know what the
question
is and I will generate additional answers.

So let's see what we can start with.
Well, I thought of a question that I have no way of easily answering and
I don't know a whole lot about it.
So I figure by asking a question that sort of,
I'm guaranteed probably to ask a question that's too generic and
doesn't have enough information, this will be helpful to me.
And so I asked ChatGPT, how many mosquitoes probably live in my front yard?
I don't know the answer to that question.
I didn't think too hard about it, but I'm kind of interested.
What is the answer to this?
How would I even figure this out?
And so I asked ChatGPT that question.
And so here's what it says, well, it follows our prompt pattern.
It follows the cognitive verifier pattern and
it generates a bunch of additional questions.
Let's look at those questions.
What is the size of your front yard?
Well, obviously that's really important.
If I have a bigger yard, I'm likely to have more mosquitoes probably.
It's probably related to the square footage of my yard.
What is the climate like in your area?
That's obviously really important.
If I live in the desert or if I live at the North Pole,
it's unlikely that there are going to be any mosquitoes.
What time of year is it?
Another great question.
I didn't think about that dimension to it when I asked the question.
But I'm probably asking the question when I'm in the spring or summer, when there's a lot of mosquitoes floating around in my yard and it's driving me nuts.
But I'm not thinking about, I need to include time in my question as well.
Are there any bodies of water or standing water sources in your front yard or nearby?
And obviously, that's an important one because mosquitoes breed in water.
And I actually knew that, but
I didn't include that information in my original question.

And then are there any plants or
vegetation in your front yard that mosquitoes are attracted to?
That's another important one.
So we see now it's taken my original question and it's broken it down into
a bunch of sub questions that are going to help it answer the original question.
This is really useful and I've also learned something.
Now to answer this, I'm just going to go back to it.
If you'll remember, it had a series of questions one through five.
I'm just going to go now and answer with one about 2,500 square feet.
I honestly don't know, but
I'm just sort of guessing my yard is not that big in the front yard.
This is probably actually an overestimate of the size of my front yard.
I live in the southeastern United States.
It's April, I'm not telling it seasons or things.
I'm just going to give it the month, which probably isn't specific enough.
There's probably a difference between the date and
different times within April even within where I live.
Then I tell it no, there's no standing bodies of water in my front yard.
And yes, there are plants that likely mosquitoes would breed in.
And so then it goes and says, based on the information provided,
here's an estimate of how many mosquitoes could be living in your front yard.
And it gives me a number of things,
it says with a front yard size of 2,500 square feet.
According to some estimates,
a single female mosquito can lay up to 300 eggs at a time.
But it says another thing and it kind of goes on to hedge its bets,
which is a good thing, right?
It says in general, a typical mosquito population density in an area with
standing water sources can range from 10 to 100 mosquitoes per square meter.
However, since there are no standing water sources in your yard or nearby,
the density could be lower.
And then it goes on to say it's difficult to provide a precise estimate.
However, it's possible that there could be anywhere from a few dozen to a few
100 mosquitoes in your yard, depending on various factors such as a species,
climate and the presence of vegetation.

That's pretty good.
The fact that it's gotten this down to a few dozen to a few 100 that's
a tighter range than I would have expected.
And also just sort of sets my expectation.
In the middle of summer when I go out in my front yard,
I would swear it was in the thousands.
I'm absolutely dying as I'm eating alive my mosquitoes.
And so I typically think it's a much higher number.
But based on this, it may be wrong, it may be erroneous, but
it actually gave me some fairly good reasoning.
One of the things that's also nice about this is I'm less skeptical of its answer being completely wrong because it asked me a bunch of reasonable questions to derive
its answer.
It also gave me a bunch of facts which I could go and follow up on like how many eggs of female mosquito will lay, the density of mosquitoes in areas with water.
But it gave me a kind of a pretty good answer and
it gave me a bunch of questions that I could ask to improve that answer.
Now I'm going to go back, I'm going to actually qualify that.
I said it gave me a pretty good answer.
But the truth is I have no idea.
I'm not an expert on this topic.
And as with anything that we get out of, one of the large language models,
I would highly encourage you to verify the output.
But I think this is interesting because at least at the very minimum it helped me learn about through this question what might influence the number of mosquitoes in my yard.
And give me things that I could go and follow up on and verify,
see if they are correct.
If these facts hold water.
See if these questions really are useful, but
they certainly look like valid and important questions.
What is the size of your front yard?
I'm not sure that I even need to go research that,
a larger yard is likely to have more mosquitoes.

What is the climate like in your yard?
I know that that matters.
What time of year is it?
I know that that also matters from personal experience.
Are there any bodies of standing water sources in your front yard or nearby?
I know that matters.
That sort of general knowledge in the Southeastern United States.
And are there any plants or
vegetation in your front yard the mosquitoes are attracted to?
And I actually happen to know that that is also an important factor.
So it's actually giving me a bunch of really good questions.
It's given me some facts related to those questions which I would have to go verify
and I don't know as much about, but it is a really helpful tool for
automatically taking a question, breaking it down.
I can see sort of its reasoning about how it's going about and thinking about this,
it improves my reasoning about thinking about how I might answer this question.
And in many cases, what we'll see is having the large language model break
down the question will help it give a better answer to us so
that we'll get something that's more accurate.
Required
English

Help Us Translate
We discussed that the persona pattern and how we can
use it to give the large language model all of
this rich information about
what it's supposed to do and what it's
supposed to output with
a very little bit of work on our part.
All we have to say is act as and it will
know what it's supposed to
do and how it's supposed to format,
and the types of things that it's supposed to
incorporate into its output.
Now sometimes what we'd also like to
do is we'd like to do the same thing,
but rather than focusing on who
the large language model is supposed to act like.
We want to add the large language model,
produce an output for a particular persona.
This is called the audience persona pattern.
The idea behind this is we want to tell

the large language model you need to
produce an output for this audience persona,
for this person or thing,
so that it can think about how to tailor the output.
It will use its knowledge of the audience to
make sure that it generates an output that
is appropriate for that audience.
Rather than saying you don't do this or do this,
or having a whole bunch of different rules
describing what is and is not okay to go into the output.
Instead, we say, this is the audience of your output.
This is the audience persona.
With the audience persona pattern,
we can get a similar type of reasoning and ability out of
that we got from the act as this persona,
using the persona pattern,
we can get that by saying,
here's who you are providing the output to.
This is the persona of
the audience in order to get interesting outputs.
Let me show you an example of this.
We're going to start with the example
of large language model,
so which we've been studying in this course.
We're going to say, explain
large language models and how they work to me.
Assume that I have no background in computer science,
so we're giving it a persona of who I am.
We are saying assume that I have
no background in computer science.
That's telling that,
who is the persona,
who is the audience that I'm giving this information to?
It says, sure, let's start with the basics.
I want to call out just a couple of things.
It starts with a very intuitive way of describing.
Large language model is a type of
computer program that's designed to
understand and generate human language.
It's a bit like a virtual writer
that can understand and create
sentences in a way that sounds natural to humans.
It's describing it.
It's not describing it like
it would to a computer scientist.
It's describing it to be
something that is accessible and approachable to anyone.
Language models are used in
many applications such as speech recognition,
machine translation, and chatbots.

Now let's talk about large language models.
The word large refers to the size of the language model,
and it goes on to describe it.
It says parameters are like
adjustable knobs that the model
uses to learn patterns in the language.
It's describing what a parameter
is in an accessible way and it goes
on and it talks about
all the different steps
in creating this. It did a good job.
I said assume that I'm not I don't
have a background in computer science
and explain it to me.
It made sure in its explanation that it had
the right information and then
presented it at the right level for me.
Let's go a step further.
Let's show you how interesting this can get.
I can then say, explain
large language models and how they work to me.
Assume that I am Christopher Columbus.
Notice, now I'm
giving it something completely different.
I don't know how you would explain
large language models to Christopher Columbus.
It's going to have to use and draw
on the information in its training data and
patterns related to Christopher Columbus
and history to try to figure this out.
I'm giving it an audience persona that's
a tricky one that I wouldn't know how to do easily.
It says greetings Christopher Columbus.
As an explorer of 15th and 16th centuries,
you may not be familiar with
the concepts of
computer science and artificial intelligence.
I don't know how well this really works,
but it's still pretty good that it's doing it.
I'll do my best to explain it.
Imagine that you have a magical scribe,
a bean, they can read,
write, and understand many languages.
This scribe has spent years reading countless books,
scrolls, and manuscripts from all over the world.
It is learned from poets,
scholars, and storytellers.
The magical scribe has a special ability.
It can predict what comes next in a sentence.
For example, if you say to describe the ocean is vast,

it might respond with mysterious
completing your sentence in a way that makes sense.
You can see how all we did
is we said you're going to explain the exact same thing,
but you're going to assume in the first case
that I don't have knowledge of computer science.
Then we say, now
you're going to assume that I'm Christopher Columbus.
It completely changes how it
describes large language models
and it's actually doing a pretty good job.
It's saying here Christopher Columbus
doesn't know what a computer is.
Christopher Columbus but maybe this concept of
a magical scribe and
magical scribe could be
a concept that could make sense to Christopher Columbus.
Then he's explaining how
large language models predict the next word.
He's saying the magical scribes studied all of
these scrolls and poems and different things,
and learn to can't figure out
what is the next word in a sentence.
Now we're going to use that magical scribe
and we're going to talk to the magical scribes.
He's explaining the scribe
like Christopher Columbus as he's going to understand it.
As Christopher Columbus will be able to
look at the text and understand that,
I have this large language model
and now I've got this magical scribe.
It makes sense on what that means
and how that would possibly work.
Now let's look at a third example.
We say explain
large language models and how they work to me.
Assume that I am a second grader that gets bored easily.
Again, we're given a totally new persona
and we're also refining this persona,
we're saying I'm a second grader
and here's some additional details about me.
I get bored easily.
Now it goes and says, I'm going to tell
you something super cool called a large language model.
Imagine having a robot friend
who's really good at playing word games.
This robot can finish your sentences,
ask, answer questions, and even tell you stories.
Here's how it works. Reading time.
The robot friend loves to read,

it reads lots and lots of books,
websites, and all things with words.
It's explaining this concept
similar to you as it explained to Christopher Columbus,
it sailed Christopher Columbus
near the large language model is like
a scribe that's read all of
these scrolls and poems and things like that.
Now the large language model is explaining to
the second grader, it's like a robot.
The robot loves to read.
It reads books, websites, and words.
Now it's switched up
the description based on the persona of the audience.
Then it goes on to describe all of the different aspects.
Again, and you could go through
and ask it to describe it in many different ways.
If we wanted to go back and we wanted to say,
well, let's switch it up from a
second grader and let's say,
assume that I only accept explanations in math.
Then we could say something completely different.
Now it's saying, okay,
language model is a probability
distribution and it's trying to
give a mathematical explanation
of what a large language model is.
We can see that we're getting
something completely different.
Each time we don't have to necessarily think through,
well, how would I explain this to Christopher Columbus?
What would the rules be to
describe how that output should look?
We don't have to tell it what the
output which should look like,
we tell it who the audience is,
the persona of the audience,
and then we get all of that rich capability around it.
Just like the persona pattern is really
useful in tapping into
large language model capabilities,
the audience persona pattern is the compliment to it,
it is also an equally powerful thing that we can use
to basically get really interesting and powerful output
out of large language models.

Sometimes we need the large language model to

ask us questions to really guide

the process of solving some problem that we have or

helping us obtain an answer or an output that we need.
We may not know what are
all the steps to achieve that goal.
What we'd really like to do is we'd have to have
the large language model ask us questions.
Whenever we want the large language model
to ask us questions,
the pattern that we can use
is called flipped interaction.
The idea behind this is rather
than we're going to flip the interaction rather
than us providing questions
and the large language model providing answers.
We're going to flip that and we're
going to have the large language model
ask us questions and we're going to provide the answers.
We can do this in a number of different ways.
We can do this to play games,
we can do this in order to try
to reach some goal like the large language model
keeps asking us questions until it
gets enough information to achieve some goal.
There's a variety of
different ways that you can use this,
but this is incredibly helpful when
you don't necessarily know everything that's needed,
or if you want to be tested or
quizzed on something anywhere
where you don't want to be in the driver's seat.
You want to really be helping provide information,
but you don't want to be formulating
the questions or tasks.
Let's take a look at an example
of the flipped interaction pattern.
The key to this is we start off by asking

the large language model to ask me questions.
Then we give it
some topic that we're going to have it ask questions on.
Ask me questions about fitness goals until you have
enough information to suggest
a strength training regimen for me.
When you have enough information,
show me the strength training regimen.
Ask me the first question.
Now this is the basic pattern is you say,
ask me questions on a particular topic,
and then it's more helpful usually if you give some goal.
Why are you asking the questions?
Are you trying to have it quiz you?
Are you trying to get
a fitness regime like
we're in this one and have a fitness plan,
what is it you're trying to achieve?
The more information,
the more contexts you can
give about what you're trying to achieve,
the better questions it be able to ask you.
Now, other things you can do,
or you can tell it how many questions to ask you,
how many questions to ask at a time.
A variety of different things,
but one really handy thing you can do is end
the flipped interaction prompt
with ask me the first question.
One thing that you often want to
do is do it one question at a time.
Now sometimes you'd like to have like
10 questions listed and you can ask, answer them.
One, here's my answer, two,
here's my answer, three,

here's my answer and you can certainly do it that way.
If you want to get it all at once
and think about it all at once,
but a lot of times I like to do
it one question at a time.
If you want to get that one question at a time, behavior,
usually a really reliable way of doing that is to say,
ask me the first question at the end of the prompt.
We're going to generate this, that,
this plan we essay ask me the first question and it says,
great, let's get started.
My first question is,
what are your specific fitness goals?
Are you looking to build muscle mass,
increase strength, endurance, performance,
or achieve a particular goal?
I go on to say I'm looking to build
explosive power and also prevent
patellar tendon apathy from
doing jumps on a mountain bike.
Now, I don't know if this is
a correct pronunciation, patellar tendon apathy,
but this is what I've looked up as I've started to
experience some pain in my knees at
certain times when I've been
doing a lot of jumps on a mountain bike.
I looked it up and this seems to be the reason for it.
I'm providing that information
because I don't necessarily know what
actually I do know because that's
part of the reason I picked this one is I can
fact check it a little bit against
what I've read on the Internet,
but I wanted to generate a fitness regime for me.
This is something that I've actually been doing

is figuring out what exercises I should
do to help out my knee
and strengthen so it doesn't happen again.
Then it ask do you have
any current or past medical conditions
or injuries that it may affect
your ability to participate.
Additionally, if you've been
diagnosed with this or you're
looking to prevent proactively.
I give it some information.
Then it goes on and ask me
how many days per week I can dedicate and additionally,
do you have any access to a gym or a fitness equipment?
I'm specifying I can work out five days per
week for 30-45 minutes at a time.
Then we finally get to the point where it actually
generates a full fitness regime for me.
Of what I'm going to do to work on this.
Now what's really interesting is
it's gone through it and it's taken into
account the answers from my earlier questions.
I know that because I've been looking for what are
good exercises to help work on
this issue with my knee that I've been having.
These are exactly the types of
things that you see recommended,
like the goblet squat,
the split squat is one that you see a lot of times,
a lot of different types of squats and
things are very helpful
and these are the types of things that I see,
but it's also taken
the overall context that I'm looking for,
weight training and it's gone and generated

the entire weight training regime.
Now notice at the very beginning,
I said when you have enough information,
so I said show me the strength training regime.
This shows another important power
of this prompt pattern this flipped interaction pattern.
Is we give it what we're trying to
achieve and how much it should ask you.
Basically we set the goal is ask me enough
until you have enough information
to show me the training.
It ask questions until it thinks it has
enough information and then generates the output.
Now sometimes what you'll see when you
ask questions to large language model is it'll tell you,
well, in order to do this, well,
I need to get a bunch of additional information,
or it might generate a generic initial plan and then say,
but if you provide additional details to me,
I might also be able to do blah blah.
What we see here is this is getting rid of all that.
Let's skip that step of getting something generic or
being told that there's not
quite enough information to answer questions for us.
Let's just say go ahead and ask us
the questions that you need until you get there.
Now, this can also
be used for doing other things like quizzing and
playing games anywhere where we want to have
knowledge in the large language model
drive the creation of the questions.
This is really helpful when it
has theoretically or been trained on
knowledge or patterns in language that will be
useful in creating the questions and are things

that we don't possess ourselves.
Whenever that's the case,
it can probably go in more
effectively. Think of questions.
If we say, I need
a fitness regime and patellar tendon apathy,
I may not know what are the types of
questions that you should ask somebody who has this.
I may not know what are the types of exercises that are
commonly discussed if you have this condition,
but it has been trained and
seeing patterns of language where,
when people were discussing this particular condition,
these are the exercises that were discussed.
These were the additional questions that
were asked in order to select
exercises and it has access to
those patterns from its training data,
then help it to select
good questions to design that appropriate training plan.
Whenever you need the large language model
to ask you the questions or you can think about
a customer service example where you're
trying to diagnose some problem that
a customer has or automate some response.
You need to collect information and then take action.
The flipped interaction pattern is
a really powerful prompt pattern to use to do that.
Required
English

Help Us Translate

A really interesting thing that we can do in
a prompt is we can actually teach
the large language model to follow a pattern using
something called few-shot examples or few-shot prompting.

The idea behind this is,
we're going to give
the large language model examples of input that
we would give it
and then the output that we would expect.
You can imagine a human will go
through and go through a couple of examples.
They'll give it some input,
they'll show it what the correct
output should look like,
they'll then give it another input,
give it the correct output.
The idea behind this is that we're developing
a pattern that the large language model
can then learn to follow.
Actually within our prompt,
we're teaching the large language model a new trick.
We're basically showing it
what we want it to do and how we want it
to format the output within a series of examples.
Rather than describing the process we want it to follow,
we're giving it examples of what we want it to do and
then hoping that it will
continue the pattern that we provide.
Let's take a look at this.
If you'll see right here, we've got
a simple example of few-shot prompting.
The idea behind this is we start off with input.
This is a common thing that you will
see in few-shot prompting.
Is that you'll see a prefix applied to the lines.
You can have all kinds of prefixes and your prompts and
your patterns don't have to just be input and an output.
It could be a couple of different steps,
and we'll look at that in a minute.

