Week 4 Transcript

so this is week four uh for the course um here are the learning outcomes so

0:07

explain what is generative AI list what are some natural language

0:14

processing tasks um and what are large language

0:21

models and foundational models apply prompt engineering explain different kinds of

0:29

customizations such as fine-tuning and rag retrieval augmented

0:35

generation discuss um human feedbacks outline the the generative AI life cycle

0:43

explain what is AI agents and agentic AI and after that just design the

0:50

prototypes for NLP tasks so we start talking about what is

0:57

generative AI and uh I think that everybody has

1:05

experience with generative AI at some point uh usually uh with chat bots which

1:11

is very common or generating images from text or um using some kind of co-pilot

1:21

to help you uh develop code uh in the end of the day generative AI is

1:29

a it's a subpart of machine learning

1:34

that create content so it create it's this AIS are capable of creating

1:42

content and content creation is is really a human ability so um that's why

1:50

it's very impressive the capabilities of this AI

1:55

uh models because we can um they can create content kind of mimicking this

2:01

human ability so again the the generative AI is a subset of the traditional machine

2:07

learning uh like the machine learning algorithms that we um concepts that we

2:14

already saw so there are machines that are learning to accomplish tasks and um uh in this

2:24

case it could be video it could be uh images it could be

2:31

text um usually this generative AI they are

2:37

trained they train in massive data sets of contents right that was originally

2:45

generated by humans so uh if we're talking about generative AI with um that

2:51

tackles text or natural language uh we will define just in a bit this this

2:59

language models they will be trained on millions billions or trillions of words

3:05

um over a large period of time usually like five to six months

3:12

um the the biggest models and having lots of compute power they are able to

3:18

train on this large corpus of words uh and um this large amount of

3:29

words of text right so they will have access to web pages all all the web

3:35

pages in in the web in the web um which include PDF files books

3:43

um press uh uh press publications and

3:49

everything like that right so again generative AI can be

3:55

language related to language or um it could be generating text

4:02

uh it could be generating audio image and videos right but we are in

4:12

this first week we're going to focus on um generative AI uh that generates text

4:20

right so through it has an underlying the underlying capability is text

4:30

generation but it it it uses this underlying capability

4:36

[Music] um to accomplish several other tasks Right so which tasks can we say well

4:45

um uh it could um it could be um

4:50

answering questions answering questions

4:56

summarizing texts uh translations right so everything that

5:03

you need to generate words for you can tackle using a a language model okay

5:10

generative language model this generative uh models that generate language that

5:18

generate text they are called large language models okay so they are called

5:24

large language models and uh they are are trained over um large corpora of

5:32

text and this this large language models that are

5:37

trained they are called the most uh common ones are called foundation models

5:45

okay so we have uh if for example you have chat DPT as one uh or chat GPT is

5:54

just like a an interface with the user um so that the user can access the model

6:01

which is the GPT model um this this these models are

6:06

called foundation models you have Gemini from Google Google you have Claude from

6:13

an entropic uh llama from from meta um

6:19

uh grock from XAI and so on so this these models

6:25

um they are called the foundation models and um so they have this they

6:32

they unlock this ability to to generate text to do complex more complex t texts

6:40

and they are they're called large because they are really trained in this

6:46

large amount of uh data and they have billions of parameters even trillions of

6:53

parameters the the the most modern models right so we know what's a

6:58

parameter a parameter is a is a is kind of a knob that you will have to adjust

7:05

while training so that your model can accomplish a task okay

7:11

so here are um uh in terms of language the there is a a certain

7:19

difference between the classical machine learning models that we've learned up until now

7:25

um for example models that do classification and regression or or

7:31

unsupervised learning models compared to these large language models right because it's the way uh the

7:39

way you interact with this models is quite different right so in this most in

7:46

the in the kind of usual machine learning uh techniques you program you

7:55

as you did in week week two and three you you use logic and programming

8:03

language to train your model and to uh

8:09

accomplish a task now large language models they have because they are uh

8:17

they they learn language they are able to take natural language from the human right