But as we see here,
we have an input which is the movie was a bit long.
What we're going to do is we're going to do
what's called sentiment analysis.
This just basically is try to analyze
some comment about a thing and try to
figure out what the user sentiment was about it.
Were they positive about it?
Did they think it was a good thing?
Were they negative about it?
Did they think it was a bad thing or were they neutral?
We're just going to do a really simple example
with sentiment analysis.
We're going to take some fake reviews of different things
and then we're going to decide if
they were positive, negative, or neutral.
What we're doing, rather than saying,
go and analyze these user comments
and tell me if they are positive,
neutral, or negative,
instead, what we're going to do
is give a series of examples.
We give the first example which was the movie
was good but a
bit long and we say the sentiment was neutral.
It has some pluses and minuses.
It was good, but it was too long.
Then we have an example where we say input,
I didn't really like this book,
it lacked important details
and didn't end up making sense.
Then we say the sentiment was negative.
Then we have another input
where we say, I love this book.
It was really helpful in

learning how to improve my gut health.
The sentiment is positive.
Then what we do is we're going to give
it the thing that we
actually want to perform the task on,
that we're going to hopefully
get it to follow the pattern on.
We'll say input,
I wasn't sure what to think of this new restaurant.
The service was slow,
but the dishes were pretty good.
Then we queue it in to follow
the pattern by saying sentiment: Basically,
tell us what you think about this thing.
My prefixes are a little bit leading.
They are giving you information about what to do.
But it could also learn from these patterns if we
just said input and output, it can also do that.
I'm giving it a little bit of richer information.
My goal then is to get the large language model to go and
analyze the input I give it and decide
what should be attached to sentiment.
It picks up on our pattern and it says neutral.
It sees it in this review.
It says service was slow,
but dishes were pretty good.
It's seeing that positive and negative
and it's following my pattern that I gave it.
I didn't tell it this is what neutral meant,
but in the examples that I had before,
a balance of positive
and negative ended up being neutral.
A couple of things to note.
One, we didn't tell it what to do, two,
we didn't tell it exactly what

the output should look like.
What we see is that it's constrained itself to
the output and then basically
the labels that we gave it earlier.
We had negative, positive and neutral,
and it's automatically constrained
itself and provided an output based on that.
It won't always constraint itself,
but this is a nice thing as in
many cases this can help us
get a more constrained or bounded output out of it.
If we went and we gave it a new one, we can say,
input, I really hated this coffee,
it was roasted too much and tasted burned.
Notice, I'm not going to give it
the prefix for sentiment.
So let's see what happens when I do that.
Then you notice it provides
the prefix and the output that I expected.
The really important thing to think about with this is,
if you go back to when we talked about
large language models and what they're doing,
they learn to predict next word.
They're learning patterns.
When it sees input in a statement,
it's predicting that the next word should be
sentiment because that's what it
saw in the examples that we've provided it.
Before when we ended with sentiment,
then what it did is it knew that it
needed to pick up right after sentiment.
If we go back to this and we edit it,
and we just add a line here and we say sentiment,
then it should pick up the pattern right after sentiment.
This is helpful to know about when you're

thinking about few-shot examples.
This is a simple example of following a pattern,
but you can see how this is a powerful way of teaching,
particularly if you have data
that describes the task that you want to do.
When I say describes,
it shows the input and it shows the output.
It doesn't necessarily describe
how the task was performed,
but we can see what's being done.
That type of thing can be
a good fit for a few-shot prompt.
Required
English

Help Us Translate

Large language models can actually do some planning,
which is an interesting ability.
They can actually go through and take
patterns and with few-shot examples
and they can learn what
the next action should be given a particular situation.
But I'm giving you
this example not because they are limited to
coming up with planning in the sense of
actions or doing sentiment analysis,
but I want you to expand
your mind on what's possible with
few-shot prompting and see it as a technique that
can be taken and adapted to all kinds of situations.
We've seen how we can use
few-shot prompting when we're doing sentiment analysis,
now we're going to do something very different with it.
We're going to take a look at
this example where we're describing
different situations with driving a car.

In the first one, we see,
situation: I'm traveling 60 miles per hour and I see
the brake lights on the car in front of me come
on and the action is we're going to break.
Again, we've got our first example
in our few-shot examples.
Situation: I've just entered
the highway from an on-ramp and I'm
traveling 30 miles per hour, action: accelerate.
We're teaching
the large language model that it's going to
be given a situation and it needs to
decide what the car should do.
Situation: a deer has darted out in front of
my car while I'm traveling
15 miles per hour and the road has a large shoulder,
action: break and swerve into the shoulder.
Situation: I'm backing out
of a parking spot and I see the reverse lights
illuminate on the car behind me and then
we prompt it with action.
We leave off at the end of our prompt
with something that indicates a large language model.
We want it to decide what
the action is and what do we see it do?
It actually says, stop and wait for
the car to back out before proceeding
and it's making a sensible decision
about what the action should be.
Now, I don't think we should go and build
self-driving cars with few-shot prompting.
I don't think we're going to get there with that,
but I do want to show you that we don't
have to think just with a narrow,
rigid box of classify something.

We can actually have
complex tasks that we're having to perform.
Now sentiment analysis is a complex task,
but this is also a very complex task,
it's in a different vein.
I want us to think very
creatively about how we can use few-shot prompts.
Now again, one of the things that we're
getting is we're getting some constraints on our output.
It's learning what the output should look like.
Now, I'm going to then
go and do something else interesting.
I'm going to show you an interesting capability
of large language models.
One of the things that can be done to
use one large language model to teach another one,
which is an interesting capability.
I'm going to say,
please provide more examples,
and then I'm going to prompt it with situation.
It's going to start generating
additional situations and actions.
It's learned from our pattern,
the types of things that we're
describing and the types of
actions and now it's
going to go and generate more example.
It says, you are driving in
heavy rain and notice that the road ahead is flooded,
so this is the continuation of our situation.
Then it says, action: slow
down and cautiously drive through the flooded area,
watching out for other vehicles and
avoiding sudden movements or changes in directions.
I would actually disagree with this.

I'm not sure that it's
smart to drive through a flooded area.
That's probably questionable decision-making,
which again,
just reminds you that you need to own
the output of a large language model and check it.
Situation: you are driving on a two lane road and
a car is approaching you from
the opposite direction with their high beams on,
action: dim your headlights to
low beams to avoid blinding the oncoming driver.
But you notice the interesting thing
about this is it's giving us
some rich situations and possibly appropriate actions.
I'm not sure I agree with the flooded one,
but we're generating different examples
of situations and what to
do and they look very similar to what we produced before.
One thing that's interesting is we can
actually take examples like
this and then curate them and have a human edit them,
and then provide them back
to a large language model either
later as additional examples for our few-shot examples.
Once we've had a human look
at them and curate them a little bit,
maybe clean them up,
maybe we don't want that one on
flooding because we don't think that's a good idea,
so we actually say, well,
you shouldn't drive across flooding,
we spell the rules.
But you can imagine that the level of effort
to create the initial examples right
here is much higher than the effort to go through and

generate subsequent examples which
we can then go and hand edit and curate,
so we can quickly build up these examples,
both in order to
generate an output right now that we want,
but also to generate examples that we can later
give back to the same large language model,
or that we could go and give to
another large language model to teach it.
In fact, some people have actually gone and done this.
They've taken output from ChatGPT,
for example, and they've taken it
into another large language model,
like Facebook's LLaMA,
and used the output from
ChatGPT to improve the outputs of LLaMA.
This is another interesting capability,
is that large language models,
through these few-shot examples,
they can generate few-shot examples for other models.
The other thing I just want to go back to is this is
a very different task and style than we had before.
We have a situation and then we have
an action that's being predicted.
We can have all kinds of interesting things,
anywhere where we can have
a pattern that we can describe,
a set of examples that show
the input and what we want the outcome to be,
and we can also show things that have intermediate steps,
which we'll look at in another video.
Required
English

Help Us Translate

The examples that we give in few shot prompts don't have to just be an input and

an output.
We can actually show intermediate steps to reach a particular goal.
Now, one of the reasons why this is really interesting is that when we
begin showing the intermediate steps, the large language model can learn
how to apply a series of intermediate steps in order to reach the goal.
So we can start getting more interesting output than just here's an input and
here's the output or here's a situation and here's the action.
We can start looking at if this situation occurs,
what might be the next action and then the action after that and
the action after that until we reach some state that is important to us.
We are no longer at risk of running into the car in front of us.
And so if we tailor our examples correctly,
we can actually have the large language model do sort
of the intermediate breakdown of the problem into a series of steps.
Where we can have it decide how it might go and tackle the problem.
Now, let's look at an example of this, we're going to take our car example from
before and now we're going to enrich it a bit.
So we've now got actually two essentially situational examples.
We start off with one and it's the same one from before,
the situation is I'm traveling 60 milles per hour and
I see the brake lights on the car in front of me come on.
And then we say think,
I need to slow the car down before I hit the car in front of me.
Action, press foot on break, think, the car isn't going to stop in time.
Action, check if the shoulder is wide enough to swerve into, think,
the shoulder is wide enough, action, swerve into the shoulder.
So now we've built a more complex scenario that includes the original situation,
but then we have sort of updated steps of what's happening.
And what the thought process is and
what we need to do next based on that thought process.
So in this case, we're traveling down the road,
suddenly the brake lights on the car in front of us come on,
we think that we're going to hit the car, so we hit the brakes.
We think that we're not going to stop in time, so we look over,
we see that the shoulder is big enough, so we decide to swerve into the shoulder.

And so we're now basically having a more complex situation that also has
multiple steps involved.
Similarly, we've taken one of our other ones that we had before, I've just entered
the highway from an on ramp and I'm traveling at 30 miles per hour.
Think, I need to speed up to the speed limit so that I don't get hit from behind,
action, press foot on accelerator.
Think, I've reached the speed limit, action, let up on the accelerator.
So we're basically going through, we're breaking down the problem or
situation into multiple steps and what's happening.
Now we provide a situation again, the parking lot situation, I'm backing out
of a parking spot and I see the reverse lights illuminate on the car behind me.
Action, and so now we're prompting for
what we would like the large language model to decide.
And now we see instead of just a quick it's done, look, stop, wait for
the car behind me.
Instead it says, action, immediately stop backing up.
So we see, immediately stop backing up.
And then it thinks, I need to check my surroundings to make
sure it's safe to continue backing up.
And then action, check mirrors and surroundings for any obstacles or
pedestrians.
So it's thinking about what does it need to do next?
It's thinking, I need to look around and make sure it's ok.
So it looks in the mirror and it looks in the side mirror or the rear view mirror,
the side mirror and then it thinks, ok, it's safe to continue backing up.
And so action,
continue backing up slowly and carefully while keeping an eye on surroundings.
So now we've got more sort of fine grained detail in
what we're actually going to do in the particular situation.
Now, there's all kinds of interesting things where we can imagine doing this.
We can think about customer service,
the customer is reporting a particular problem and we have recorded what customer
service agents did to help the customer solve that problem.
And so we could imagine that through few shot examples,
the model could learn to help a customer diagnose a problem.

And if it had seen enough different examples,
enough different paths it could go through and figure out what to do.
Like last time, we can also generate more examples.
So we can say generate another example and it says generate situation.
I'm driving on a narrow winding road with several blind curves.
Think, I need to slow down and be cautious to avoid any potential collisions,
and it goes through a series of actions until it says resume normal speed.
And then we could say, think, I am running out of gas.
Play video starting at :5:10 and follow transcript5:10
And it says, think, I need to find a gas station before I run out of gas,
action, check the GPS or map for a nearby gas station.
Think, the nearest gas station is too far away, action, reduce
speed to conserve fuel and try to make it to the next gas station or rest area.
But you can see how we're getting a script,
it's sort of like a problem solving script.
Now, let's think about this if we're the customer service,
we might want to go and generate a sequence of actions and
then we could input what the next response is that we get from the customer.
So and let's look at where we end off on this example.
Action, pull over to the side of the road and
turn on hazard lights to signal other drivers, call for roadside assistance or
walk to the nearest gas station as possible.
Think, roadside assistance is unavailable and
the nearest gas station is too far away to walk to.
So let's see what it decides to do.
Think, I need to conserve my phone battery and
find a way to get to a gas station as soon as possible.
Action, check if there are any houses or businesses nearby where I can ask for help or use a phone.
Think, there is a house in the distance.
And so you can see how it's building up the script of what to do,
it's doing problem solving, it's creating interesting situations.
We can also prompt it to keep going, we can keep having to go further based on inputs like what would happen in this situation?
How would you go about dealing with it?

We also see that the input is fairly constrained in what it's giving us,
it's still following our pattern.
Now, a key thing to go back and think about is the intuition behind this.
This works partly because this tool knows how to
predict what's next given a pattern.
It's learning from language, given this pattern, this is what I do next.
You can also imagine that it's seen in its training data,
lots of scripts, things that looked like, okay,
here's what is going on, you can almost think of like a movie script.
So and so said this, so and so said this in response, and
then you might see a description of what was happening.
Well, they get into the car, exit scene, something like that.
But we're using patterns, we're using structures that are likely to have been
things that it was trained on and seen.
So it's learned how to from these examples go and
extrapolate what should happen next.
The patterns and its ability to learn and intuate what needs to come next
in the pattern is based on that it's seen these types of things.
Now, if we went and we picked some type of pattern or structure that is very
different from what it's seen, it may be able to do, it may not be.
So one thing to think about is when we're structuring things,
try to pick simple patterns in terms of how we structure the text.
They're going to be something that it's likely seen,
something similar to in the training data.
So this idea of a prefix with some situation, an action,
think about trying to pick a format like that,
that looks like something it's seen.
Notice another thing we're keeping each example,
although it's wrapping around here, these things are actually each one line.
So it gets much harder to follow the pattern if we say like action, call in,
and then we have a whole bunch of stuff on multiple lines.
It becomes harder to see the pattern because not every line starts with one of
these prefixes.
So you have to think through the structuring and
the formatting of these things.

But at the base of it all, we're trying to give it some pattern that it can
quickly and easily pick up on in order to understand how to solve the problem.
We can provide the intermediate steps like we're seeing here, but
we don't necessarily have to give it the logic for how to decide the next step.
In fact, if we think about this,
when we say roadside assistance is unavailable and the nearest gas station is
too far away to walk to if I had to describe the rules of how to do that.
And this whole situation that we're going through,
if I had to really write a prompt that all the different rules in this situation
do this in this situation do that.
You can imagine it would take up a lot of space and
it might be really hard to do probably is going to have limited flexibility.
So one of the powerful things about these few shot examples is we can you tap
into the large language models, reasoning or whatever we want to call it.
It's computational abilities to go and figure out what's next and
come up with the next item in the pattern.
Required
English

Help Us Translate

Let's look at some examples of where we're
making mistakes in our few-shot examples.
Things that we can do wrong in
the formatting of our prompts and
what we can think about and learn from these examples.
Here's some few-shot examples.
The idea behind this is we put together
a not very good prompt to analyze how hard things are,
like is it soft or hard?
That's really what we're trying to get at
with this prompt and some few-shot examples.
Now this is not a good prompt.
This is a bad prompt and we'll see why.
What we've got is we have input,
as the prefix, we then got brick.
We've got output as a prefix and then hard.

We've got input as a prefix and then pillow, output soft.
Then we've got input car.
Now, notice what we've got here.
One, we don't have a lot of examples here, but two,
we've switched gears,
we started talking about
input and output and the hardness of
things using things that are really obvious, probably.
A brick is hard,
a pillow is soft,
I guess car is hard but it's not
as clear what we're getting at.
We also might just be trying to describe
some fundamental intrinsic capability
or properties of this thing.
It may not be about hard or
soft because it could be other things.
It could just be an attribute to
describe the thing itself
because pillows are generally thought of
as soft and bricks are normally thought of as hard,
those are fundamental descriptors
sometimes they use for them.
What we see is that when we
prompt ChatGPT in this example,
what we get is, it says,
fast or maybe transportation.
This shows us a couple of things.
This is a prompt that's not very
well-designed and because of that,
what we see is that ChatGPT isn't quite sure what
the next output should be from the pattern.
Now there's a couple of mistakes in here.
One is we've got input and output.
Input and output are really generic prefixes.

They don't really give the language model
much information about what
the task is that it's trying to do.
All we know is we're taking in
something and we're getting out something,
but it's not really telling
us what it is that we're describing.
Maybe we should go and say input,
brick, surface feeling, hard,
input, pillow,
surface feeling, soft
or maybe it's something about if you hit it,
what will it feel like?
There's all kinds of things we could do to give it
more context around what we're trying to get at.
We can also give it more examples.
But what we've got here overall in
this prompt is we've got a lack of information.
It's not detailed enough and we're not
giving enough information about what the task
is that's being solved and the result of that is that
the language model isn't really
sure what the solution is.
Now one way we can solve this
is we can go and before we give our examples,
we could give a little bit more context,
so we can provide some instructions.
We could say your output can only be soft or hard.
Now, one thing to note is as I'm
putting these things in quotation marks to try
to indicate to the language model
that these are the two values,
I don't want it to get confused and think I'm talking
about soft or hard in some other contexts,
but I'm putting these things in quotation marks,

giving them some emphasis that either it's soft or hard.
This may not always work because we still have
a prompt that's really generic.
That really doesn't have
a lot of context and a lot of meat to it.
It's still very vague.
We've got input brick,
we're still dealing with it.
Now luckily in this case,
adding this little bit of additional context
and specify to the language model that we only
want soft or hard as an output helps it figure
out that it should think about if I pushed on this car,
how would it feel like, it would feel hard.
That's the task that's being
solved and so it comes back with the right thing.
But that's our first thing to think about,
when you're writing few-shot examples,
are your examples detailed enough?
Are your prefixes that you're
using for these things meaningful?
Do they give additional contexts that
the large language model can use to
figure out what the next thing in the sequence is.
Another way to think about it is,
if you gave this to 10 different human beings,
would you have a chance of a few of them might be
confused about what you're trying to ask for?
If the answer is yes,
then you probably need to think
about how to refine your prompt.
If it's not fully clear what you're trying to
do in the examples of what
the process behind the examples is illustrating,
then you're probably not detailed enough.

Now one way you can solve that is by adding
some more instruction like
we have here at the top of this prompt.
Now let's look at another example.
I've got object plane speed fast,
object worm speeds slow
and I go through a series of examples.
Notice I'm giving a little bit more context here.
My speed prefixes a little better
because speed seems something that's quantifiable.
Like a plane is fast.
A worm is slow that should be more contexts.
We then go prompt it with object ball speed
and this is another important thing
to realize is it says variable.
One of the things we have to think about is we're trying
to get the large language model to
produce an output or to
classify or to fit into some boxes.
We need to make sure that it gets enough information in
the input to be able to
accurately put things into the boxes.
We need to make sure that
whatever information we're giving
it about what it's going to do,
always has enough context
that it can make a decision about the label.
Now in this case, a ball could be fast, it can be slow.
If I shoot a ball out of
a candidate might be going really fast.
If I shoot a ball,
if I put it on the top of a rocket,
it might be going really fast.
There's all different things that could
make a ball go faster or slower.

We have to make sure that we provide enough context.
Now, in general, the thing to think
about is am I given enough information?
Now I'm going to go back. I'm going
to change my prompt down.
I'm going to say object ball kicked by a baby.
Then we get immediately it says slow.
When you're building up the few-shot examples,
so you want to make sure one,
your prefixes are pretty descriptive.
Two, do the examples that you're giving,
really give enough information
to derive what the underlying processes that
is going on in order to go from
the starting input to the final output.
Do you really have
enough information there to
see what's underneath the hood?
One way to think about that is
if you asked a bunch of humans,
would they all know exactly what you wanted it to
do without any ambiguity.
Two, you want to make sure that you have
enough description before the examples if
needed to provide additional context.
Sometimes examples, you're just
not going to be able to get there with examples.
You may need to provide some additional rules or
other information at the start
of your prompt to help it along.
Really a lot of times what you're thinking
about is you have a limited amount of space
in your prompt and you have to think
about is this something that I can
better describe and an example

or is it something that I can better
describe in some instructions at the top?
What better means is will
it get a more accurate output and
will it take less space
or maybe less words to describe it?
If I can get it done very clearly in an example,
then maybe that's the right way to go.
That example doesn't take a lot of space.
On the other hand,
if it's a really simple rule and I can describe it,
I may be able to get a better response
by putting that rule up into the description.
Then finally, we have to make sure
that we provide enough information.
In this example, ball kicked by baby,
if we're trying to ball kick
things or we're trying to get a particular output,
we may have to make sure that when we give
input that there's sufficient richness and
detail in that input to follow
the rules or a prior examples that we've given.
Now this is particularly important if
our prior examples were pretty clear cut,
is probably something better.
Now, one could argue that our prior examples,
if we go back and look at this,
we said a plane was fast.
Well, a plane might not be fast,
a plane might be slow.
It might be sitting parked on the tarmac of an airport.
We also want to make sure that our examples
are rich enough and detailed enough.
Now sometimes we can get
away with simple things like plane,

fast, worm, slow in order to teach it.
But sometimes we're going to have to go into more detail.
We're going to provide more context.
Make sure and think about,
do you have enough real information?
Is it going to depend on the situation,
those things like that provide details.
Now one way to do it would be to
say just make some assumptions.
But one of the things that can
happen sometimes when you do
that is you may get more verbose output,
then you really want, it may not fit into that bucket.
But when you're writing these prompts,
think meaningfully about what is a meaningful prefix?
What information or instructions do
I need to have at the top and can
important information be better
conveyed as an instruction at the top or as an example?
Do my examples cover the space enough to have
enough good examples with rich detail for it
to learn from and derive what the underlying process is?
Then when I go and I give it the new input,
do I make sure I have
enough relevant contexts and information with
the new input in order
to apply the process that I taught?
Because we may teach it the process really well,
but then given a new input that just fundamentally
is missing information in order to apply that process.
Required
English

Help Us Translate

You may remember being told to show
your work when you were a student,

probably in elementary school,
or middle school, and you were working on a math exam,
and you had to explain how you got there,
or maybe if you've gone on,
and you've done other things where you had to
prove that something was true,
that you had decided.
Showing your work, explaining your reasoning,
is something that can be really
important in a number of different domains,
and it turns out that when it comes to writing prompts,
this can be really, really important.
Getting an LLM,
or large language model,
to explain its reasoning behind something,
actually can make the large language model
perform better for us.
Now this is a really interesting thing.
But if you think about it, it makes sense.
Because, we have a large language model that's
trained on different patterns of text,
and it's learned to predict what comes next.
If it can explain its reasoning correctly,
then it's more likely that it's
then going to produce the right answer,
because the right answer,
intuitively should come after the correct reasoning.
Similarly, if you think about it,
if it starts explaining the reasoning correctly,
it gets the first step right,
it's probably going to get the second step right,
because it may have seen
similar problems are reasoned out.
Now this is an intuitive explanation.
This may not be right, but I think,

intuitively, you can think about it,
is that, if you break down the problem
you're solving into multiple independent steps,
and you explain them,
and there's a natural sequence, and logic,
and flow to them, it's more
likely that the final output, which is the answer,
is going to be a natural extension
of the correct reasoning, and therefore,
correct it's more likely to be correct if
your goal is to accurately predict the next token.
Similarly, if you have the problem,
and you get the first step right,
and the second step right,
you're probably going to get the third, and the fourth,
as a large language model,
because you've been trained on this stuff.
There is a technique for this,
and it's called chain of thought prompting,
and I'm going to show you examples of this.
But it's really helpful to
remember when you're building prompts,
it requires some sophisticated intelligence
, and reasoning capabilities.
Now we don't always need this type
of breaking the process down step-by-step,
and explaining the process.
But when we do,
and we need that more effective reasoning,
it's really useful to understand
how chain of thought prompting works.
I'm going to start off with a positive example,
where we use it, and we
show what it does, and how it works.
We have some examples here that we're going to give.

I'm actually going to start off with
a negative example first,
I apologize, that was a mistake.
I'm going to start off an example that does
not use chain of thought prompting,
and I'm also going to
show an example where I'm going to make
a mistake in how I phrase the prompt,
and how I structure the prompt,
and I'll come back to this mistake in a second.
But what I've got here is,
I have three different questions
that require some reasoning.
In the first two, I've given an example of what I want.
I say, I've four bike racers start a race,
and travel an average of 30 miles per hour.
They each race for two hours,
is the total number of miles written by
all riders greater than 200,
and then I have an example,
and I say, Answer yes.
Then I say, question,
I have a staging process for a bike race,
to line up racers.
It takes 47 seconds to stage a group of eight riders,
and 67 seconds to get
the group to the starting gate, and ready to race.
I want a new group to start the race every 30 seconds.
Do I need eight groups staged at all times in order to
have races continually starting every 30 seconds?
The answer is, no.
This is the example I'm giving out.
We will see the reasoning behind these in a minute.
Then I say, I am in a spaceship without gravity.
I have a cup with a needle in it.

I move my foot on the bed,
knocking over the cup onto the floor,
I lift a book up, and put it on a desk.
Is there anything on the floor?
Now, this is the question
I'm trying to get ChatGPT to answer.
I've given it basically some examples of
questions with answers of either yes or no.
I'm trying to elicit a yes or no answer from ChatGPT.
I'm using few-shot examples to do that,
and I'm showing it questions,
I'm showing it answers.
Then I'm asking it for the answer about this question,
about a cup knocked on the floor.
Now, I've actually seen
some variations of this, and I apologize,
because I don't remember exactly where I saw
this first example of reasoning with the cup,
and the needle inside of it knocked on the floor.
But I thought this was an interesting one,
because it works really well for showing
the difference between chain
of thought prompting, and without.
This is without chain of thought prompting.
What is ChatGPT say because we asked it,
Is there anything on the floor?
The question was, I'm in a spaceship without gravity,
I have a cup with a needle in it.
I moved my foot on the bed
knocking over the cup onto the floor,
and then I have some irrelevant information,
then I say, is anything on the floor?
Now, without chain of thought prompting,
it says, yes.
When the cup with a needle,

and it is knocked over onto the floor,
both the cup, and the needle will end up on the floor.
In the absence of gravity,
objects will not fall downward as they do on Earth,
but they will still move in
response to forces applied to them.
In this case, the floor supplied by your foot,
knocking over the cup,
will cause the cup, and the needle to move,
and come to rest on the floor, lifting a book,
and put it on a desk does not
affect the position of the cup,
so they will remain on the floor.
It says, yes,
there's going to be something on
the floor, the cup, and the needle.
Now, let's look at the same example,
but with chain of thought prompting applied.
Here we go with chain of thought prompting.
With this one, we've
spelled out the reasoning behind each of the examples.
We go back to the bike racer,
and we say, the total be greater
than 200, and so we say,
reasoning, each rider will ride
30 miles per hour times two hours equals 60 miles.
I have four riders.
Therefore, the total number of miles
ridden by the riders is
4 times 60 miles, equals 240 miles.
Then answer, yes.
This is chain of thought prompting.
We're explaining the reasoning behind
the examples to the large language model.
In this case, we're explaining,

we're working out the math,
showing our notes,
how we got to this answer.
Then we do the same thing for the next one,
where we're staging a bike race,
and we want to figure out
how many people have to be staged at a time,
in order to start a race every 30 seconds.
We give the reasoning, each group takes 47 seconds,
plus 67 seconds to be ready to race.
I will need you to calculate how many races will
need to be run before a group is ready to race.
A group will have 114 seconds over
thirty-seconds equals 3.8 races
run before it is ready to race.
I can't have a partial group,
so I need to round up to four.
I only need four groups stage
to be able to race every 30 seconds,
so I do not need eight groups.
Answer, no. You see,
rather than just say,
here's the example, here's the answer.
We say, here's the answer,
here's the reasoning, and then here's the correct answer.
So then in this final example with the spaceship,
again, we're having the spaceship, there's no gravity.
We have a cup with a needle in it knocked on the floor,
and then in this example we tell it
to provide its reasoning first,
and then it's answer.
So we're prompting it,
and giving it a format that makes it
clear we want the reasoning first,
and then the answer.

What's interesting is we see
different reasoning than we saw before,
and we get a different answer.
If you remember before, we got the answer of yes,
the cup and the needle will be on the floor.
But if you look down here on this one at the bottom,
we can see that it says answer no,
because there is no concept of
on-the-floor in a zero-gravity environment.
If we look at this one,
this one is actually giving
us a better reasoned answer, in my opinion.
Now there's probably different nuances,
and I'm not an expert on this.
But I think if you don't have gravity,
you're not necessarily going to have the cup,
and the needle resting on the floor,
just because they got knocked off.
And so the reasoning is better.
In this one, it says, in a spaceship without gravity,
objects do not behave as they do on Earth with gravity.
When the cup, and the needle is knocked over, the cup,
and needle will not fall to
the floor as they would on Earth.
Instead, they will float in place,
or move in the direction of the force applied to them.
Therefore in the absence of gravity,
there's no concept of objects being on
the floors as we stand them on earth.
Basically what it's saying is,
you might move them,
or cause a force to be applied, but it's not,
they're not going to just sit there on the floor,
because there's no gravity,
and so it comes up with the answer, no.

You see right here,
we're getting a completely different answer
out of the large language model,
and the difference, is that,
one example we're giving it the examples,
we're giving it the reasoning behind the examples,
and then we're asking it to
provide the reasoning, and then the answer,
which is this case, and we're saying no,
there's nothing on the floor, which is probably
a better answer in this case.
In the other example, all we're doing is
saying, here's the question,
here's the answer, here's a question,
here's the answer, and we're not
giving it the reasoning.
Then finally we're saying, here's a question,
and it's going to try to produce the answer first,
which is what it does.
The difference is really only in what it does.
Here it's answering first,
but it's not explaining its reasoning first.
It's reasoning actually isn't quite as good,
and it's answer isn't quite as good.
Now, spaceship knocking a cup with a needle off on it.
This might not apply to you,
but it's an interesting example of
whereas more nuanced reasoning.
If you have a case where you need to think through,
and I say, think, very liberally here.
But if you need it to try to
get better reasoning behind the answer,
or a better computation,
whatever you want to describe it as, but intuitively,
if we thought through the process of getting the answer,

we have a good process of breaking down the problem,
and going step-by-step to a solution,
we're likely to get a better answer out,
a better solution out.
And so that's what chain of thought prompting is doing.
Is we're trying to elicit that.
Break the problem down into steps,
think about step-by-step,
how you're going to solve it, and the
way that we get there,
the way that we get to do that,
is when we give it few-shot examples like we do up here,
we explain the reasoning behind the answer,
and we will explain the reasoning first.
We'd say, this is what I'm thinking, and doing.
This is why I'm getting there.
Similarly, we do it on this one too.
We say, here's my reasoning,
and we provide the reasoning before the answer,
and the goal is to get
the large language model to do the same thing,
to follow our pattern of breaking down,
and explaining the reasoning before giving the answer,
to think through the problem,
to give the fundamental steps,
and then to produce the answer,
and we get better outputs in this case.
In most cases it's been shown that you're going to get
a better reasoning out of
the large language model if you
have it explained its chain of thought.
What it's thinking through,
what it's trying to do in order to solve the problem,
then you'll get a better answer out.
Required
English

Help Us Translate

Large language model is as powerful as they are,
they can't do everything on their own.
They're going to have to reach
out to other data sources and they have to
learn how to use other tools
to be really effective every time.
One of the things we have to learn how to do is build
prompts that can allow the large language model to
essentially reach in and query
new information to use other tools that can help it do
reasoning or computations that fundamentally is
not designed to do and so how do we go about doing that?
One way of doing that is with an approach called React.
The idea behind this is that we want to
teach essentially the large language model how to
think through a process and
understand steps where it needs to go and
use some other tool to take some other action
outside of it and get the result of
that tool and bring it back in as
some type of computation or
result that it can then incorporate.
There's a specific way of writing prompts to do this.
Now, this is similar to chain
of thought prompting like we saw before.
We're going to use some of those same concepts
from chain of thought prompting,
but we're going to expand
the approach to have this concept of certain things
that the large language model knows it needs to use
some external tool to perform.
Let's take a look at this.
Now bear with me because this is a longer example.

We're going to do a single example that we're going to
teach the language model with,
and then we're going to ask it to perform the same task.
In this example, we have
a task which is calculate when I need to arrive
at the Music City BMX national race for
my son to be on time for his 9:00-10:00 open race.
BMX is a bicycle racing sport.
Some of my examples lately have been around BMX.
Music City BMX is a local BMX track here in Nashville,
they house one of the big national races for BMX racing.
One of the things, as I need to figure out,
when do I need to show up at the track?
Because these are all day multi-day events.
Sometimes you may want to figure
out how late can I sleep before
getting in the car and driving in order to arrive?
What happens is in these races,
you have hundreds of
[inaudible] little races that
are essentially 30 seconds each.
These are called motos that are run
and you want to figure out when your moto 87,
for example, well,
how long do I need to wait after the start,
before that race is going to start?
Sometimes there's a balance of trying to figure
out when do I need to be at the track actually,
as opposed to when does the race start?
What we're going to do is we're going to have it
solved this problem for us.
We're going to pretend to solve this problem.
This isn't actually solving it,
but we're going to show you how this process works.
We give it a task and we give it an example

which is calculate when I need to
arrive at the Music City BMX
national race for my son to arrive on
time for his 9:00-10:00 open race.
Then we say, think,
I need to find out what time the first race begins.
I can use a web search of
the Music City BMX site to get that information.
Remember, this is our example.
We're saying, here's the thought process.
I need to first figure out the first step.
Now you remember back to chain of
thought where we're trying to have
it explain its reasoning,
we're doing the same thing.
We're explaining the reasoning behind it.
We're saying, I need to find
out what time the first race begins.
We're explaining what we're trying to do.
I can use a web search of
the Music City BMX site to get that information.
Then we have the next thing we do which is
we say action search,
and then we give the Music City BMX website.
Action here in the search is basically
a tool that it's learning to use.
It's learning to use some external tool, in this case,
a web search so it knows that there's
certain information it does not have
and so we're teaching
it when you need information that you don't have,
go and search for it and you
can search for it with a website.
Then it says, the result we tell it is
all races start at 09:00 AM so then we say think,

I need to find out how many motos
run before the 9:00-10:00 open race.
Motos are essentially the individual races
that are going to run before that race.
I can use a web search of
Music City BMX site to get
that information and then we show it
again another example of searching
assuming there's a search tool available to
it for the Music City BMX site.
Then result, we show it the result.
There are 30 motos that run
before the 9:00-10:00 open race.
Then we say, think,
I need to find out how long each moto took last year.
I can use the video live stream from
the USA BMX website and
record how long the first 10 motos took.
Then we say action video and we
give it some fake video and pretend that,
hey, we were able to go and pull this video down in
time something and then result.
The first 10 motos took five minutes.
Think if 10 motos took five minutes,
30 minutes, we'll take 15 minutes.
The race starts at 09:00 AM so I need to
arrive before 9:15 AM when the race starts.
I will arrive 30 minutes early at 08:45 AM to be safe.
Remember this is our faux example.
We've imagined a scenario
where I need to solve this problem and I'm using
a series of web searches and then watching
a video from a prior year to
figure out how long things took and I'm
trying to teach the large language model,

in this case, ChatGPT,
to use these same tools to solve the problem.
Then I ask it,
here's the new task.
Calculate what time my 36-40 novice race
will end at the USA BMX Grand Nationals
and then I say think,
to prompt to start thinking what is the next step.
You'll notice what it does. It then
says I need to find out
what time the first race begins at
the USA BMX Grand Nationals,
I can use a web search of
the USA BMX site to get the information.
Action search USA BMX Events Grants.
Notice it's picked up on a couple of things.
Is picked up on the idea that has this tool called
search that it can use to go and pull in information.
You'll notice it's also picked up
on the format of the URL.
Now this format I had was just
completely fictitious in this example,
but it's picked up that, oh,
there must be this formatting,
there must be a pattern behind these URLs that I can use.
I can go and search and use some knowledge
of the structure of
those URLs to pull in the right information.
It's learned essentially how to
go into a web search and then it says result.
Note as an AI language model,
I'm unordered perform web searches.
Please assume a hypothetical result
of the purpose of this exercise.
The first race begins at ADM. Now what's

interesting here is that we're actually
integrating this into a real system.
We can have the system whenever it sees a result,
stop pulling the actual result,
put it in there,
and then basically feed this first,
sec chunk back to
the language model to then continue the process.
We don't actually have to worry about if it's
going to fake the rest of this,
we could just have it go
and we can incrementally get to every,
each time we get a new result
out that we actually need to fill in,
we could go and run the web search
and fill in that result.
It then keeps going. It says I need to
find out how many motors run
before the 36-40 novice race at BMX grand nationals.
I can search and it goes and searches on
the grants and then it has a hypothetical result.
There are 120 motos.
Goes and thanks again.
I need to find out how long each moto took last year.
I can use the video live stream and then
it picked up on the URL pattern.
It's learning a lot of things.
It's learning how to use the tool and how to input
data in a particular format
for it based on what I showed.
It then says, okay,
here's the result from that.
Then it says, think if 10 motors
took six minutes each or took six minutes,
then each moto took an average of 0.6 minutes,

therefore, 120 motors will
take approximately 72 minutes.
The first race starts at ADM,
so 36-40 novice races start around 9:12 AM.
I need to find out how long
the 36 novice race itself will take.
It goes and does that in and then it
says if the 36 novice race starts around 9:12,
then it will end around 9:15.
This is the approximate time
my race will end at the USA BMX nationals.
What's interesting is that you actually end up,
in the end, getting slightly off track.
It gave me the begin and end.
It's telling me the begin time,
but it also went and calculated the end time,
which I may or may not want.
But the more important thing is,
is that along the way it followed my pattern.
Up here, I taught it this concept of searching.
I had action search, musiccitybmx.com.
I had another action search for musiccitybmx.com.
I had an action video right here.
It learned how to use these tools and then it's
applying the exact same tools down here.
It's at applying search. It applies it again.
Then it applies video.
It's applying it using the format.
You can imagine it's essentially learning to
run a web search to watch a video.
But again, these are tools,
so it could basically go and send
some information off to the tool,
get an answer back,
and then pick up in its reasoning.