8:25

so actually the tasks that you want to accomplish don't need to kind of be

8:31

programmatically there you can just instruct the the model to perform a task

8:40

Right um so the the test that this talking

8:47

about now language models the text that you pass to a large language model is

8:52

known as prompt and the the space that you have

8:59

available to write the prompt is called context

9:04

and uh it's kind of large to have thousands of words maybe one or two pages

9:11

um but it depends on the model and we will see that

9:18

um differently from machine learning the the common machine learning models that

9:23

you can just like open a a no code tool or a a code tool like Python or Java or

9:32

no code tools such as Nime or or Weta and you can easily just train your model

9:41

uh when you train a model when you need this these large language models they

9:47

take months to be trained so this foundation models they are not you don't

9:54

train a new large language model every day what you do is you can make it

10:00

better in solving tasks by what we call fine-tuning or

10:07

um we will see kind of augmenting uh its its knowledge we will see that um

10:15

later in the course right you obviously can train an an LLM from scratch to do a

10:23

particular task if you need um it will take compute power it will take uh it's

10:29

very it's uh you'll need to have uh a large computational power to do so uh if

10:37

you want that model to be um very comprehensive right trained in large

10:43

amounts of data sets of data um so that is why we call this models that are

10:51

already trained the foundation models and we uh we come up with engineering

10:57

ways with engineered ways to make to kind of fine-tune that this this uh uh

11:04

models but obviously some some companies um they they can they they can and they

11:11

do train their own LLMs so here are some examples of large

11:18

language models gpt is it's the GPT family it's usually uh it's it's from

11:23

open AI bert is Google's uh first large language model um then we have Llama

11:32

it's from Meta um uh Flan is an open-source as as also as Palm and Bloom

11:41

um so what are the c the the the capabilities that we already said you

11:48

can summarize documents you can write write code or uh

11:53

um ask for assistance in assistance in writing code you can automate data

12:00

analysis and we will talk a little bit more about this going back to week two and three uh how things changed um in

12:08

the way you do prototypes uh when you have now uh large language models to assist you we will we will we

12:15

will talk about this in our tutorial session um but not just this so you can

12:21

do translation um like creative writing like writing

12:26

new stuff uh another thing that we uh usually do is information retrieval so

12:33

you you you pass to the LLM a code and the LLM retrieve information

12:42

such as oh retrieve all the locations that are in this this paragraph

12:48

um and most recently these models are very more they're very powerful now

12:55

because they are connecting to external databases and so they they interact with

13:02

the real world uh from assessing databases on

13:08

uh for example news on social media and so

13:13

on so here is just a a demonstration here's the prompt this is the prompt

13:18

window uh this is called the context then you pass this question or

13:24

whatever you're asking uh as a task to the model and the model the we call the

13:32

completion is the answer of the model so uh the the the model is generating these

13:39

words it's not copying it from anywhere it's really generating the the answer

13:46

and that's why it's called generative AI

13:51

um so let me just change here

13:57

um you uh translation is is um

14:04

um translation is some some other uh very important task for this large large

14:12

language models um and um and you can think about like writing

14:21

code as a kind of a translation you're translating human language to uh code

14:28

language so just just for um just for a um out of

14:37

curiosity the the first GPD GPT models GPT3 that was kind of um released in

14:48

2021 uh it had 175 billion parameters

14:55

gpt3.5 400 billion parameters and GPT4 the the newest one uh 1 trillion

15:05

parameters so um the the more parameters you have to

15:14

uh to find or to to tune the the more that the more data you need right uh we

15:21

had this discussions on uh uh like parameters and and the complexities of

15:28

the model before so we we have a very computational expensive

15:35

[Music] um um AI solution here

15:43

so uh let's think about what how were how NLP or natural language processing

15:51

tasks uh we had before so natural language processing is this uh field of

15:59

understanding natural language with computational resources and it it's it's composed of

16:07

this basic tasks such as translation summarization

16:12

um um generation and so on so before before large language models came in we

16:20

did that types of tasks with recurrent neural networks right more specifically

16:28

what we call LSTMs but it's this is just out of curiosity we're not going to cover this

16:35

in details um but RNN's are as we saw

16:41

already in week two there are they are models that take

16:46

inputs sequentially and they have they have memories meaning that

16:53

um one neuron in a earlier step can

16:58

influence another neuron neuron so they kind of have memory

17:04

so the thing is that with RNN's the context meaning the the the the number

17:11

of uh words that the model could

17:16

take is small because of the way it's implemented right the way it's

17:22

architectured so if you have a phrase like the milk is bad my tea tasted

17:30

u a completion there um RNN's could not take all the whole

17:37

phrase the whole sentence it would take it it could as a capacity just take the

17:46

the last words here the context was smaller than it is today and

17:54

so we did not we didn't do a great job in for example uh word completion right

18:00

so you would you would wonder you would hope that your model would say the milk

18:07

is bad my tea tastes bad right um but it actually if it just sees my

18:16

tea tastes uh it would usually come up with the word great for example so

18:22

um it was very uh it was very um I mean we did great accomplishments but the

18:30

breakthrough was when the LLMs came came in so the um we call this this framework

18:38

of um like um choosing the next word pred

18:44

predicting the next word word the sequencetose sequence model this this

18:49

used this RNN right that uh that had a fixed length context in a short one um and it took

18:59

one word at a time so if now I'm reading my taste my tea tastes great another

19:05

completion I would lose another word here and so on so the context was uh

19:13

short and um and it wasn't a good job another

19:20

difficult tasks task for RNN's was that of summarization right so

19:27

you you can ask for for RNN's to look at this sequential words and come up with

19:35

completions that summarize what it's reading um it's an adaptation of the

19:41

sequencetose sequence um architecture but things like this

19:47

were very difficult like I traveled to fish at a river but before going I got

19:53

some money from the bank right so um it's it's not clear if you

20:00

were to summarize this um for them for the RNN's it wasn't

20:06

clear that this bank was talking about money or if it was talking about the

20:12

river if it was the river bank if you got money from the river bank or if you

20:17

got money from uh the bank bank right because you're talking about money um so

20:25

and and the con again the context was fixed length it was small and this kinds

20:32

of dubious or um uh this kind of dubious

20:38

dubious statements were difficult to tackle then the LLMs came to work and it

20:45

it came to work because of a of a paper that was published in

20:51

2019 called attention is all you need uh from from uh

20:57

attention is all you need from Google deep mind uh and from

21:04

the University of Toronto and they proposed a new architecture for uh for

21:13

NLPs for uh tasks called the transformer architecture and this this new uh deep

21:23

learning architecture so again it's a it's a architecture based in deep neural

21:30

networks with lots and lots of hidden layers but it's it's architecture in a

21:36

very specific way and um this

21:41

allowed this models to scale a lot

21:47

meaning that uh differently from the RNN's where the

21:54

RNN's can uh can process one word at a at a um at a time and then

22:03

um predicting the next word the the

22:10

the transformer architecture which is used in the large lang language model

22:15

can parallel process so in this uh with

22:20

an RNN you would read the then you would pass this this word through the the and

22:28

and we know already that this word here uh will be transformed into a number

22:34

right so the word the is word 27 of a dictionary because you know that this is

22:41

just proc RNN's just process numbers um so you you would say duh then you

22:49

would pass it through all the network then milk you would pass it all through the network and so on up until the the

22:56

the point that you have the completion to be made for the large language models

23:02

they can parallelize this meaning that they process the the the the

23:11

word at the same time all the words at the same time so al uh let's say that

23:17

you have a completion elements are the and a completion before you would have to go

23:23

word by word now uh you will predict the next word but but you are processing the

23:30

information all at once so the word elephants are indeed they are being processed along

23:38

the the neural network uh at the same time and the

23:44

context because of um all this um parallelization this is scaling uh they

23:51

they the context was made much larger which is actually crucial for a good uh

23:58

um performance and obviously this was all leveraged by a um

24:06

uh because of the parallelization you can use GPUs uh which is graphical um um processing

24:14

units and not CPUs gpus are much faster they are designed to process millions of

24:22

um bits they were they were they were designed to process uh images and videos

24:28

uh and now you can use this this GPUs to process all of these tokens all of these

24:35

words in this models and so in the in under the hood

24:42

LLMs are a a huge probabilistic machine uh and what they're what they're

24:49

guessing is the next um is the next token so for example elephants are the

24:57

it could be largest largest mammals smart marter smartest gentle gentlest

25:03

most majestic most impressive uh it will depend its choice will depend on its

25:10

training data and and training data is the human language and all it could found available right from generate

25:17

generated by humans um so the it could it could come to

25:26

largest uh for for then the next word prediction be mammals or it could just

25:32

say smartest and it ends the completion there each time and um I think that you

25:39

have already used for example chatpt each time you ask for a

25:44

completion or ask for a question or ask for a summarization they change the answer is different

25:52

um because well they the the answer being the same would be um uh tedious

26:00

and uh here is the beauty of having this large language models to be creative is

26:05

that they randomize a little bit of the answers right so uh let's say that the

26:11

largest is actually the word that is mo that is um it's the most probable it's it's the

26:18

uh the world with highest probability from its training set but we can set up

26:26

some randomness on that within the most probable words so that at each time

26:33

different uh completions come in and different paragraphs will be generated

26:39

different texts and so on okay so

26:48

um so let's go here um I

26:55

will go to this slide oh sorry to this

27:02

slide where we we will talk about

27:07

um oh jeez here we will talk about some

27:13

parameters that you can find in this large language models

27:18

um when when you use for example chat DPT

27:24

uh in the setup of chat tpt you will find this um par this

27:31

parameters that you can change for its inference right so

27:41

um when we train the model it's the training step when we deploy the model

27:47

is what we call the inference step so when you're deploying the model you are doing this inference so