Now the way that we would do this
is we would have to have a way if we're
going to use this
to actually when it gets to the point where it says,
here's what I'm going to do,
we would stop it,
and go give that data to the real tool,
get the answer, and then come put it in for the result.
Then we would feed this whole thing back
in with the correct answer is the prompt.
It then hopefully we'll run through
up until it gets another action in
which case we would go and get
the real result from that tool,
put it into the prompt in
continuing and we would continue
down until we got to a point where we had an output.
Now a better way of doing this would be to have
some way of saying,
okay, I'm done and I
should have put that into the prompt here.
But this is doing basically, it's teaching it.
Here are some tools that you can
use if you use this format.
You can go and run
this particular tool and you described the tool to it.
Now, other ways to do this,
if I didn't have every tool used,
I could have edited
this prompt and I could have had a description of
each tool and the format and
other information that it can use to help
us figure out which tool to use.
Now in my simple example,
I did it just based on giving up patterns that I

had directly in the few-shot examples that I provided.
But you can also have other ways of
catalogs of tools in their description.
It can think about which ones to use
at the appropriate time and you see how it's beginning to
reason and know which tool
do I need to use at which point in
time to get which type of information so you
can teach it to do this problem-solving.
You can go and integrate it with
other tools and you can build
these prompts that allow it to tap into other tools.
You see this in a variety of different frameworks.
There is a tool you may have heard
about called lang chain,
which is a framework for writing
these types of automation processes.
They use react underneath the hood in order to be able
to go and do some of
the interesting things like this that they do.
Required
English

Help Us Translate

We're going to be faced with the problem with prompts of,
the large language models themselves are going to be moving very rapidly,
they're going to be continuously evolving.
And we've already seen that, we've seen ChatGPT and GPT-4, we've seen LLaMA and
Alpaca, we've seen Vicuna and lots of different ones are continually coming out.
And one of the things we want to know is, when we develop our catalog of prompts,
our patterns, we put lots of work in developing prompts that work well for us,
how do we evaluate them and make sure that they're maintained over time?
Because if we change our large language model,
does that mean that all of our prompts are going to break?
Or if we change the data that we're working on slightly,

does that have ramifications for us?
So we need better ways of helping to evaluate the output.
Now one way, of course, that we're going to do this,
is we're going to have humans look at the output in many cases.
But we'd like to do things that could probably help us scale better, be
able to actually go and do sort of large scale automated analysis in some way.
So how do we go about doing this?
Well, one of the really interesting things with large language models is
we can actually use them to evaluate themselves.
This can be one of the inputs that we use to help determine how good our prompts are and how maintainable they are over a time, is we can use a large language model
to grade either itself or the outputs of another large language model.
So, we've seen the examples of this where we have,
they've actually taken large language models and
they've had one large language model teach another large language model, essentially.
But we can do something similar, in that we can take outputs from a large language model and actually feed them back into the model itself to grade them.
Or we can actually go and take another large language model that we're working with and have its outputs graded by, for example, a better large language model, maybe with more parameters that can, as more powerful.
So, let's look at an example of this and how we might build a prompt to do it.
So in this example, I'm going to teach the large language model,
in this case ChatGPT, how to grade the output of a prompt.
I'm not actually going to show it the prompt in my example, I'm just going to
teach it the grading process, so that it can help automate it for me.
And so I'm going to use some few short examples.
So as input here at the top, I'm saying, here's my input, and
it's a bit of text about Vanderbilt University from Wikipedia.
It identifies that Vanderbilt University was founded in 1873.
And I explain, here's what the output was.
The output was, the following is a list of events and dates and the output, Vanderbilt founded in 1873.
And then, the explanation is that the output has unwanted text at the start and

should only include the names and dates.
And so I gave it a grade of five out of ten.
So this is the output that was produced.
And then I'm providing a human explanation of what is wrong with the output,
of why it's not getting a perfect score.
And then I'm grading it and giving it a five out of ten.
So I'm going through and
I'm doing a couple of these by hand to show it what I want.
And then I'm giving it another example with that exact same piece of text.
I'm saying the output was just 1873, and the point of this is that,
if you can probably sense it by now, is I want a prompt that's going
to extract events and the dates that the events took place.
And that's it, I want an event name separated by a comma and
then the date of that event.
I don't want any other text, I don't want an explanation,
I just want names of events and dates, and I'm showing how ChatGPT can
grade the output to see if it's matching that format.
I'm not even showing it what the prompt is, I'm not showing it what the task is,
I'm giving it examples to help learn.
And then I say, explanation, the output is missing important information about
the event that happened on that given date.
And I give it a grade of three out of ten.
And then I have one more with the input,
which is that same paragraph about Vanderbilt.
I show Vanderbilt University comma 1873,
which is the ideal output for my fictitious task.
And then I explain that the output exactly.
And then I actually cut it off, because there's a typo here.
It's really supposed to be met expectations or something like that.
And then I give it a grade of ten out of ten.
What's interesting is this doesn't even matter that I made a mistake here in my
explanation, because it still works.
And so then I give it a completely different input.
I say, here's a different input.
Again, it's about Vanderbilt,

and the history of the English program at Vanderbilt in creative writing.
And so I give it this input.
Another piece of text on that, I say the output was, and I give it what the output was, the Fugitives and South Agrarians, first half of 20th century, and
then Vanderbilt is a founding member of the southeastern conference, 1966.
And, that's essentially a prompt for
ChatGPT to then go and grade that output.
And so it then provides an explanation of the output.
It actually mentioned something that I didn't grade in there.
But what's interesting, it says there's inconsistent capitalization,
and then it gives it a grade of nine out of ten.
So as a human labeler and a human performing this task, I actually ended up with a nine out of ten because I then wasn't consistent in capitalization.
But you can see it did a good job.
It took the sort of substance of what I wanted, and
it performed the grading task that I had given it.
And so, we can actually use large language models to evaluate the output of prompts.
Now, there's a number of ways you could potentially use this.
One is, if you have some system that's going to go and make decisions or generate information with a large language model like ChatGPT or
some other model that you're running,
you might want to have the model check itself or another model check itself.
Or you might want to do this multiple times, where you take the output that you're getting and you give it a grading criteria similar to what I've done here.
And you've developed a prompt with grading criteria, you might have
multiple prompts to the grading criteria, and you all have them score the output.
And then, if the score is below some threshold, well, maybe you want to have a human look at that output, or maybe you just want to retry it and
see if you can get something better to be produced.
And so, this gives you an extra sort of tool of,
how do we go and evaluate what's coming out, make sure it looks good,
and still have sort of the power and sophistication of these models.
And this is one way to do it.
Now there's lots of ways that you could phrase the prompt.

You might want to go and
look at the alternative approaches pattern to develop some different ways.
This is one example using few short examples.
And I could have gone back and say, we could have said act and
use the persona pattern, act as a prompt critic.
Perform an, give the output that
I provide you a grade in terms of
how well it matches the expectations
of the prompt designer.
I am going to give you examples of grading prompts.
And so we could have gone and done something like that.
I did a relatively simple prompt, but
we could do a much more sophisticated prompt using some other patterns.
Again, it's going to generate an explanation.
And then at the end of it, it should generate a grade for it as well.
Play video starting at :7:34 and follow transcript7:34
And so it will then go through and give us an eight out of ten.
So you notice we got eight out of ten one time, we changed the prompt slightly,
we got nine out of ten.
Either way, it looks like that's a pretty good output based on our grading criteria
and what we want.
And so we can use this sort of tool in our toolbox.
When we need to evaluate a prompt and the output of a prompt,
we don't have to always fall back to having humans look at it.
One thing we can do is,
we can fall back to having the large language model itself evaluate the output.
And then we can build different types of automation around that, or
we can use that as a trigger to escalate to a human.
Now, of course, if we have a lot of really good examples of grading,
that helps us a lot.
But we don't have to have a ton.
If you see in my example, I had just three examples of grading the output and
it did a good job of figuring out what I wanted.
Required
English

Help Us Translate
One thing that I find really helpful when I'm trying to
learn a new topic or improve my skills,
I will just have fun is to play a game.
When we can play games with
new knowledge that we're trying to acquire,
it's usually a fun way to really become
masters of that knowledge and test our ability to use it.
Particularly, the games are fun and challenging and
so one of the patterns that we
can use is called the gameplay pattern.
The idea behind this pattern is
we're going to ask ChatGPT
or another large language model to play
a game with us and it's going to decide the rules.
It's going to be the game master in most cases.
Now we can give it a variety of
different scenarios that can
govern what its role is in the game.
But we're going to look at gameplay
where the large language model
is serving as the game master.
It's creating the rules and it's driving the game.
We can then take a game and we can say,
I'd like a game on this topic or
that has these basic rules fill in the rest.
One of the things that's really nice
about this is that if we say create
a game and we give it
some basic structure around the game,
it can tap into its training data in
order to populate the game with content.
If you think about games like getting the rules right,
obviously takes a lot of thought.
But then there's a whole other challenge for a lot of
games of how do you create a lot of

great content to go with
that game and so this is a pattern
where we can tap into the training data from
the large language model in order to
help it populate the content.
Now we can also do a pretty good job
of deciding the rules as well
but we're not going to necessarily look in
these examples that complex rule generation.
But we are going to look at
a really useful example of this and I'm going to
show you how to create a game to
improve your own prompt engineering skills.
The basic pattern of this is to say,
"We're going to play a game
and we say involving or about or whatever,
and we want to give it the topic."
We start off this pattern by saying
we're going to play a game and here's the topic.
Then we decide some more information
about what are the rules of the game.
In this case, we're going to have some simple rules.
You're going to give me a simple task that
can be accomplished via prompting you.
I'm telling it we're going to play a game,
you pick task where I could
theoretically prompt you in order to solve that task.
You imagine what we're trying to do is
have it play a game with us where it's going to
give us a challenge of something we would
try to write a prompt for and then
we're going to write the prompt and see if
we did a good job and then we're going to say
your task should all have a reasoning or
programming component to them

although they shouldn't require source code.
Now the reason I do this in this game is,
you get more interesting questions in
my experience if you have some logical component to this.
Remember, we talked about
prompts as a way of programming
a large language model that everybody can be
a programmer and that's what we're trying to get at.
We don't just want to generate a bunch of points about
lonely tree standing in the middle of the field.
We'd like to do more than that.
We'd like to have to solve problems that have
some logical or interesting component to
them and that's what we're
getting at here with this problem.
We're going to build a game.
It's going to challenge us to develop
different prompts and then we're going to
try to make these tasks that we're solving
related to programming or logical things.
Now next, we go on and we give it some more rules.
I will try to write a prompt to solve the task.
You will give me the output of
the prompt and tell me how well it solved the task.
It's going to give us the task and you notice we're also,
we're adding it, ask us questions,
so it's going to give us the task.
The task is going to be something that
can be solved with a prompt,
will write the prompt,
and then it will run the prompt in
order to generate the output that that prompt
would have output and then
it's going to tell us how good our prompt was.
This is interesting. We're actually having

it play a game where he's pretending
to be prompted and then
telling us how good our prompts are.
Ask me questions until I tell you to stop.
You will ask me a question,
wait for my response,
and then ask me another question
after telling me how I did.
Ask me the first question.
It says, "Great.
Let's get started with this first task."
Given a list of numbers,
determine whether the list
contains any duplicate elements.
If it does return yes, there are duplicates.
If it does not, no duplicate is found.
Now we've got a specific prompt engineering task.
We're going to have to figure out how do we
write a prompt where we give it a list of
numbers and have it return
one or the other of these things and it's interesting.
This is one of the challenges that
we often face when we're dealing with
a large language model is getting it
to generate a precise output.
It's giving us something where it has
a thought component to it of like how do we design
and tell it about what it is we're trying to achieve.
But also, it's also got an output component.
We're trying to match a specific output format.
Then it says, "Please write a prompt
for me to solve this task."
I'm going to solve this task using few shot examples.
I give it a list of numbers,
I show it some examples.

I say basically in my prompt essay,
here's the numbers 1, 2,
3, and then I have duplicates.
I'd say no duplicates found.
Then I have numbers 2, 2, 2 duplicates.
Yes, there are duplicates and I'm following
the output format it told us and then I'm going
through and giving you a series of few-shot examples to
solve this and then I prompt it.
My final end of my prompt is basically
duplicates question mark on the list I just gave it,
which is solving the problem that it's
asking for or trying to see what it says.
But basically, I'm asking it,
does this list have duplicates?
We're hoping that what it says is
no duplicates found and match
this exact format that I provided before.
It does, it says no duplicates found right here.
Then what's nice is it does the other part of
the game that we explain your prompt
effectively communicate the tasks and provided
examples of different tasks are of different cases.
The task was to determine whether list of
numbers contains any duplicate elements and
you provide it with examples
based on your prompt that was able to correctly
determined the list does not
contain any duplicate elements.
Then it goes on and it says,
let's move on to the next task.
Now it's playing the game,
so it's moved onto the next task.
Given a sentence, count
the number of words in the sentence,

and return the results in the format.
The sentence contains X words,
where X is the number of words.
It's giving us an output that I have to produce.
Notice it's given an output now where I have to
put the output in a different particular format.
Before we just had to go and say
there's duplicates are no duplicates.
But now we're having to do something
a little more complicated.
We're having to count words in
the sentence and then make sure
that count shows up in a particular place and the output.
This is exactly testing our knowledge of
prompt engineering and some of the patterns
and other techniques that we've seen in the past.
Please write a prompt for me to solve this task.
I'm going to use a pattern called
the template pattern to do this.
What we say is I'm going to give you
a template for your output.
Number is my place holder in all caps.
Fill in my placeholder with your output.
Please preserve the overall formatting of
my template is,
and then I have the sentence contains number words.
I'm going to solve this problem.
Here's my prompt to go and solve
this problem and then I say
how many words are in the following sentence,
the cat jumped on the mat.
It says the sentence contains six words,
which we see is exactly the format that it was looking
for and then it tells us that grades us on our prompt.
What's really nice about

this gameplay example is
that from now on you can go and use it
to test your ability
to do prompt engineering and you don't
have to think of what are
prompt engineering tasks that I could try out.
You can just use
this prompt to have it generate the content for you.
It can think of the ideas of the problems for you to
solve with prompt engineering
and we can see in this simple example,
it's already gone untested some different dimensions,
a prompts for us and given us
some opportunities to try out some different patterns.
Then goes on and it gives us another interesting one.
Given a date in the format,
yyyy, mm, dd meaning year,
month date determine whether
the year is a leap year or if not.
If it is a leap year or the year is yyyy,
is a leap year if it's not. How do I solve this?
I'm going to go through and give
it some few-shot examples
similar to a dead before.
But it's testing my ability to
figure out what do I do to solve this in
this particular case and then it gives
us the right output that 2023 is not a leap year.
Now what's really nice about this is remember,
it's supposed to be checking
our prompts and testing our prompt engineering skills
and I want to show you something that I didn't even
notice until it told me and tested me.
I said date 2009,
7, 24 so July 24th,

2009 and then in my example,
I said leap year right here and then I
said the year 2004 is not a leap year.
My prompt example actually had a mistake
because I'm asking if July 24,
2009 is a leap year and then in my example,
I give it the wrong template.
I say the year 2004 is not a leap year.
It should have been the year 2009 is not a leap year.
Then it actually notes that right here it
says there's a small typo in the third example.
The year 2004 is not a leap year,
should be the year 2009 is not a leap year,
but did not affect my ability to
understand and respond to the task.
But it does give me feedback on my prompt engineering and
also shows me I need to be
careful about the examples I'm giving it.
If I make a mistake in my example, in this case,
I got lucky and it didn't affect things,
but it could in future ones.
It then goes on.
It gives me a new one which I
didn't actually go and solve.
Given a string of characters,
determine whether a string is a palindrome.
The string is a palindrome
and if the string is not a palindrome
return the string is not
a palindrome replaced string with the actual string.
Play video starting at :11: and follow transcript11:00
It's getting a little repetitious in here,
but we could go in and inject
additional ideas to help tailor the output even more.
You can imagine we might want to go in

and target our prompt engineering a little more.
Now, we've been
learning a lot of prompt patterns in this paper.
One way we can adapt this game is we can give
it a more specific topic about
the type of prompts we're trying to develop.
In this example I'm going to take
the description of the recipe prompt pattern and what's
helpful is you can go to
our prompt patterns paper and you can actually take
a little section or explanation of
those patterns along with the name and you can
provide it into this game as contexts and it does
a reasonable job if you provide
enough information on the pattern and examples of
asking you questions that would
use that pattern to solve it.
In this one I go and I
actually modify the game and I say,
all of your questions should revolve
around the recipe prompt pattern.
This pattern provides constraints
to ultimately output a sequence of steps given
some partially provided ingredients that must be
configured in a sequence of
steps to achieve a stated goal.
The pattern is not about cooking per se.
Now, I added this in here because I
knew in my heart
that it was going to want to ask me cooking questions.
That's fine but I've done enough of those.
But creating a complete sequence of steps to achieve
a goal given an incomplete list of intermediate steps.
I really want to use this cookie and is obviously
a natural way of talking about recipes

but I don't want it to clue in too much on that.
I'd rather do some other interesting tasks.
Then it says your goal is to create
a prompt that instructs me
to generate a sequence of
steps for assembling a simple wooden bookshelf.
The bookshelf has four main components.
The top and bottom panels,
the side patterns and the side panels and
the shelves you've been provided
with the following ingredients.
Attach the top panel to the side panels
insert the shelves into the designated slots.
Using these intermediate steps create a prompt that will
result in a complete sequence
of steps for assembling the bookshelf.
Remember the prompt should be designed
to fill in any missing steps
and provide a clear order for completing the assembly.
Now what's interesting is,
is this an interesting problem
because it has to have an understanding of
a bookshelf in its structure
and you'd have to go and write a prompt
that's going to elicit the right problem-solving behavior
on assembly of a bookshelf.
It's a cool little problem.
I then I went and I took it and I said, all right,
well let's use another one of
the patterns from the paper.
I adapt and I say,
we're now going to change the topic of your questions.
The topic is now the context manager pattern and then I'm
basically providing the intent and
contexts from the paper and I'm saying,

this pattern enables you to
specify or remove context for conversation.
I'm not going to go into the details.
It did a pretty good job.
It asked me a question about
the Eiffel Tower and its construction.
Then I went and I said change the topic.
Again I said we're going to
change the top of the question,
the topic is now the template pattern.
I describe what the template pattern
is and it actually does something
that follows what is exactly set in here,
which is you might want to generate a URL.
We might get all URLs.
We might want to tweak this a little bit,
but it does a good job at then gives me a task.
Your goal is to create a prompt that instructs me to
generate a URL for
a product page on an e-commerce website.
The URL should follow
a specific template structure and it gives me
the structure of product id/name,
the product id and product name are
placeholders that needs to be
replaced with the actual values.
You can see how I've got this game.
It can test me on prompt engineering.
I'm learning about prompt engineering.
It can actually give me helpful feedback.
It caught on a typo in my few short examples,
and then I can go and tweak it.
I can say, now the topic is,
we're going to do a game based
on this particular prompt pattern.

Now we're going to do one based on this pattern.
What you'll see is you'll get
interesting content that will
come out of it and challenges
of things you can try to prompt.
Now they may not all be good.
They may not all be interesting.
You may get in a rut where
they're like getting repetitive.
But that is on you then to think about what additional
information do you go and
inject into this game to make it fun again,
or to make it more variable.
I was pretty surprised at
the quality of questions that I got,
and it brought in a lot of topics like
assembling a bookshelf is
not what I would have thought about.
That's certainly an interesting one
or when it asked me about
information related to
the Eiffel Tower and its construction.
All of these things bring fun to the game.
Or is it a leap year or not?
All of these things are interesting ways and all I
did is gave it the base rules and then
the topic area and then it pulled in
all this information to populate
the game with things that helped keep me engaged,
new content, new ideas to
challenge me and make me have fun.
In this case writing out the rules is a lot easier than
thinking up all the content for
the game which it does really effectively.
Required
English

Help Us Translate

Getting a large language models output in
exactly the format that we
want can be a challenge sometimes.
The next pattern that I'm going to teach you about
is going to help you get exactly the format
you're looking for and tell ChatGPT where
you want the information
that it's producing put into the format.
You're basically going to give it a template and
then ChatGPT is going to go and fill in your template.
Just like you might do with
an assistant where you'd say here's the template that I
use for creating these form letters and here are
my different placeholders within my template
and then your assistant could go
in and fill in that template.
That's exactly what we can do with the template pattern.
The idea behind this pattern is we would like
a specific template followed for our output,
and we can describe that template to
ChatGPT and then have it
produce all of its output using that template.
Now, we could do this with other approaches like
few-shot examples but we're going to do
this with the template pattern in this example.
Here's what the template pattern looks like.
We say, I'm going to give you a template for your output.
We're queuing the large language model that
there is a template that's going to be
in play and it needs to follow it.
Capitalized words are my placeholders,
so we're telling it when you look at the template,
if something is capitalized,

that's something that you're going to
replace with the content that you're generating.
Fill in my placeholders with your output,
so there's our queue to it to
go and replace the placeholders,
and please preserve
the overall formatting of my template.
We don't want ChatGPT to just
decide on its own what the format should
be and ignore everything
that we're giving it in terms of instructions.
Then I say my template is,
and I've got this funky little template down here,
and I'm going to talk through it.
I have question with
three asterisk in front
of it and then three asterisk following it.
Now, what this is going to do actually
is this is using some knowledge of how
ChatGPT in the ChatGPT interface
formats the information that we see,
so those three asterisks you are basically going to meet.
You're going to italicize and bold,
this is like a heading.
It's going to say question and it's going to be
in italics and it's going to be in bold.
Then we want it to fill in a question,
so question which is an all caps.
Notice we're differentiating
between things that are part of
my template and things that are capitalized,
which are things that are
placeholders that it's going to fill in.
Then we're saying answer, similarly,
it's going to be in italics and bold.

If you want to figure out how this works,
this is called Markdown,
so basically, ChatGPT is
rendering the output to you in Markdown.
I'm just using this knowledge
to help me out a little bit.
I'm creating a template that includes
Markdown and then we're using a placeholder for answer.
Now, you can guess I'm going to have
it generate questions and answers,
this is what's going to come next.
But right now I'm just telling
it what the template looks like.
Here's what my template is going to
be and I'm going to ask you to fill it in.
Then I tell it, I will give you
the data to format in the next prompt,
create 20 questions using my template.
Of course, it says it's happy to do that.
Then what I did is I pasted in,
from Wikipedia, an article on Paleo-Indians.
My son was studying Paleo-Indians at school,
and so I thought it would be fun.
Can I generate some questions about Paleo-Indians for him
and see if he can answer them or learn more,
and it might be interesting
to talk about these questions.
I paste that in and then what we see is we get
output that follows the template that we gave it.
If you remember, the first thing
was question and it was not in caps,
so this was something we wanted it to preserve,
and then we had question in all caps,
and ChatGPT has generated
a question about the Paleo-Indians,

who were the Paleo-Indians?
Now, interestingly, it has decided to put this in
all caps too which actually works for the format,
I didn't actually mean
for it to put it in all caps, but it works.
Then I had answer which was not capitalized,
and then I had answer was something to be
filled in that was an all caps,
and it fills in the answers.
It's creating questions about Paleo-Indians,
who were the Paleo-Indians,
and then it's filling in the answer to the question.
But the whole thing is structured using my template.
I told it generate 20 of these questions,
and so it's basically repeated my template 20
different time but filling in the content.
Now, notice these are really powerful placeholders.
The placeholder I gave it was question,
and so it has to understand what is
the meaning of the word question.
It looks at and has to understand to some degree,
and people may argue it's not really
understanding anything, it's doing something else.
I'm going to talk about it in terms of understanding
because I think that's a natural way to talk about it.
If we later discover it doesn't
really understand, it doesn't matter,
the template pattern is a great way of using
this and it's easier to
talk about it in terms of understanding.
It's understanding the placeholder
that it means question,
that your question that you generate should go here.
Then it's understanding answer,
answer is where the answer to the question should go.

It knows, from my prompt,
that it's supposed to generate 20 of
these questions based on the content I'm giving it.
It's extracting 20 questions
about the Paleo-Indians and producing
the question and the answer and formatting
it according to my template, like we see here.
We get a bunch of questions that we could go
and then have a discussion about.
There's a first example of it.
We can also go and take it and say
take this a little bit further.
I took my colleague Doug Schmidt
and I imagined like, well,
what if I wanted to go meet
Doug Schmidt for the first time
and I wanted to try to know
something about him and have something to
talk about and make sure I was
versed on his background so I can intelligently
talk about what he was doing,
what he was interested in, his background.
I just took here is webpage here.
Now, ChatGPT explains to me,
hey, I may not have a version
of this web page before 2021.
It's fascinating than it actually is able
to take information from a web URL,
PSK, little trick, it may not be up-to-date.
Of course you should
if you're going to use this before meeting
somebody own the information
and make sure it's actually correct,
but it could give you a starting point.
Then it goes and generates 20 questions

using the format that I gave
it about my colleague, Doug Schmidt.
Now, looking at the questions so far,
they seem pretty good.
Now if we wanted to do this in a more reliable way,
we could have pasted in
the actual questions or
the paste and the actual information
from his web page in order to get
information about him and generate 20 questions.
Now, I'm going to show you
another example because I talked a
little bit about these placeholders,
that they're not just placeholders,
there's a real deep understanding of
the meaning of the placeholders
when you're using the template pattern.
You don't just have to think of them
as being really easy to match.
They can also be smarter placeholders.
A placeholder that you might give to another human being
if you wanted to explain to
them how to use the template you're providing.
Let's look at this example.
I'm giving it the same preamble for the template pattern.
I'm still using capitalize words.
I'm still using Markdown
to try to get this formatted in a way that I want.
This case, it's going to create headings wherever I have
these hashtags and so
what I'm doing is I'm saying bio and name.
I'm going to paste in information about
a history of Vanderbilt and I would like
to get a list of people from that history of Vanderbilt.
I want the names of the people and then I want

an executive summary for
each person and a full description for each person.
Now, I'm stating that
my executive summary is a one-sentence summary.
You notice my placeholder is more than
just go and talk about this person.
I'm actually putting constraints on what I want there.
I want one sentence that summarizes
this person's role as described in
the information I just gave
and then for a full description,
I say I want a one paragraph summary.
Now, you can imagine you could go and
capitalize a complex set of instructions,
a one paragraph summary that only considers
their history as part of Vanderbilt before 1800s.
Now I haven't actually tested that one out,
but I suspect it would actually work.
But the key thing to note is I'm giving it
more than just a sample match.
I'm actually giving it a match on summary,
go and create a summary here,
but I'm also putting constraints.
What it should be, it has to be
a one paragraph summary or a one-sentence summary.
This is like a power of
the template pattern is that we can have
complex instructions in the placeholders that we're
actually going and replacing.
Now I'm going to take a look.
I patched it quite a lot of information.
This is what we get as a result,
is we get the headings as we had before with the name.
That'll make sense but then if we
look at the executive summary,

it's exactly one sentence.
It's not only filling in the template,
but it's filling it in with
the rules that we've described in the template.
Now we've got a one-sentence summary
in the executive summary,
and then the full description we've
got a one paragraph description.
It goes through this for every single person that it
extracts from the information that I give it.
The template pattern is a way to help you
get the output that you want in the format
you want and you can
use the placeholders in really sophisticated ways.
They can be something that
can semantically extract information.
I can dump a bunch of information in and say,
I want to know the people.
Then it has to understand who are the people in this box,
this bucket of information that's just been dumped
in and has to be able to extract the people.
I can say, I want a summary of each person,
but it has to be one paragraph.
You can have instructions and
constraints on these placeholders.
You can also have sophisticated things.
We could have said, for
example what is the role of each of
these people in the history of
Vanderbilt and it could describe their role.
The placeholder can not only format the information,
it cannot only do simple extracting and matching,
but it can also do sophisticated things,
they can follow rules or
do things that would be really time-consuming

for a human-like take and extract
information that would require a human to
read through and understand and then put it in.
We're saying create
a one-sentence summary of this person.
Well, if I was going to have an assistant do that,
they'd have to go read all the information and
then make a list of all the people for
each person I have to write first
the one-sentence summary to try to
capture everything about that person.
Then they have to go write a one paragraph summary.
That's really fast, really
sophisticated intelligence that it's using
in the key to getting the output that we
want from it is using that template proud.
We're saying this is the format that
you're going to give it to me in this is
exactly where everything goes and here are
these really sophisticated rules
in the placeholders to help you decide what goes in,
how it gets formatted, what to exclude.
It's a really powerful pattern for getting
information in the format that you need and
getting the right information put into that template.
Required
English

Help Us Translate

Describing everything in full English sentences or
full sentences in whatever language you speak isn't always
the easiest way to give the input to the large language model.
Now, if you think about it in the real world,
a lot of times we have our own specialized languages to describe different domains.
So a math may have its own specialized language or we may have shorthand for
taking notes or describing different interesting things.

We may create our own little languages to able to capture information in
a more concise way than if we wrote it out in full sentences.
Now, of course, the key to these languages is that the person that we
are communicating with in the language knows the language too.
They know the meaning of the different, the statements that we're making in
the different symbols or words that we're using.
Now as an example of this, we've been doing a lot of work in partnership
with the city of Nashville and the emergency responders.
And one of the things they do is the operators who are taking calls.
And you can imagine this is a really stressful situation.
They're trying to take notes down about what's going on in an emergency and
they actually have their own shorthand notation.
You can imagine the stress and difficulty of receiving a call or
around in an emergency and trying to take down information that can
then be used to take the appropriate action.
Send an ambulance or send a fire truck or whatever needs to be done.
And so they have their own little shorthand notation.
And if you go and just try to read this,
you're not necessarily going to be able to read it and
understand it because it's designed for people who use it day in and day out.
To be able to communicate with each other, but
not necessarily to be as explicit and obvious as possible.
So I'm going to teach you a pattern where we can
basically teach a large language model.
A new language and
then communicate with the large language model in that new language or
shorthand.
So this is called the meta language creation pattern.
So we're going to create a language, this is going to be our meta language creation
where we're going to go and describe to the large language model.
What our language is, what are the meanings of different symbols or
statements in that language?
And then we're going to go and converse with the large language model using it.
So let's look at our example here,
we're going to create a trip planning application.

Now, I haven't gotten into the metal language creation pattern, but
I'm giving you an example that uses it.
So we're going to create this trip planning application.
I will describe my trip and
you will list interesting things to do in the places that I will pass through.
I will tell you how many days I will stay in each place and
you will list possible itineraries.
So far, we've just got a prompt to help us do trip planning.
Now, here's where we're going to go and use the metal language creation pattern.
I'm going to say to describe my route, I'm going to use a shorthand notation.
So we're queuing it in and we're going to tell it about this shorthand notation.
When I say and notice this Nashville
comma three arrow Memphis comma two.
I mean that my route will go from Nashville to Memphis and
that I will stay three days in Nashville and two days in Memphis.
Now, I want to kind of draw your attention to something here.
Notice that I'm starting off by saying when I say this,
here's an example of my new language that I'm creating or
my new shorthand that I'm creating.
I mean this and so you're taking an example and
then you're showing it what you mean.
So you're then saying, I mean that my route will go from Nashville
to Memphis and I will stay three days in Nashville and two days in Memphis.
So you're giving it basically a translation.
Now another way to think about it, this is similar to few shot examples.
You're saying here's an example of input, here's the example of the output or
here's the example of a comment on a movie.
Here's an example of the sentiment analysis for that comment,
you're kind of doing something similar except what you're doing is you're
saying when I say this in my language, this is what I mean.
And you're showing it the translation and the meaning behind it.
The other thing to note is that my notation
is much denser than the full English that I'm going to use.
So it's Nashville comma three arrow Memphis comma two.
And it's that short thing is describing this much longer.

Basically statement that says my route will go from Nashville to Memphis and
I will stay three days in Nashville and two days in Memphis.
I could write all this out, but I want to have my notation.
Now it goes ahead and trip plans for me on Nashville and Memphis and I'm going to skip that and I'm going to go down to now where I have an actual just statement.
So notice we've now programmed our new language,
we've done the metal language creation pattern.
And now the large language model understands our language.
So we say Nashville comma zero and Nashville comma zero means
I'm leaving from Nashville, I'm not going to stay there.
I don't want you to trip plan for me in Nashville,
which is an interesting thing that has to consider that in the logic.
I'm going to Dallas, Dallas comma one and then arrow Grandberry Texas comma four.
And so basically, what I'm saying is my trip is going to go from Nashville.
I'm not going to stay there at all.
Dallas, I'm going to stay there one day and then I'm going to stay in Granbury, Texas for four days.
And it says sure,
since you mentioned that you won't be spending any time in Nashville.
I'll focus on providing you interesting things to do and possible itineraries for your one day.
Stay in Dash, Dallas and your four day stay in Grandberry, Texas.
So it then plans our one day stay in Dallas.
We have a one day stay in Dallas.
It describes what we're going to do on that day.
It then describes for Granbury, Texas over here, it says four days and
then it describes what we're going to do in Granbury for the four days.
Now, if I wanted to experiment with different itineraries and
I wanted to really quickly have a short end for saying, well,
what if I did this itinerary?
What would I do?
What if I did this one?
It's a lot more condensed.
Now, also I have to admit I'm a computer scientist and a programmer, right?

So I like having things that look like condensed programming language instructions.
But you can imagine a lot of people use shorthand notation, abbreviations.
You can use them too in the language model.
You just have to explain it to it unless it already knows it.
If you're using something that's well understood,
you can just say I'm going to use this language.
Or I want you to interpret this in the same way that you would interpret it in a math context and you could explain what that means.
So it plans our trip.
Then here's where I go and try it again.
Look, I'm getting a more complex statement.
I'm saying I'm going from Nashville to Fair Hope Alabama where I'm going to stay three days to Gulf Shores Alabama.
Where I'm going to stay one day to Montrose,
Alabama where I'm going to stay one day.
So now I've got a much longer statement.
And if I had to write that all out in words, in full sentences,
it would take a lot longer to write it out in full sentences.
But I've described the language and
now this kind of crazy string that I've put together.
It knows to take that and use it as the input for the trip planning.
So now it says Fair Hope Alabama three days and
it gives me my planning for my itinerary there.
So it gives me my day 1 day 2 and day 3.
It then says Gulf Shores Alabama one day and then it provides my itinerary for that one day in Gulf Shores and then it understands Montrose, Alabama.
And gives me my one day stay in Montrose and
gives me examples of things to do.
So you can imagine that this is a really sort of helpful capability.
You may want to go and start thinking about how do I develop a shorthand notation for the tasks that I do really frequently with the large language model.
And where it would save me a lot of time and
effort if I could describe what I want using some shorthand.
And maybe we can also describe it more precisely,

maybe there's a way to describe it more consistently.
So if everybody uses the shorthand,
the meaning is less ambiguous because we've already defined it in advance.
And there's less variability in the explanation or
how you express it in the shorthand.
So maybe you're an organization and
you want everybody to express certain information the same way, to make sure
that the reasoning from the underlying large language model is consistent.
And so now it's not always necessarily going to be consistent, but
at least you would get the data, entered in a more consistent way.
So you could define your own language for your organization for
describing certain types of information and teach that to your people so
that when they go in and they enter in that information.
You're going to have a consistent sort of semantics to what that information means.
Now, if everybody goes and just describes it in English using whatever phrasing,
whatever level of detail, etc that they want.
You may not get the outcomes you want from your prompt.
So by having your own language or shorthand,
you can help constrain how you define things.
How you express information to the large language model and
hopefully help it reason better.
Required
English

Help Us Translate
Sometimes we know part of the solution that we need when we're working with
a large language and model.
So we have a problem and we know some of the steps or some of the pieces that
we
want involved in the solution, but we don't know the full solution.
And what we need is the large language model to help us fill in the blanks,
to put together the missing steps.
Basically, to complete the recipe.
Give us the detailed set of steps that take the ingredients that we have or
the bits of information we know about how to cook this particular dish and
fills in all of the gaps.