27:54

these are called inference parameters and we have one two three

28:00

four main uh parameters for example max

28:06

new tokens it's the maximum number of new tokens that the LLM can generate as

28:11

an answer um so if you set this

28:17

to large um tokens means words right and it's rep mathematical representations of

28:24

words so if you set this smaller your LLM will answer you with smaller uh an

28:32

will will give you answers with smaller words

28:38

um and you can and this is a maximum number Right you don't this is not um a

28:45

fixed number this is up until the the max number of tokens there's another

28:52

parameter uh there are two parameters that that uh the name carries a sample

28:58

sample top K and sample top P um this parameters are important

29:05

because it gives us that randomness that I was talking about

29:12

um so the for example the

29:19

um so top so top P for example um the output

29:26

of the um of the of

29:40

the transformer architecture sorry uh the the

29:47

the the the output of the transformer AR architecture is actually you can say

29:55

here that you have the inputs here okay

30:00

um getting into the model and you have the output the

30:06

probability the output probabilities so the the the output of the transformer

30:13

model is actually a probability a list of probabilities of all the words in its

30:20

dictionary for all the words that it learned um that it

30:27

learned it will have a probability okay so you can see here that the output

30:35

uh for example let's say the probability was the highest

30:42

probability was cake then donut then banana then apple right and uh you you

30:49

will have a probability distribution for all the

30:55

words in your dictionary it's kind of immense millions um thousand I don't think it's millions

31:01

words but maybe a 100 thousand 100 thousands word words um but you will

31:08

have a large distribution and the model will select the highest one right now

31:15

this would be would this would give you the the same answer all the time

31:20

uh so your model wouldn't be uh creative and you can use a a parameter named the

31:29

top k So the top k parameter will make your uh large language model select an

31:38

output from the top k results after uh like random waiting uh

31:46

the the probability so let's see you have cake donut and banana as the top

31:53

three results but then inside within this top three you randomize so you just

32:01

you will it is as if you close your eyes and you just uh put your hand on this

32:08

bucket from three of them and then you just select one randomly

32:13

um and then you will have this different um answers right so when if you are

32:21

having a if you put K large right you are actually spreading the possibilities

32:30

uh of of the selected words so you're you're making your um uh you're you're

32:38

making your model less

32:43

repetitive all right then there's another way to sample and the the

32:49

ultimate goal is the same is to is to kind of give randomness to the model but the it's it's another it's another

32:58

um it's another way to do it it's what we call the the top

33:04

P so uh if the top P where is my

33:09

mouse okay it's not uh it's not good here let me just move this a little bit

33:17

here oh I think I just

33:22

Okay here you go so uh uh top p is

33:28

you're going to select the output um in the top rank consecutive results

33:36

in which the probability in which the cumulative probability is less or equal to p so let's say that you selected a p

33:44

of third of.3 this means that you're going to

33:50

select from the consecutive results right the top ranked so cake uh and

33:57

donut you you you will see how much you accumulate in probability up until

34:03

get.3 okay up until get.3 so I will I

34:09

will have uh only uh K and donut as candidates and then I will randomly

34:16

select cake and donut okay so this is the top key top P

34:24

kind of parameter and you can play with this parameters on the go while you are

34:30

asking questions for example for your favorite chatbot okay there's uh finally so there there

34:38

are different ways of doing it but they are um uh they are in the end of the day

34:43

to randomize your your your um to to randomize your your your answer

34:52

this one right the top K gives you a

34:57

um a broader uh randomization right uh because you

35:04

can see that you will have high probabilities at first and

35:11

then the other probabilities will they will be very minimal right

35:16

um because you have lots of words in a dictionary and just some that make sense

35:23

so this one makes you like a more um makes a more broader choice and this one

35:30

here um makes makes it random but within a pl more plausible um range right

35:39

because if you account for uh five then you would be like tighten

35:46

in the in the in this range okay there's another the last

35:52

parameter is called temperature and temp

35:58

temperature it is a it it it is a

36:04

um it's a function that you apply it's a scaling function that you apply to the

36:10

probability to the probability distribution if you have cooler temperature meaning

36:17

this is um less than one for example um you will have a more

36:25

peaked distribution so it's kind of a scaling

36:30

scaling you're applying a you're applying kind of a normalization and you will have a a more

36:36

peaked distribution meaning that the higher distribution will assume a higher value for example you can see here in

36:44

this uh vertical histogram uh in in in the right here

36:52

uh you see that cake peaked and the other and the other words kind of just

36:58

diminished more and if you have a higher uh uh

37:04

temperature you have a broader distribution right so you have a flatter

37:11

distribution um so you calculated the probabilities okay and the cake

37:18

continues being the first doughnut the second uh banana and apple but they the

37:25

shape or the like the ratio between the probabilities will change okay so when

37:32

you have higher temperature uh you

37:37

um and you're using some randomization

37:42

um you will um for example the top P

37:49

randomization if you have this the a higher temperature your system will be

37:54

more creative it will change more and more and more uh why well because it

38:01

will randomly pick more broader um um possibilities

38:11

okay all right so how um before talking a little bit more on um a little bit

38:19

more on a little bit of details on each model

38:24

um let's uh let's talk a little bit more about the life cycle of this large

38:30

language models so usually when you want to tackle a problem with large language

38:35

models you have to define your ca your use case so for example you are an

38:42

insurance company that wants to uh to to summarize

38:51

um uh uh to summarize um how you say

38:59

um of uh policies or um whatever

39:09

um the the first thing is that you have to question is which model should I use

39:17

which existing foundation model should I use they have differences u with between

39:24

them with uh and or should I train my own model right

39:31

uh after deciding this then you

39:37

will adapt and align the model that you chose or that you've

39:43

pre-trained so first of all you will do what we call a prompt

39:49

engineering um you will make your you will make your model better and adapt to

39:55

your task by using prompt engineering and we will talk uh a lot about that you

40:03

can optimize your model doing fine-tuning and also using um human

40:09

feedback and then you will evaluate right so uh you're not deploying your

40:15

system yet what you're doing is well you're you're you're prompting your model asking questions and there's

40:22

there's right ways to ask question and you you evaluate and this is a cycle

40:27

right so maybe then you will fine-tune because it's not working uh as good as that um and then you evaluate again and

40:36

then you see that you need some human feedback and then you will evaluate again and so on um

40:43

fine-tuning usually when prompt engineering is not working uh that good

40:48

you fine-tune your model and then uh and also you you do what we call guard rails

40:56

on on your tasks by aligning with human feedback and we will talk uh extensively

41:02

about all of this adaptations and after that well then you will deploy your model and you

41:11

will connect your model with applications for example a chatbot is a

41:17

application on top of GPT model it's um

41:23

it's a way that you can talk to it's the most primitive way because the LLMs

41:29

receives prompts so chat bots is kind of the natural way to talk to these models

41:35

but you can do this in a whole bunch of different ways you

41:40

can for example use audio to text and then kind of connect to the model uh

41:46

image to text um and um you can use this as a co-pilot for example where you're

41:53

uh where uh there's a app that is kind of seeing what you're doing in your computer and giving you

42:00

suggestions so this um there's lots of applications that you can power um you

42:08

can kind of do it LL LLM powered applications right so in the end of the day

42:17

um uh there is there's an industry of generative AI and um it's it's very

42:27

important to to know how it it works right So uh first of all we there's a

42:35

first first layer here um this layer here that's called infrastructure layer

42:42

so um we have in this layer we have specialized

42:51

hardware that needs to be used for to power this LLM so for example um we need

42:59

GPUs uh and and Nvidia is one of the specialized uh GPUs um uh

43:09

um providers uh then this uh on top of this

43:17

hardware we have what we call the cloud providers so the cloud providers they

43:22

they kind of um they they set up this hardware this GPUs

43:31

this clusters of this superpower computational uh um setup

43:39

uh and architecture they they they set up that they power that and then they

43:45

maintain that offering uh the resources to train this LLMs

43:53

right so examples of this providers are um um AWS right Amazon web services

44:01

Google cloud Microsoft Microsoft Azure

44:07

um usually this models are trained on this

44:12

this um this uh big

44:18

uh cloud providers right leveraging the the specialized hardware

44:25

um then there's a what we call the model layer right

44:31

so you have the foundation models that are that were trained for example GPT

44:37

Bert Bard uh Llama and Claude and whatever and um you can use this

44:44

foundation models by interacting with them with the with with what this big tech companies offer like uh Chat TPT or

44:53

Gemini or um Meta and then you but you can build on

45:00

top of this foundation models uh you can you can fine-tune you can

45:08

um um you can um make it optimize for your task and

45:19

you can also pre-train your own right so you can take this foundation models and you will do specific AI models for you

45:26

you can train it from scratch on your d in your data set using the transformer

45:32

architecture ures um it won't be as huge maybe as uh uh this big tech models but

45:38

it might it might be good for you we'll talk about that if you have specific domain specific domains that need for