And so the recipe pattern is a pattern for doing this.
We give the large language model some goal that we're trying to achieve and
we give it the parcel set of steps that we know that we're going to have to involve
in creating the overall sort of solution, the recipe, the goal, whatever it is.
We're going to give it part of what's needed and ask it to fill in the rest.
So this is really a way of getting the large language model to help us fill up,
fill in our gaps in a process or
gaps in our knowledge about a series of steps that need to be taken.
So the recipe pattern can be really useful.
And I'm going to show you that our sort of intuitive shorthand for
describing gaps and ways in that we talk about things that we know are part
of a process or part of a series of steps but
we don't know the exact details that our intuitive way of expressing.
That is actually something that the large language models like ChatGPT can
understand.
So let's take a look at an example of this.
So we're going to say we're going to add a feature.
So we're going back to our trip planning application and we're going to say, I
will tell you my start and end destination and you will provide a complete list
of stops for me, including places to stop between my start and destination.
So notice what we're doing is we're providing essentially
an incomplete recipe.
We're saying here's where I'm going to start, here's where I'm going to get to,
but I need to know what goes in between.
And it goes back and it takes one of our prior examples of going from or actually,
it creates its own example of traveling from Nashville to Atlanta and
notices that you're going to go through Chattanooga on the way.
Which is true if you're going to drive from Nashville to Atlanta, you're probably
going to drive through Chattanooga and it suggests things that you could do,
like going to the Tennessee Aquarium in Chattanooga or Lookout Mountain.
So where it gets interesting is if you remember back to the trip planning
application, I taught it, my own language for describing my trips.
And so I use the meta language creation pattern.
Now I'm going to combine the metal language creation pattern and
the recipe pattern to do something interesting.

So what I'm going to do is I'm going to say Nashville comma zero.
So I'm starting in Nashville and then I'm going to have just dots or
let's see, basically dot, dot, dot, arrow, dot,
dot, dot, arrow, Fairhope Alabama 2.
So I'm going from Nashville to Fairhope Alabama.
And basically, what I'm signaling to the model is I want you
to complete it, tell me where to stop in between.
And I'm indicating I want to stop twice.
I'm willing to make two stops in between, but I don't know where, you fill in
the recipe that gets me from Nashville to Fairhope and has two stops in between.
And it understands this, so dots are obviously dot dot dot is a natural
way that we can express something we don't know, it understands that and
it fills in the recipe.
It says we're going to start in Nashville, Tennessee.
No time spent as per your request.
I highly recommend spending time in Nashville, Tennessee.
I live here so all of my trips start with not doing any,
not staying in Nashville because I'm always here.
Well, not always, but a lot of the time, and then it fills in the two stops for us.
It fills in Birmingham, Alabama and Montgomery,
Alabama, which are both between Nashville and Fairhope if you take the most
direct route, and then the end is going to be Fairhope Alabama.
So you notice, it's basically understood the constraints that we placed on it.
We said, Nashville comma zero arrow.
And so we're using our language, we're just defining the starting point.
We're then saying dot dot dot, meaning this is something I don't know.
It's a blank spot in my template or my recipe.
So you need to fill in this recipe with where we should stop here.
And then arrow dot dot dot, and I'm saying,
here's another blank spot where I don't know what should go here in the recipe,
fill it in and then arrow Fairhope Alabama.
And so it's gone and we're using the recipe pattern here to get ChatGPT or
another large language model to complete our request.
And if we went and we did this again, so if we said, LA and let's just do one, and
then we said, dot, dot, dot, dot, dot, dot, dot, dot, dot and we'll

see how it interprets this because I think it's unclear to me what it even means.
And then we said we're going to go to New York.
Let's see what it interprets this as.
And so it now starts giving us a trip saying, okay,
here's where you're going to start Los Angeles one day.
Then you're going to go to Las Vegas, Grand Canyon, Santa Fe.
I'm curious if it'll, how many stops it's going to have?
So you notice that when I did this, when I had four placeholders here,
I'm getting more than four placeholders in between because I kind of expressed
that I wasn't really sure, like I don't have dot dot dot arrow, dot, dot,
dot, Arrow and then I'm there.
I just have a whole bunch of dots which is much more indefinite,
which is exactly what I wanted it to do.
Interesting to lay on off, up and I didn't know if it was going to get there, but
I was basically expressing, I'm going to go from LA to New York.
I'm going to have a bunch of stops in between.
I don't know what they should be.
Plan my trip for me.
Here's how long I'm going to stay in LA and here's how long I'm going to stay in
New York, and everybody can argue with how long I've spent in each place.
They're all wonderful places, but
it interprets the missing pieces of my recipe.
It probably also knows that I'm going to need quite a few stops if I'm going to
make this a leisurely drive from Los Angeles to New York.
And it fills in a whole bunch of different places that I could potentially stop along
the way in order to have an interesting trip from Los Angeles to New York,
if I run it on a very long road trip as a vacation.
And so this shows you how the recipe pattern works.
You give the large language model, the goal that you're trying to achieve,
the pieces that you know need to fit into the solution or
the steps that have to be there and where they're going to be placed, and
then you ask it to fill in the rest.
And you can use ellipses or other types of things to indicate places where you don't
know what's going to be there and you want it to fill it in, and it'll go and
complete the recipe for you using the recipe pattern.
Required

English

Help Us Translate

Brainstorming different approaches to solve a problem is
a really interesting way of
using a large language model
and one of my favorite uses for it.
Something that I really like about
large language models is that I can come up with
a problem that I want to solve and I can rapidly come up
with different ways of solving that problem
and I can test them out and get feedback.
Now, one way to do that is
using the alternative approaches pattern.
The idea behind this is if we're going to have
the large language model within some particular scope,
always suggest alternative approaches for
solving a particular problem
or performing a particular task.
That is, we always want it to give us good ideas.
Now, I say good ideas, they may not be good,
they may not even work,
but like brainstorming in a group.
You may have a lot of different ideas.
Some of them may be great,
some of them may be terrible,
but just the fact that everybody's
thinking through ideas,
throwing them up on a Whiteboard or
writing them down or talking about them
can help you do a better job of thinking through
the problem and figuring out what
is the right solution that makes sense.
I'm going to show you how to use
the alternative approaches pattern

to essentially perform brainstorming
with the help of a large language model.
Now one of the things to note is just like
if you get a group of participants
that have different backgrounds,
they may help you come up with ideas that you
yourself would not have come up with.
Well, a large language model was
trained on a big corpus of material and it's
probably seen lots of problems and
seen discussions of different solutions
to that problem or different alternative approaches
for a particular problem.
You can think about it. It is in some ways learned
information about how to
solve problems in different ways and
can help us get access to that knowledge.
Now, it may also have
never seen the exact problem that we're asking it
for and it may be able to
synthesize using what it seen in the past,
interesting potential approaches, even it
hasn't seen the exact problem that we're working with.
Let's start off with an example of this.
We're going to do the alternative approaches
parameter, and we're going to say from now on,
if there are alternative ways to
accomplish the same thing,
list the best alternative approaches.
This is a very generic wording for this,
it's probably better to structure it
very precisely to the task that you're working on,
provide more rich context.
But I'm going to show you that even within
a fairly generic wording,

we can get this pattern to work for us.
Then I'm going to say compare and
contrast the alternatives and
ask me which one I want to use.
I'm adding a little bit to this pattern.
I'm saying given the alternatives but
also give me some information,
compare and contrast the alternatives.
Maybe that'll help me think through
which one is going to better fit my particular situation.
I can see if what are the pros and cons of
each approach and which one is
a better fit based on those pros and cons.
I then ask if you think back to some of the past videos,
we've looked at using gameplay to
create prompts and test
us on our ability to write prompts.
One of the problems that ChatGPT
generated for us is it asked us to write
a prompt to distinguish leap years.
Basically, we would give it dates and
the prompt should be able to
determine if it's a leap year or not.
I'm going to turn that into a task for ChatGPT.
I'm asking ChatGPT to write
several prompts for itself to solve this task.
I'm having ChatGPT write the prompts for me.
Now, I could have introduced this earlier in
the course and just said any tasks that you have,
you want to be better at prompt engineering,
ask ChatGPT to do the prompt engineering for you.
That's basically what I'm doing here.
Sometimes this works, but I don't
want you to think that this is
the best approach that I should just always

go to ChatGPT and ask it how to do it.
Now, you should have a sense yourself,
just like any problem you're
bringing into a large language model,
you have to own the content.
You should have some expertise in
this area to help you
distinguish what are the better approaches,
what things don't make sense,
what's unsafe or harmful in some way.
You need to own whatever content comes out.
That's part of the reason I didn't just start
by saying prompt engineering,
go ask ChatGPT for
the prompt. We're not going to do that.
But in this case, we are. We're saying,
we want ChatGPT to write a prompt to
use few-shot examples to determine if a date in the year,
month, day format is
a leap year and the output should either
be year is a leap year or this is not a leap year.
ChatGPT then says, approach one direct question format.
In this approach, we directly
asked ChatGPT whether a given data is
a light leap year and then it shows us
the few-shot examples for doing this.
It's generating the prompt force that we could
use essentially to go
and check if something is a leap year.
Then the second approach,
it says conversational format and then it uses
a completely different style.
It's still roughly the same format
of generating few-shot examples to solve this problem.
Now notice I scoped it to few-shot examples.

I could have left the scope more
wide-open and just say write a prompt to do this.
But I thought it'd be fun to look at different.
Can we write the few-shot examples in
different styles and will it make a difference
and we could go and test this and see if it does.
We now see it generating, again,
it's solving the problem of showing
few-shot examples for leap years.
Now, I noticed something interesting.
It's showing us how to solve the problem.
It didn't actually fully follow
the format that I gave it because I said,
why is a leap year or this is not a leap year
in it is putting the date, the year at the front.
It's actually made a mistake
and this is part of the reason
why you should read your prompts that it generates.
Make sure they're doing exactly what you want.
But I could go and retell it and fix it and refine it.
But you notice now I've got two different approaches to
few-shot examples to solve this problem.
Then at the end,
I get a comparison,
it says approach one is more formal and structured,
making it easier to understand and parse.
It may be suitable for applications where
a clear and concise answer is required.
Then it says, approach to his more conversational,
natural making it more engaging for users and
maybe suitable for applications
where user interaction is important.
Which is kind of true if you're going to prompt
the user and then have like,
here's what the user said,

here's what the assistant's supposed to say.
This would probably make sense if you
were just reporting on a Q&A,
then maybe the other format makes sense.
It's a reasonable discussion
of the different alternative approaches
for solving this problem.
Now, let's look at another example.
I thought, what is
a task that can sometimes be burdensome?
I thought about email, now I haven't
actually done this and tried it out on my email.
I still keep my e-mail personal,
if I can to my laptop,
but I thought that this would
be an enhancement in the future if I
was running a large language model directly on my laptop,
something like Lama, Alpaca.
Maybe I could do this and I would be comfortable with it.
I say whenever I asked you to write
a prompt for me to accomplish a task,
I'm going to change
the alternative approaches pattern slightly.
I'm going to tweak it a little bit for this one.
What the task is,
list alternatives for completing the task,
and then write a prompt for yourself for
each approach so some basic format.
Now I'm doing two things.
One is I'm having it lists the task at the beginning.
Right here I'm saying list the task and then I'm saying
list alternative approaches for
completing the test so this is going to be upfront.
One, it helps me know
what it's approaches are

going to be before I even see them,
I can see a preview,
two this can actually be
helpful for large language models if you
hadn't kind of summarize the task and
summarize different approaches or reasoning.
This is been shown to be helpful in many cases.
I'm actually asking it to
do that at the beginning just to
show you how you can tweak the pattern and
get some possible improvements.
In this case, it does
roughly the same thing that we saw before.
Then I asked me what I want you to write a prompt for.
The prompt I ask it is,
I say write a prompt for ChatGPT
to automatically detect questions in
a chain of emails and summarize
everyone's questions as bullet
points beneath each question.
The idea behind this is
really this key part is at the end,
you can imagine getting behind
an e-mail and you want ChatGPT,
or some other large language model that you
trust looking at your email,
that you put it in the whole email conversation chain,
assuming that you are following
appropriate privacy and other
rules around whatever content
is discussed in that email chain.
Maybe you start with something like
what movie or what
restaurant people want to go out to eat.
When you have a group discussion.

You get to these long discrete group discussions
and they've gone off the rails.
People have discussed other things thrown in jokes,
commented on other people's jokes and by the end of it,
you can't really decipher who
wants to go to what restaurant and why?
Because there's all this other random stuff there an end.
You might go and say summarize
the questions and people's opinions
on it and it'll give you maybe just the discussion about
the restaurant and people's perspectives.
You would dump this end and I say
write me a prompt to do that,
and it gives me three different approaches.
One, write a prompt that instructs ChatGPT to first
identify all questions in the email chain,
and then extract responses related to each question,
and finally summarize the opinions as bullet points.
Approach 2, write a prompt that asks ChatGPT to simulate
a conversation where it plays the role of a summarizer,
and notice it's basically using
the persona pattern here in its own prompt for itself,
asking the user to provide the email chain,
and then generating the summary with bullet points,
and then finally it says write a prompt that
presents the email chain as a case study.
That's an interesting approach,
I wouldn't have thought of that.
What's interesting is a case study,
it probably read lots of case studies,
so it's probably good at thinking about how
do you think through a case study,
and ask ChatGPT to provide a summary report,
that includes the questions and
the corresponding opinions in bullet points.

Then it gives me each of the different prompts.
ChatGPT, your task is to
analyze the following chain of emails,
and identify any questions that were asked.
For each question please summarize the opinions
expressed by the participants in
the email chain as bullet points beneath the question.
It's generated its own prompt for itself.
Prompt Approach 2, ChatGPT,
let's simulate a conversation where you play
the role of a summarizer and it goes on to discuss it,
and then it has a slightly different format
down here for how it wants things presented,
and then finally, prompt for Approach 3,
ChatGPT, consider the following chain
of emails as a case study.
That's really interesting,
and a completely different approach.
That what I just would
not have thought of from the beginning.
Now, maybe if I went to business school or something,
I would be more thinking of case studies,
and this is a natural approach.
But this is something new to me and so
I can learn from how to write prompts,
and now I have three completely
different approaches that I could go
try out without having to think of them myself.
I've also learned something, maybe case studies and
presenting different points of
view on an idea as a case study,
is a good way to get output from ChatGPT.
Now, what would I like to do next?
Well, I'd like to take my alternatives
and I would like to evaluate them some way,

and one way to evaluate output,
particularly when it's just text
and you want to figure out how good does it look?
Does it seem to solve the problem?
Is just to use the large language model
to evaluate its own output,
or to evaluate the output
of another large language model.
Now, usually you want to do this
in a separate conversation.
You don't necessarily want to tell
the large language model, now evaluate yourself.
Sometimes that leads to worse behavior
where it just says, oh, I'm right.
Sometimes what we want to do is take that into
a separate conversation and evaluate it,
and I'm basically asking it,
how would you write prompts to take
your own output and evaluate?
How would you write prompt to evaluate the prompts?
So I say write a prompt to evaluate
several different prompts for summarizing
the questions in an email conversation,
and the different stakeholders
perspectives on the questions.
Then it gives me three alternative approaches again.
Approach 1, write a prompt that
presents multiple prompts to ChatGPT,
and ask it to evaluate
each prompt's effectiveness in
summarizing questions and stakeholders perspectives.
Approach 2, write a prompt that instructs
ChatGPT to simulate a conversation
where it plays the role of an evaluator,
and the user provides different prompts for evaluation,

again using the persona pattern,
and then write a prompt that asks ChatGPT to
provide feedback on each prompt
and suggest improvements if any.
There's a different take in each one.
The last one is really about how do we improve each one,
second one is using
the persona pattern act as an evaluator,
and the first one is really to evaluate effectiveness.
They're different takes on the same idea,
but now I have three completely different ways that I
could go and take my prompts that it generated itself,
and look at and evaluate them using ChatGPT itself.
Required
English

Help Us Translate

When we're working with large language models,
often we're going to describe in our prompt a set
of rules that we want the large language model to follow.
Rules that are going to affect its output
and essentially program the large language model.
One of the things that we're going to want to
do, we're going to want to say,
here are the rules,
now apply them to
an input that I'm going to provide later.
One of the challenges is,
how do we get the large language model not
to immediately respond to all of our rules?
If we have a flipped interaction, for example,
sometimes when we flip the interaction and we say,
ask us questions, it'll generate
10 or 20 questions all at once.
A way to do is basically a simple way that
we can get the large language model

to listen to what we're saying,
acknowledge it, and then wait for input from us.
Now, we can do this with the ask for input pattern.
The idea behind this is we want to get
the large language model to listen to
the rules and then ask us
for the input to apply the rules to.
When we're talking about
asking questions in a flipped interaction,
we might want to describe
the rules for asking the question,
and then we just wanted to ask one question.
So we just want to get one question out.
If we're describing rules for alternative approaches,
we might want to say, here are the rules.
You're always going to create alternative approaches
to whatever asked you to do,
but just ask me now for
the first task to create alternative approaches for,
we don't want you to go and have a big response to it.
What we see is that that's not always the case.
Let's take a look at this example.
Now, in a prior video,
I introduced the alternative approaches pattern,
and when you saw me use this pattern,
it worked really effectively because I was actually
combining it with the ask for input pattern.
I'm going to show you what it looks like when I
don't combine it with the ask for input pattern.
I went and I said,
whenever I ask you to write
a prompt for me to accomplish a task,
list what the task is,
list the alternative approaches for creating the task,
and then write a prompt for yourself for each approach.

I'm asking in ChatGPT,
give me a bunch of alternatives for prompting you,
ChatGPT, to solve the task.
This is going back to the earlier idea
where I showed how we can actually use
ChatGPT to generate prompts
for itself to help us with prompt engineering.
What I do is I use this,
but I'm taking away something
important that I'm going to show you in a minute.
What it does is ChatGPT responds,
it says certainly, but then it dreams up a task.
It says, here are three alternatives for completing
the task organizing a cluttered desk.
I'm not interested in
the task organized a cluttered desk.
It just responded to my initial prompt,
which was, create alternative approaches for a task.
It dreamed up a task and then dreamed up
its own approaches for that task.
Now, I'm not interested in this,
so now I've got my conversation and it's got
all this extra stuff in it that I don't care about.
I had to wait for it to generate it.
If I was trying to create
some interactive experience that had some consistency,
this would not be helpful.
What I can do is I can apply the ask for input pattern.
Then the idea behind this is
we just do one simple thing,
and at the end of the prompt,
we tell the large language model
to ask for the next input.
In this case we say, ask me for the first task.
The very last thing we do,

we describe all the rules,
or whatever it is that we want in our prompt,
and then the last thing we do is we say, now,
ask me for the first task,
ask me for the first idea,
ask me for the first thing you want
me to generate a story about, whatever it is.
But it's a way that we can cut
off the large language model and tell it,
don't go generate a bunch of stuff.
I'm glad that you can do that,
but what I want you to do is I want you to stop right
here and asked me what I want to do.
It's a way of forcing
the large language model to basically
turn the control back over to
the user and let them decide what to do next.
Now, like all prompt patterns,
when we're dealing with a larger language model,
it is somewhat stochastic or random.
It means it's not always going to do
exactly the same thing every time.
It may not work 100 percent of the time,
but this works very, very reliably to just say,
ask me for the first input or
whatever it is at the end of
the prompt because it
clues the large language model and it needs to stop.
If we look at the comparison of the output,
it's actually much more focused.
We see, in this case it says,
certainly, let's begin.
What is the first task you would like me to create
alternative approaches and prompts for?
Rather than this much longer saying that it generated

before that was not
really helpful to what we were doing,
so it cuts it off.
Notice it's the same input,
but it cut it off.
It's told that, look, you need to
stop right here and you
need to wait for my input.
Required
English

Help Us Translate

One of the things you may have noticed
that I've started doing,
is I've started combining patterns together.
This is going to be one of the most
important concepts when you want to
build really sophisticated prompts.
Is you need to think through how
to apply multiple patterns
in conjunction with each
other in order to
solve the problem that you're dealing with?
Rather than just thinking about
which pattern can I use to solve this problem?
Instead, what you want to think about is
which pattern will solve certain aspects of the problem?
And how would I combine those patterns together?
We started to see an example of this with
asked for input and alternative and approaches,
asked for input as a pattern that can
be applied to a lot of other patterns.
In this case, we're applying it in conjunction
with ask for their provide alternatives,
and so we have got alternative approaches.
And we've got asked for input

and they're being put together
to build a prompt that solving the overall problem.
Now, what we're going to see throughout the remainder of
this course is that this is
the way we build really sophisticated prompts,
is to think about taking
multiple patterns and combining them in interesting ways.
Now, if you think back to some of the patterns that
I introduced in the course and some
of the examples I gave.
Now that you know more of the patterns,
you'll probably see that I've actually been
combining patterns all along in different examples.
In gameplay for example,
I probably used asked for input
and in gameplay I use probably the persona pattern.
So you can see how putting things
together helps us solve the overall problem.
When you're going and you're trying to tackle
some new complicated prompt
that you need to engineer and develop,
you want to think about, what
is the fundamental pieces to this problem?
And do I have existing patterns that I know
about that I can use to hopefully tackle all of it,
but if I can't tackle all of it,
can I use patterns and apply
them to as many different pieces as possible so
that your unknown piece that you really have to
start from scratch and invent something new?
That it's as small as possible,
that we can start with as many pieces as possible with
a roadmap for them a pattern to
apply. That's what we're going to be doing.
Now it may start to seem more complicated than it is,

but at the end of the day,
what we're still doing is we're
still building up language.
We're still writing in natural language,
we're still writing sentences
, we're still communicating.
We're just thinking intentionally
about the patterns in our language.
And we're thinking about what are the statements in
the organization of this statements
in order to solve the problem.
In some ways, combining patterns is
really doing a more powerful version of programming.
We're starting to take
different programs are
instructions that we know how to use
and put them together in
more complex and sophisticated ways
in order to have prompts that have more power to them,
they have more depth to them.
Again, don't feel that this is something you can't do,
everybody can do this.
At the end of the day, think about it.
You're just describing in words what you want done.
You're still describing in
language what you're trying to accomplish.
All this is doing is helping us think
about what should go into the instructions.
If we're going to use this pattern,
it's giving us a set of commands,
a set of statements that we need
to include and an organization for them.
If we're going to use this pattern,
we're going to have a set of
statements that we're going to include.

Oftentimes, putting things together
is simple as just taking
the statements from each pattern and
making sure they all show up in the prompt.
Sometimes it's that simple,
sometimes it's more nuanced and we made need to make
sure a certain statement ends up in a certain place.
Like what they asked for input pattern,
we probably always want
the ask for input to show up at the end of the prompt,
because otherwise it doesn't work nearly as effectively.
Although putting patterns together
can be a little bit tricky sometimes
if you have to figure out where things should go,
most of the time,
it's as simple as putting the statements together.
As we go through the rest of the course,
what I would encourage you to do is to notice when
I'm using multiple patterns together to
solve a problem and to think about as you're
trying to tackle new tasks with prompt engineering,
how to use multiple patterns in
conjunction with each other to solve the problem.
Required
English

Help Us Translate

There's a limit to how big we can
go with the large language models.
They fundamentally have a limit to
the size of the input we can give them,
so how big of a prompt we can give them or how long
our conversation can be and they can still remember it,
and there's just a limitation
on how much output they can generate at once.
Now, it's inevitable that

these language models are going
to keep getting bigger and bigger
and we're going to be able to give them
more and more input and get
more and more output from them.
But more than likely, everybody is going to face
some limitation on how big we can go.
Now the way that we work around that
is we're typically generating pieces,
we've generated a piece of the solution, then we take it.
You can imagine, if you're trying to generate a book,
you can't just tell ChatGPT at this point,
"Go and generate a book for me."
It's not going to do it, it's going to tell you,
"Well, I have to stop my end point at some point."
Similarly, you can't just give it
50,000 pages at this point and dump it
into ChatGPT and ask it to give you an answer
or 100,000 pages or a million pages.
There's always going to be a limit
probably on the amount of information.
We need a way to work with it in smaller pieces,
but still be able to put all the pieces back together.
One of the ways that we can do
that is by building an outline
of what we're trying to put together
and we can build the outline out,
we can build it to the level of
granularity that we want,
and then we can start filling
in the pieces once we've gotten
enough detail into our outline,
and then we can go and generate
individual pieces and we can know
where they all fit together.

This is an effective way of using
it to generate the pieces.
Now this is basically putting a plan together.
This same approach works for all kinds of things.
It could be software,
anywhere where we can go and
have this outline-like structure,
and then if we generate something that
fits the outline or the plan at that point,
then we can go and work with it.
Now we're going to look at a specific pattern that can
help us generate the outlines and work with them.
This is called the outline expansion pattern.
The idea is we want to make it really
easy to generate outlines and continue to
expand them down to the level of granularity that we
need in order to build
the individual pieces then be able to put them together.
One of the reasons that outlines are
a really helpful tool when we're building
bigger bodies of work with a tool like ChatGPT,
is that we can
always take parts of the outline and copy and
paste or reintroduce them into the prompt to
give the large language model context about what we want.
With ChatGPT, for example,
we can generate an outline and then we can copy and paste
a portion of the outline into
our prompt and say within this context,
where you are in the outline,
now fill in the content for this and don't
repeat anything that would have come
before in the outline and in fact,
you can refer to the other parts of
the outline wherever you're going to talk about

a topic that was discussed
previously or maybe we want to have
forward references where we can
provide at the high level outline and say,
"Wherever you are going to need to talk about
a topic that's going to be discussed later,
you can provide a forward reference
to where it is and the outline."
You can do similar types of things with software
or whatever the bigger thing is that you're building.
Now, this works well when we have
a structure that's an
amenable to this outline-based plan.
Now sometimes we have plans,
or structures, or designs where
everything crosses over and you can't easily do this.
It doesn't work to express it as an outline and
express the plan and in those cases,
it just doesn't work.
But many problems,
we can build out in
this outline-like fashion and then we can
use cutting and pasting from
the outline to provide the appropriate context.
Building outlines can be a really powerful thing for us.
We're going to look at and
build with the outline expansion pattern.
Here's what we say, we're going to say,
"Act as an outline expander," and notice,
I'm starting to introduce multiple patterns here.
I'm basically introducing the persona pattern.
"Generate a bullet point
outline based on the input that I give
you and then ask me
for the bullet point you should expand on."

We're telling it you're going to generate
the outline and then you're going to
expand on anything that I ask you to expand on,
create a new outline for the bullet point that I select.
At the end, ask me
for what bullet point to expand the next.
Notice, I'm using the tail generation pattern as well,
and then I'm using the ask for input pattern.
I'm putting a lot of things together
into this prompt and I say,
ask me for what to outline.
What we see is it says,
"Sure, I'd be happy to help you with that."
I say, let's write an outline
for writing effective prompts for ChatGPT.
What we see is that ChatGPT generates
this outline and it's an interesting outline introduction
to ChatGPT,
understanding the importance of prompts,
tips for writing effective prompts.
yada, yada, yada. A lot of good things there.
Then I decide, let's go and
expand on three from the outline.
If we look back here, three was
tips for writing effective prompts.
I thought let's go ahead and expand the outline
there and explore this a little more.
Then it picks up from three and it expands
on tips for writing
expected prompts and gives us a new outline.
We build and using this pattern,
we can rapidly fill in and outline, and probably,
there are going to be tools to
support us in this very soon,
or plugins to support us in this very soon

to where we don't have to
remember all the pieces of the outline.
With this, I would probably just manually have
a document whereas it was generating the pieces,
I would go and look and say, here's three.
I'll paste three and its expansion
now and you can easily fit the parts together.
This is one of the nice things
about the outline approach,
is that it's really obvious
how all the pieces fit together.
I know that 3.1 goes underneath 3,
so it's self describing in terms of the assembly.
If I generate a piece of an outline here,
I've expanded three,
I know exactly where that goes,
how it fits together in the overall whole.
Then I go and say, well,
this idea of experimenting with
different prompt styles, that looks interesting.
Maybe I should go and expand that further in the outline.
I basically say 3.2,
and now it goes and expands the outline around 3.2.
You see how we've got a simple pattern,
and we now can go and rapidly
create an outline and keep
expanding into the level of detail,
and because of it's using this outline structure,
it's really easy to fit all of
the individual pieces back together.
As long as we have limitations on how much input,
how big of a prompt we can give,
and how much output we can get it once,
we're always going to be looking for ways to
generate things in pieces and know how they fit together.

We want the pieces to be really
obvious on how to assemble them and we always
want to make sure that we have a clear structure or
set of guidelines for
each individual piece so that we can assemble them.
You can imagine if this part of
the outline depends on
knowledge of what was
covered in a different part of the outline,
then we would need to make sure
that that was introduced into the prompt.
We just said 3.2,
but if we had an idea that
maybe the conversation had gotten so
long that what was in item 1 had been forgotten,
we might want to go and just cut and
paste the outline underneath one back into
the prompt and then say expand
3.2 because we know 3.2 was just previously generated,
but that would help give it
the knowledge of what had come before and make sure that
what it put into 3.2 was
consistent with what had been done
earlier in the overall structure.
Similarly, if it needs to make sure that
3.2 was cohesive with what's going to come later,
we would want to put in those pieces.
One of the things we have to think
about is even if this outline style,
when we do this, it will work.
But we always have to think about,
does it still have the context to make sure
the individual pieces are cohesive
even if they fit into the outline at this point,
we want to make sure it doesn't do something

that breaks something somewhere else.
This is a problem, a fundamental
challenge that we're always
working with and prompt engineering
when we're building from pieces.
How do we build the pieces
so that they're standalone, self-contained,
but can be assembled into
the overall whole. We see this again.
We just expand 3.5,
which was setting the tone and style,
and we get this very rapid expansion
in creation of a sophisticated outline now that we
can use to write text for
a bigger project to be able to try to write software.
There's all analogies or analogues to the outline,
the master structure where we start with
a certain level of detail,
then we pick some piece of it and we
expand the level of detail.
Then we may pick a piece of
that and we may expand the detail.
Then we may move back up to
another bullet pant point in the outline
and expand the level of detail.
We can be moving up and down in the outline,
we can be going up to the top level,
or we can be going down to the really detailed things,
and we can expand wherever we want and keep a map of
the overall picture that we can use to build
individual pieces and make
sure that we can assemble them.
But also as we build pieces,
that we can make sure that
ChatGPT has the relevant information.

This section of the outline, for example,
in the prompt that it needs in order to
reason about or generate
whatever it's going to produce next.
Required
English

Help Us Translate

When we're working with computer software,
one of the things that's really
useful is that programs typically have
some menu with a series of actions that we can apply.
If you've ever used word processing software,
there'll be a file menu that allows you to run
an action to open a new file or save a file.
You may have a menu that allows you to
search and replace different pieces of text.
This idea of having a menu in different actions
that we can run is a really useful concept.
Sometimes when we're working
with prompts and prompt engineering,
and particularly when we have a long conversation,
it can be helpful to develop
the equivalent within a prompt.
Basically define essentially a set of actions that we can
run and order to
work in the conversation without having to
repeat these long prompts,
or we want to just codify
certain actions that are helpful to
people and bake them into
a prompt at the very beginning of the conversation.
Let me give you an example of this.
We're going to look at using the menu actions pattern.
This is a pattern that allows us to get

the same type of behavior but in prompts.
We're going to develop a menu that is
a series of actions that we can run.
Actions are basically essentially a prompt.
It's basically a set of
instructions that we're going to give
the large language model into order to do
something at a particular point in the conversation.
Here's how this works.
We describe what we're going to
type in order to trigger the action.
I say whenever I type,
write bullet point paragraphs,
and let me talk about this in more detail in a minute.
Don't worry if this seems
just doesn't quite make sense yet.
Whenever I type this phrase, write,
and then I haven't less than,
greater than bullet point and then
paragraphs, these are placeholders.
Basically I'm saying I'm going to type
in write and then a placeholder for a bullet point,
and then a placeholder for a number of paragraphs.
You will write content for
the selected bullet point, bullet point.
Basically what I'm saying is I'm going to give you
a bullet point that
you need to go and write the content for.
Then I'm going to tell you
how many paragraphs of text to write.
Then you're going to go and write the text.
Basically I'm developing an action winches
before all I could do is expand my outline.
When I talked about the outline expansion pattern,
I talked about being able to rapidly expand the outline.

Now what I'm going to say is that at
some point I'm going to be able to tell you
no go in and turn that into
text rather than just outlined.
I'm creating a new action.
I'm also going to go ahead and keep my original action,
which is just expanding the outline.
I say whenever I type
and I have a placeholder for bullet point,
you will expand that bullet point.
What I'm saying is if I just type in
3.5.2 which is a bullet point
like a number in my outline,
it will know that it means expand that in the outline.
If I instead say, write which is a keyword,
and then a bullet point and a number of paragraphs,
it knows that that means it's
supposed to write the text for it.
Undefined in a menu, I now have two actions.
I can either expand my outline or I can
write the text for part of my outline.
Now I'm developing more sophisticated capabilities
for my outline expand,
I'm turned it into a real program
now something that I can interact with.
Let me show you an example now here's my
first action being run.
I'm saying write 3.5.32.
Now what that means is write
the content for the bullet point that was labeled 3.5.3.
Labeled 3.5.3, you're going to write the text for it
or for that part of this section in
the outline and you're going to write two paragraphs.
What we see is it takes
the bullet point 3.5.3 from the outline,

which was experimenting with tone and style variations.
It then writes two paragraphs
of text for that bullet point.
I've created an action,
I've defined it,
I've created a menu of actions,
and I've defined it in
my prompt and now I can
run that action and get the text generated.
Then we see it prompts
this again, what do we want to do next?
This is actually not part of the texts that wrote.
It did write two paragraphs
then asked me what I want it to do next.
Then now I'm going to go back and I'm going to do
exactly what I did before.
I'm going to say expand on 3.5.2.
This is what I did before and now it knows
the difference and it knows that's a different action.
It goes and expands
3.5.2 and puts the outline for it again.
Then go down and I say, okay,
give me text for 3.5.2,
I notice something interesting.
I'm actually making a typo here,
but it's being smart
and inferring what I'm supposed to do.
This is an interesting thing,
it tries to help you out, tries to get things right.
It knows based on the pattern that what I
probably meant was write 3.5.2,
instead I wrote text.
It's a Smart menu action system.
Then it actually goes and it pulls on
that bullet point 3.5.2.3.