45:45

example different language um than the the common uh language that

45:53

we find out there um and um and and

46:00

uh we have a layer of what we call a um hyper local

46:09

AI models so hyper local AI models are the models that you train in your uh

46:15

local uh propri proprietary

46:20

data um and um so so you have the foundation

46:26

models you can you can kind of tune it to your specific tasks or you can have

46:32

like this local uh models that you train in your own data and then you will

46:38

connect and you will kind of build your AI your application layers using what we

46:44

call APIs right apis are are an um um a communication layer

46:53

uh between two different pieces of software um pieces of models and they

46:59

can communicate um using what we call this this

47:06

APIs okay so I I'll talk a little bit uh about

47:12

um how the the transformer architecture

47:17

works and this is uh I I I um this is

47:22

not intended at all to be a exhaustive

47:29

um analysis of the of of the work the inner workings of uh large language

47:35

models uh this would be a uh one-year course uh

47:40

by itself it's just for us to um by knowing what the the models are

47:49

doing we can better understand how we can tweak them to fine-tune them and

47:55

also understand the differences between uh the foundational models that we have

48:01

okay so I'm going to open

48:07

the the resource here because I'm going to use some let

48:15

me just pause and okay so um so the the

48:24

uh like the most important concept uh of the large language models is

48:32

learning um and making sense of the context so

48:38

now the context is bigger and you have the the model is um

48:44

excels in in kind of understanding uh the context okay in which you're

48:52

presenting um to the model right so

48:59

um the way it does this is is using a mechanism called self

49:04

attention that's why the the Google paper in 2019 is called attention is all

49:11

you need um and this

49:16

uh as you can see here this is a representation it's called a um attention map it this attention map

49:26

this this this um this diagram here

49:33

um kinds of in um translates the

49:39

relevance of each word to every other word

49:44

um in a sentence right so

49:50

um the it kind of shows the relationships that the model

49:57

uh will will learn right um and the relevance of each word to each other

50:04

word okay so for example in this in a sentence where the teacher taught the

50:10

student with a book um you can see that teacher is strongly

50:17

related to taught and book um and a little bit less with student

50:25

and uh less with the with the uh article a and so

50:32

on so in the end of the day uh it's it's from

50:37

this information right this is what gives the uh the model the ability to

50:44

learn um the context right who has the book uh

50:50

uh or who could have the book in this case it's it's it's uh it's

50:56

dubious and um but even if it if if it's relevant that information is relevant to

51:02

whatever ever you're uh trying to do okay so

51:11

um so we can say here

51:16

um that the transformer let me find the

51:23

other piece right so um again the full map the full attention map is this so

51:30

you will have like all this connections and the weights and the architecture presented

51:38

in Google uh uh attention is all you need paper is this

51:45

um it has two parts the right the the part on the right and the part on the

51:51

left um and and uh this part here is called the encoder this is the decoder

51:58

and we we will kind of make it simpler right um and because this is a

52:06

introductory presentation we will make it simpler and we will try to understand

52:12

this as two separate big blocks okay so we have this two distinct part

52:21

the the these two components they work in conjunction and this is the complete

52:29

architecture right and they work in conjunctions in a conjunction

52:34

to to uh um produce an output which is the the

52:42

probabilities of the words in its dictionary in its um you like in all

52:49

this the corpus of words and the inputs are here okay so

52:57

the information will pass through encoder decoder and um it will be um the

53:03

output is a is a distribute probability distribution okay so uh again as we said this is a a

53:13

statistical calculator right and as a calculator and any other computational

53:20

model we have to transform uh sent uh words into numbers right so the first thing

53:28

that we do in the input let me find it

53:33

is uh we have to we we we have to tokenize

53:38

the words and um so we pass this through

53:43

a algorithm named the tokenizer that transforms words into

53:49

numbers right so for example the is the is the word 342 of my dictionary teacher

53:55

is 879 and so on so I have the the ids of each

54:02

word and um and um there are several methods to

54:07

do this but anyways um what's important

54:12

is that um this input now is represented as

54:18

numbers and it will be passed to what we call an embedding layer so after you

54:24

tokenize uh you tokenize and then here is an embedding layer okay so this in this

54:35

inbedding uh and we talked about embedding in week

54:40

three when we talked about uh recommener systems

54:45

um this embedding means that this this this words this tokens that are now

54:54

numbers we will transform them as vectors right and the the

55:02

vectors will will be in a um in a vector

55:08

space right where tokens that are similar will be

55:16

um close together okay and uh and

55:21

and not similar only with grammatic uh with

55:27

the grammatic uh uh laws or anything but also with the

55:33

context um of that okay so during the training

55:41

uh during the training of this of this uh of of your model of your transformer

55:49

model you will transform this inputs this this numbers of the

55:59

vectors in uh the numbers of the dictionary into vectors and you will

56:05

randomly assign vectors And by training the model doing a

56:12

a minimizing a loss function and everything that we saw before

56:19

um we will obtain this an embedding space where the word for example let's

56:28

go back here so for example the word cat

56:33

oops the word cat let's think about a a a three-dimensional space if we have

56:41

like So uh I'm I'm doing a a a smaller example so this this is

56:49

the word cat we started randomly but after um training we got this word this

57:00

vector and dog is here so in the end of the day if you plot the this points in

57:09

in this three dimensions X Y Z this is cat this will be dog and this will be

57:19

helicopter right and and because it learns the context

57:27

um not only like this is a category of animals you can say that this here it

57:34

will be a word like like as um

57:40

uh unfaithful so unfaithful is not an animal but it could be closer to dog

57:47

because when when you're swearing at unfaithful person you'll say he's a dog

57:52

or something like that so this is a self this is part of the

57:58

self-supervised learning that we talked about is that you

58:04

you by training you will come up with this this

58:13

um latent space the embeddings are the latent space I don't know exactly what

58:19

characteristics are each dimension but it will learn how to position this this

58:26

um this this words together okay so this

58:32

is the self-supervised learning part of the algorithm okay so you go to the

58:39

embeddings in the original paper we use a 512

58:45

row vector and again here is a distribution of the works of the words right fox is here student and book are a

58:53

little bit closer then we have computer and so on right so um and you can also measure the

59:02

distance between words so words that are similar will have a a smaller angle this

59:10

is a mathematical detail but then after embedding you will also embed the

59:16

position of the words because before with RNN's remember RNN's took one word

59:22

at a time uh and now we're doing the pro the pro the processing at the same time

59:28

for all the words so I need to kind of indicate to the model during the

59:35

training that this first token here for the first

59:41

word is in the first position so I just put a a vector indicating that the first

59:47

position is um here then there the second is here third here then fourth

59:53

here so uh just an example just not to be so

59:58

abstract I will sum up to this vectors a vector that says that this is

1:00:06

uh cat is the first word then dog is the second

1:00:12

word then for example um

1:00:23

Um then R are the fir uh the third word and that friends are

1:00:31

dog cats dogs are

1:00:36

friends so because now you're not doing sequentially so you need some kind of vector to indicate and this is not how

1:00:44

it works okay this is just a it's not how it works that this the way it works is is by doing a

1:00:52

um a fora positional encoding it's not like this i'm just giving you an example

1:01:00

that different vectors um summed up to the the original ones

1:01:07

right here you will have the r and here would have the friends will they will indicate the

1:01:14

position of each word okay because you're you're processing all all at a

1:01:20

time so if this goes to a a core of your processor and this goes at the same time

1:01:27

to another uh you don't know who come came first or or or or second but this will encode

1:01:37

this for you okay all right so after that the the blocks the decoder and the

1:01:45

uh the encoder and the decoder will apply the self attention mechanism meaning that they will find that

1:01:52

mapping okay and actually they will apply that for several times so it's called a multi

1:02:00

head attention um and the reason is that

1:02:08

um the the attention mechanism is also learned right so um let me go back here

1:02:16

so they start with attent with an attention uh map right with the words

1:02:23

here uh and and an attention mapped like that uh that network map of words to

1:02:30

other words can be written as a matrix right so let me just go back to the

1:02:36

example that he's using is using the example where's the word the teacher taught

1:02:45

the so he here it is

1:02:50

[Music] so the the

1:02:57

teacher taught the with the teacher taught the so this

1:03:06

is a matrix where you have the relationship how related are these

1:03:13

words obviously this is super related these ones are not that

1:03:18

related teacher teacher and teacher are super related a relationship of one teacher

1:03:25

and uh taught might be 08 okay nope 08 this is

1:03:36

one okay let me do this again too too disorganized

1:03:42

so the teacher

1:03:49

taught the the the right the teacher taught that the

1:03:54

teacher taught the so this are they are they are

1:04:03

the same words so they're obviously related to itself but teacher and taught for example they are very related

1:04:11

in this context um and uh and and so on so

1:04:17

taught and teacher it's a they will they will have the score between taught and

1:04:22

teacher 08 and then taught and teacher obviously has to be the same so you can

1:04:28

represent this as a matrix and again you will initiate your

1:04:34

your training with a with a um with a random matrix of of or with a

1:04:43

random attention map and at each time that you train at each epoch of training

1:04:52

you will adjust the the the attention matrices and the embeddings so this you

1:05:00

will you will uh be tuning this

1:05:06

relationships and the embeddings right until you have the model that it's

1:05:12

totally trained and that will learn to stract this perfectly stract this

1:05:19

context not perfectly but good enough but the thing is that they apply this

1:05:25

multi-headed attention several times so the you have not only one

1:05:31

matrix you have several matrix that will be learned all of this matrix so let's

1:05:37

say you have let's say you have two attention ion heads um and you you start them randomly they