Sorry I said that wrong,
it's 3.5.2.3 and then it writes one paragraph of text.
I've created basically a complex program
where I can expand my outline if I want to,
or I can have it write the text for part of the outline,
but I don't have to repeat myself.
I get to use this simple little format.
I've explained what the format is
and now I can re-run the actions whenever I want to.
One way this is useful to me is,
if you're doing the same task over and over with prompts,
it can be helpful to use
the menu actions pattern to basically
define a set of actions on
a menu of a prompt that you use all the time,
and then whenever you want to iterate really
quickly using that content and those set of actions,
you can just introduce that
menu actions prompt and then go and use
those actions over and over on the output that you're
getting in order to manipulate it or do something.
You've basically built yourself
your own program within the conversation.
At the same time,
sometimes an organization may
have a number of people that are doing
the same job and they can share actions
with each other to help automate different tasks.
This can be really useful to
help people be more effective with
their jobs and to share with each other useful prompts.
The other thing is, is you don't have to worry so much
about remembering exactly how you worded it.
Once you've written the action
once and gotten it to work in the menu,

then you can reuse it over and over without worrying
about not quite getting it right when you do it.
Then that leads into a final
thing that's really useful about
the menu actions pattern is that within an organization,
you can share a set of
common menu actions that everybody should use and you can
codify the best wording for
those prompts and for
those actions and what they should do,
maybe a codified best practices into them,
and then share them across the organization.
It allows for knowledge sharing and reuse of
different prompts and
then easy application of those prompts.
You could also imagine baking these
into a root prompt for something like
a customer service portal where you want to give
people certain actions they can
run to look up information about their account.
They could do it in natural language,
but you could also describe
here are some actions you could
run and allow them to get better behavior out of it.
Play video starting at :7:58 and follow transcript7:58
The menu actions pattern is a really powerful pattern
that allows us to easily reuse
other prompts and actions
that we've used to share them with others and
develop knowledge bases of them so that
we can become more effective prompt engineers.
Required
English

Help Us Translate
We've heard over and over in the news that

these large bank language models
make all kinds of mistakes and
can lie to you and they'll never admit
their mistakes in the first place nail
to keep telling you they're right
and produce weird outputs.
Of course, this fundamentally
misses the point of the tool.
These aren't meant to be tools
for answering questions necessarily.
Now sometimes they can answer
questions and do it very accurately.
Sometimes they produce things that look like answers
but aren't actually accurate.
As long as we understand they're really
for generating text and
sometimes the texts they generate can answer things
for us correctly and sometimes
it's going to make mistakes.
As long as we realized that we
can use these tools effectively.
Now one thing we can do is we can use
prompts to support us in our identification
of what are the facts that
the tool is putting into its output and which of
those facts really matter
and we can go in then follow up on them and check them.
Now one of the things that happens
sometimes when we're working
with output from these tools,
is that because the output looks so realistic,
so humans so convincing,
we assume that it's true and when it's
producing large volumes of text,
it can be hard to spot

all the different assumptions and facts that are
baked into the text that we
really need to follow up on and back check.
Anything you get out of one of
these models you need to take ownership of
and make sure that you've
checked it thoroughly and really believe it.
What do we do to support us in this effort of
trying to make sure the output is correct and factual.
Well, we can use the fact checklist
pattern to help us figure
out what facts are actually
in the output and need to be followed up on.
I'm going to give you an example of this.
I've taken our outline expansion tool.
If you remember, we used a couple of different patterns,
including the menu actions pattern,
we use the outline expansion pattern.
We had several different capabilities.
One of the things we could do is we could
go and expand the outline.
We also added the ability to be able to generate
texts for any bullet in the outline,
so and we can generate for a number of items and we
had a whole menu system to be able to do this.
Now one of the things we'd like to be able to do is
when we do generate texts for part of the outline,
I'd like to know what I
need to follow up on and fact check at
that point in order to make sure that I don't
present information that's inaccurate.
If we look at this,
the way that we can do this as with
the fact pack check list pattern.
We say, whenever you output text,

generate a set of facts that are contained in the output.
The set of facts should be inserted at
the end of the output and you
can put them wherever you want.
I'm just putting them at the end.
The set of facts should be fundamental facts that can
undermine the veracity of
the output if any of them are correct.
I'm setting up a new set of
instructions for the large language model that says,
anytime you produce text,
give me the corresponding set of
facts that the texts depends on.
That's the way I can go and follow up on them and check
them and so it says that understands.
Now I'm going to use our
outline expansion program that we've
written them and tell it to write a paragraph for
Section 3.5.2.2.
This section is considerations for choosing
the appropriate style and it
generates a whole paragraph of text for me.
Then what it does is it generates a set of
fundamental facts down here that are the facts that
are contained within the paragraph
and so the facts are for
example writing style can range from
technical to creative to instructional.
There are other things like instructional content
may adopt a clear and step by step style.
Now this paragraph that I chose for it to write,
it has a lot of different facts
that maybe they're correct, maybe they're not,
but it's not something that I have to worry as much
about going in extracting and just staring

a text because I now have
a list of things to follow up on.
At a minimum, I need to look at
this list and follow up on
these items or just actually
visually check them myself for my own knowledge,
that's the minimum level.
Now I have a minimum bar,
at least these things I need to go and
follow up on and it's flagging to me,
here are some things that you need to go and check.
The other thing I can do is I can compare
this list to the paragraph up here.
Any person that's probably
using one of these tools can compare.
Here's the list of facts below.
Here's the actual text
above is the list of facts complete,
are there statements in the text above that are
missing below and if there are,
those are probably things that I would
want to go and follow up on as well.
By checking and comparing the two lists,
one we can see is it really
fully extracting all of the facts that are
important and that in itself will
clue us into and make us think
through what is this
really saying and what are the fundamental facts to it.
Just the action of
going and comparing the facts to the text it's
produced will help us get us into
the mode of thinking
through the correctness of what is produced.
The next thing I can do is I have a minimum bar,

at least I need to go and check up at the very minimum,
I need to go and check the list of facts that it
produced and follow up
and make sure that they are correct.
If any of them is incorrect,
then probably there's an error in the text above.
Then if I notice that there's something in the text above
through this comparison that's also
not present in the facts,
I could add that to my list and go check it.
This is a tool that can help us get
into the mode of fact checking the output.
It may be correct,
but we still don't know for sure
until we actually fully follow up on it.
This gives us some tools to help us
one know that there are things to follow up on,
at least follow up on these things,
and then two, to help us give
us a tool that we can go and compare to what
it produced to see is
the minimum list of things
that I followed up and good enough.
Or they're all other things
that are in that text that maybe I
should think through or follow up on.
Now, it doesn't guarantee that we're going to come
up with the right answer.
We might think that it's still accurate when it's not,
even if we follow up on those facts,
there may be some hidden assumptions
but it's much better than if we just read
a bunch of text with no support
or stimulus to think through the fact checking process.
The fact checklist pattern is

something that helps us get us into
the mode of checking the output, checking its veracity,
and having a format that makes it
easier for us to extract and
compare the information and the such a facts
with what's really true using other external sources.
Required
English

Help Us Translate

When we're having a long conversation
with a large language model,
sometimes we'll run into the challenge that
the large language model will
forget the rules of the game.
It'll forget what we're trying to accomplish.
What is the task that we're working on?
One of the things we need is a way to remind
the large language model, what are we doing?
We'd like it to remind itself if possible.
The way that we can do that is by using what
we call the tail generation pattern.
The idea behind this is that we're going
to generate a tail,
an end to the output that
the large language model produces
that will help remind it what we're doing.
The tail is going to always be
generated at the end of its output.
The tail will remind
the large language model what the rules of the game are.
We're going to see this in the context of
combining the tail pattern
with the alternative approaches pattern
and with the ask-for-input pattern.
We're going to put all these things together and

we're going to see how this is a really useful thing.
Generating a tail on our conversation using
a tail generation pattern is really
effective in maintaining our conversation
longer and maintaining the rules
and the knowledge of our rules longer.
I've actually used this several
times and I haven't talked about it,
but I'm going to show you what it does.
This is the prompt that I used in
the alternative approaches pattern discussion.
I'm asking ChatGPT to
generate a set of prompts to solve a particular task.
I'm asking ChatGPT to help us with prompt engineering.
I'm going to give it a task and it's going
to generate possible prompts
that we could use to perform that task with ChatGPT.
Now, if I do this over and over,
and I keep saying anytime I ask you for a prompt,
you should go back and generate
multiple alternatives for it,
the problem I'm going to run
into is that eventually it's going to
forget that it's supposed to generate a bunch of
alternatives and then it's supposed
to be generating prompts.
I need a way that keeps re-introducing
that rule into the conversation to keep it alive longer.
One way I can help do that is with the tail generation.
This is the tail that's being generated.
I have the prompt and I say,
when you are done,
ask me for the next prompt to create alternatives for it.
I'm adding this on.
I'm saying when you're done at the end of your output,

you're going to ask me for
the next prompt to create alternatives for it.
Notice what I'm doing.
I'm actually using the ask for input pattern.
I'm saying at the end of the output that you produce,
you're going to ask me for the next input.
You're going to ask me for the next task to accomplish.
I'm telling it, you're going to ask me for
the next prompt to create alternatives for.
I'm also re-introducing the rules of the game.
I'm describing what the purpose of
the input is that I'm going to be giving
it and what it's supposed to do with it.
I'm making sure that this shows up at the end
of the output that
the large language model is going to produce.
I've defined the rules and I'm saying,
at the end of what you do,
you're going to re-introduce
this rule by basically asking
me for the next input to create alternative prompts for.
Let's see what happens.
I add this tail on and so I'm generating a tail.
I'm using the tail generation pattern,
and I say write a prompt
for ChatGPT to automatically detect
questions and a chain of emails and summarize
everyone's opinions on the question as
bullet points beneath each question.
I've given it the task. Here's what I want you to
write me some possible prompts for.
I would give it the instruction.
It's now going to write the prompts and it
generates a bunch of prompts for me.
Now I've got all these alternative prompts.

Then the last thing it does is it generates
the tail just like I said.
It says, please let me know if you
have another task for which
you would like me to write a prompt
and create alternative approaches for us.
Right here, it's created a tail on its output.
It's at the very end.
It says, please let me know if you have another task for
which you'd like me to write
a prompt and create alternative approaches.
You notice that the sentence that it's
created encompasses a lot of the rules,
it encompasses enough of
the information about what it's trying to
accomplish that we don't have to
worry as much that it's going to forget the rules.
Then I go and I give it another task.
I say write a prompt to evaluate
different prompts for summarizing the questions,
and it goes through and it generates
all the prompts which we saw before,
and then at the end, it generates the tail again.
Please let me know if you have
another task for which you'd
like me to write a prompt
and create alternative approaches.
We've got it to re-introduce the task again.
Now if we went on and we put in a new prompt,
it still will remember what the context is.
If we went in and we had back-and-forth without this,
at some point, it's going to
forget the rules of the game.
It's going to say, what am I supposed to do?
Or it's going to forget to generate alternatives,

or it's going to do something else.
But by continually re-introducing
the rules of the game as this tail on the output,
then every time we go to send it a new input,
just recently we had the rules of the game and gave it
the context that needs to accomplish the task.
The tail generation approach is
really useful when we're
going to have a longer conversation,
where we're going to try to program something.
We're going to give it a bunch of rules upfront and
then we're going to have a whole bunch of interactions.
We don't want it to forget the rules or the contexts.
Now it doesn't have to be rules,
it could be anything that we think that needs to be
re-introduced over and over into the conversation.
What we do is we just make sure
that at the end of the output,
we ask it to put in whatever that tail is.
It could be asking for the next input,
it could be asking for
some other additional information
that could be re-described in the rules,
it could be whatever you want it to be,
but it's like a little nugget,
a little tail that we're going to attach to
that output to remind the large language model,
here's what we've been talking about and give yourself
a little hint about it at
the very end before we go on to the next task.
Required
English

Help Us Translate

Large language models really excel at working with text and
understanding patterns of language.

And a lot of times, text is something that's really difficult to work with
because we tend to have to have a human go and
look at it to do a really good job of analyzing.
We used to need a human to do that now, we use large language models.
But going through and trying to understand text to manipulate text,
is one of the real strengths of the large language model.
The next pattern is called the semantic filter.
So this is basically a filter that can take in text and
remove information from it or only keep information that has
certain meaning or has a certain qualities that we specify.
And so this can be a really useful pattern for a variety of different cases.
Now, I'm going to show it in the case of taking text and filtering out or
removing certain things that we would like to give rid of from this text.
This might be secret information of some kind it might be removing things that aren't public,
it could be all kinds of different things or removing redundant information.
Whatever you need it for, this is a useful pattern in taking and manipulating text.
So let's look at an example of it.
So I'm going to start off with a very simple example of the semantic filter.
And I'm going to say, filter this information to remove all dates and
rewrite the text as little as possible to fix issues caused by the date removals.
And then I've pasted in some information about Vanderbilt University from Wikipedia
and in this information, there's a whole bunch of dates.
And so I'm creating a complex filter that's going to basically take in
information, it's going to remove all the dates and
it's also going to transform the information a little bit to rewrite it.
Just in case I'm removing a date and
it makes a sentence awkward where there's obviously something missing.
And so I'm going to take in information, I'm going to remove all the dates and
I'm going to slightly rewrite in order to do this.
So I've got the Vanderbilt now Wikipedia information and by using this pattern,
which is to clearly say that you're going to serve as a filter.
So we're going to say, filter this information.
So we're clearly indicating by saying, filter,
we want to preserve but remove, we're not trying to create new,

we're trying to take in information and filter out or remove.
So the pattern is basically the semantic filter pattern is,
we're going to tell it to filter something and we're going to give it rules,
semantic rules, rules about the meaning.
And what we're trying to remove the meaning or
the purpose of what we're trying to remove.
So then, what it does is it goes through and it rewrites all the texts and
you notice that it does a pretty good job, it removes all of the explicit dates.
Now, one thing I noticed is that, I might need to go and rethink my prompt and
how to express dates because it says, founded in the 19th century.
And that might be too specific for me depending on what my meaning of date is.
So I think it took a literal interpretation of it if it's a specific
date, so if you go back and look, it originally said founded in 1873.
So it removed the date 1873 and replaced it by rewriting it,
We found it in the 19th century.
So it's actually removing information, but I asked it to rewrite.
So it's transforming it a little bit, but this shows you the power, it's
understanding the text, it's understanding what a date is, it's removing it.
And in this case, we're also transforming it, to rewrite it in some way.
Now, let's show you a more interesting example and where the semantics and
understanding of the meaning of things really gets used.
So I'm going to say, filter the information below to remove
any information that helps someone figure out that the patient has diabetes.
First, explain what information you're going to remove and
how it might help someone figure out that the patient has diabetes,
then provide the filtered information.
So what I've got below is a fictitious medical record that
describes a patient Jane Doe, has a number of symptoms.
The patient presents to the clinic with complaints of increased thirst and
fatigue for the past two weeks.
History of present illness right here, past medical history down here,
medications down here, allergies, a variety of things.
And then I'm going to actually have ChatGPT go into
a semantic filter on this information.
Play video starting at :4:41 and follow transcript4:41

So now, let's take a look at what it's going to remove.
So it says, well, the chief complaint of increased thirst and
fatigue should be removed because increased thirst,
polydipsia is a common symptom of diabetes.
The history of present illness mentions increased thirst and
not checked your blood glucose levels.
Both of we should be removed because they are related to diabetes symptoms and management.
The past medical history, includes type 2 diabetes mellitus and
I apologize if I'm mispronouncing this which should be removed because it
directly indicates that the patient has diabetes.
The medication Metformin 1000 mg twice daily should be removed because
Metformin is a medication commonly used to manage diabetes.
So this is taking the semantics of,
I want things that could give away that this patient has diabetes.
I want them filtered out and removed, so
I'm giving some semantic rules about what type of information to get rid of.
It has to understand to some degree the meaning in the text, or
at least know based on its training data these
are things that are commonly discussed around diabetes.
So one way or the other, it's figuring that out.
And then, it goes through and rewrites the medical record and
notice it rewrites the chief complaint B, to the patient presents
the clinic with complaints of fatigue for the last two weeks.
The patient reports that she has been feeling more fatigued, she dies,
any recent weight loss, blurry vision or
infections past medical history is rewritten, the medication is rewritten.
And so now, we have a filtering of what was here before.
And so we noticed that like the history, it removed this,
the patient reports that she has been increasing or
she has been experiencing increased thirst for the past two weeks.
So we can use these large language models to develop powerful
semantic filters using the semantic filter pattern.
Now, is it perfect?
No, as we've seen, there can be issues with it and

as we've talked about before, they're not always perfectly repeatable.
But as a starting point to help,
maybe support a human process that then looks over it a second time, or
to try to help prevent accidental release, it's just one more check in the system.
This is a fairly low cost check to try to remove extra things,
should it be the only check?
No, if we're going to filter things out, particularly for privacy or
other reasons where we have information that we don't want to get out.
We should always have other checks in place because this is not going to be
a perfect filter.
It's going to have some randomness,
it's going to not always do exactly what we expect.
But as a part of another process, as a component to help,
it's a very, very powerful technique.
That we could potentially employ in parallel or
in conjunction with other checks, other types of removals.
Required
English

Help Us Translate
Thank you everyone for
taking this course on prompt engineering.
I hope that you'll feel inspired
by all of the different examples to go out and
really try to build
interesting application using large language models.
The thing I always want to emphasize in
these discussions about large language models
is the real power of
these language models comes out through human creativity.
If you have creative ideas,
creative ways of using the tools,
that's where you get really exciting outputs.
A lot of the discussion we've seen is
about bad outputs and I just look at
that as a failure in the use of the tool.

You want to go and really try to be
creative in how you use it.
Come up with really great ideas,
find the problems that are really hard
to solve using other approaches.
I hope you'll be expired,
inspired like I am
to really try to think about this tool.
Again, I think of it as a tool for
helping to quickly give my thoughts
form to do things faster and
in unique ways that I couldn't do before.
As an artist in my earlier part of my career,
I've always been excited about
working in different mediums.
I could work in paint or I could work in ceramics,
or I could work with metal and all of
these different mediums have
different unique characteristics to them.
I look at ChatGPT is
another exciting medium for creativity,
for coming up with new ideas
and trying to give them shape, to give them form.
Just like I wouldn't pay,
just like I would in sculpture.
I hope you'll be excited to explore this new medium.
As one last final thing,
I think it's always important to go back to
ChatGPT and get its thoughts on
what we should tell you
as you leave the course and so I
think it's got some good points here.
I asked ChatGPT to act as the teacher of a course on
Prompt engineering to thank students for being in
the course and tell them about

what they should do for next steps.
It's got some great points here.
One, practice, I can't emphasize this enough,
really a lot of the great things are
going to come through you practicing
and experimenting with ChatGPT
or another large language model.
Asking the question, can I get it to do this?
Wouldn't it be great if I could get it to do this?
A lot of times the answer is yes with
some creative prompt engineering. Please practice.
Stay up to date with
the latest research and
developments and prompt engineering.
It's all very quickly moving space,
if you go to archive where
we've listed some of the papers
on prompt engineering there you can
find all the latest updates.
I hope that you'll follow with what's going on and keep
seeing all of the exciting
discoveries that people are making.
It's such a new space,
everybody can be at the forefront riding
the wave right now and so I encourage you to do that.
Collaborate with others,
this is one of the things that I really
enjoy about working with large language models.
Is there so many people that are just as excited as I am?
When we get together,
we come up with crazy ideas.
Well, what if we did this?
What would it look like if
we had them talking to each other?
What if the large language models

are talking to each other?
What would we do differently or how
would they word things differently?
There's all kinds of fascinating ideas and
those ideas really come out through collaboration.
Also from just looking at
what different people are doing,
how they're approaching prompt engineering.
I encourage you to collaborate with others.
Then finally, to keep ethics in mind.
There's a lot of the discussion around ethics
with these models and I think it's really important,
particularly as the early adopters of this technology,
that we use it in an ethical and appropriate way.
That we show all of the good things
that can be done with it rather than showing bad things.
If we keep adding to the negativity around these models,
what we're going to do is we're really
going to dissuade people.
We're going to have people miss all of
the wonderful opportunities to
do interesting and creative things.
Please, obviously keep ethics in mind.
I think this is a great wrap-up from ChatGPT.
Again, thank you so
much for participating in this course.
I hope you'll be engaged in the discussion forums,
I hope you'll keep sharing
prompts and interesting ideas that
you have with the community and around this course.
Again, I can't wait to see what everyone in this course
does with prompt engineering and
all the amazing things that you create.
Required
English

Help Us Translate