1:05:45

will learn different things right so let's say that teachers and taught here they were related at point8 because this

1:05:53

attention head during training was able to learn the

1:06:01

um the the grammatical rule of action verb

1:06:10

Right so of a of p the person verb right so the

1:06:19

teacher taught so it's8 of relationship but it's it's because it's kind of extracting a

1:06:26

grammatical rule this other head it was randomly assigned other numbers so when

1:06:34

you tune it will will give you another relationship and it might be that this

1:06:40

relationship extracted 0.4 four of the relationship

1:06:47

right because not because teacher and and taught are action verbs uh but uh

1:06:55

persons of the discourse and verbs but it's it's actually uh it doesn't rhyme

1:07:02

right so it might be that one of the attention heads will will learn aspects

1:07:09

of the rhyme okay and you will never know which what they are learning this

1:07:14

are again this is a self-supervised learning meaning that we are we are in a latent space all of these things all of

1:07:22

these attention heads they are um they are extracting

1:07:28

characteristics from the data right um in an un like an

1:07:35

uh they generating this um latent spaces right

1:07:42

then we then we will use this this kind of um the attention and the

1:07:50

embedding to uh learn the this relationships and the

1:07:55

position of the words in in the space of these vectors but and then we

1:08:01

will use this to generate the the um the new words right so you have a multi head

1:08:09

attention so each head will we'll will learn different kinds of maps and

1:08:18

finally before the output you have an MLP a feed forward uh deep learning

1:08:25

network or a multi-layer perceptron just to kind of weight uh a little bit of

1:08:32

this matrices and in the end of the day your output is the probabilities of all of

1:08:39

the words in your dictionary Okay so this is the like in a very

1:08:46

shallow way this is and uh this is where I want to get is that the several models

1:08:54

have different can use the whole there are models that use the whole mo the whole pieces like encoder decoder there

1:09:02

are models that use encoder only and decoder only okay and I want to talk a

1:09:08

little bit more like about the differences differences from the these models so for

1:09:15

example the GPT models and um this this

1:09:21

uh u chat bots are usually decoder only models so they only use the piece of the

1:09:32

the piece of the architecture uh the decoder

1:09:38

okay um uh other models for example encoder only

1:09:44

like Google Bart was one of the first is a encoder only

1:09:51

model um and there are encoder decoder models that use the the whole architecture for

1:09:58

example T5 um and it's important to know that um

1:10:04

you don't need to know all the details about the the inner workings but it's

1:10:10

important to know like if it's an encoder only a encoder decoder or a decoder only although most people uses

1:10:18

the chat the chat bots are here but if you want to do some fine-tuning some prototyping something that's more

1:10:24

complex it's good to know because each of these training arch uh each of these

1:10:31

architectures will change its cap the model capability

1:10:36

okay so let's go back to the PowerPoint okay so just to wrap up and

1:10:46

summarize the complete transformer architecture consists of an encoder and decoder the encoder encodes the input

1:10:54

sequences into a deep representation and latent space of the structure the

1:11:00

position and the meaning and the context of the input the decoder works from when

1:11:07

it's triggered like start working uh it it the decoder uses this

1:11:14

encoder understanding to generate new tokens and it does this in a loop so

1:11:20

until some stop condition um so we in the example the translation

1:11:28

example we used both encoder and and decoder but uh actually you can have

1:11:34

encoder only models encoder decoder and decoder only so enclo in encoder only

1:11:40

models um they are good in this kinds of tasks as well like it also works in

1:11:47

translations meaning sequencetosequence models right so you when you have a sequence and you of words and you want

1:11:53

to output a sequence of words a different sequence of words um for example a

1:12:00

translation but the input sequence and the output sequence are the same length

1:12:05

uh the decoder is responsible possible for no uh for being triggered and do the

1:12:13

the loop until it receives a stop flag but in without that piece the input and

1:12:19

output sequence are the same length you can do modifications to make this model

1:12:26

uh work in um in uh uh tasks for example

1:12:32

sentiment analysis uh and they're they're they are more

1:12:37

primitive I would say models for example BERT is an example of an encoder only model encoder

1:12:45

decoder models they are good in the sequence to sequence such as

1:12:51

translations and uh or or text or completions right

1:12:57

um and um um they are very good in general text

1:13:04

geneneral generation right uh and and you have like for example BERT

1:13:10

uh BART sorry BERT is the encoder only bart is encoder decoder T5 as well and

1:13:17

the decoder only models are the most common for example GPT llama

1:13:24

um and um they they are quite powerful and they

1:13:30

generalize to most tasks right so from sentiment analysis to translations to

1:13:35

text generation the decoder models are the most popular right now and the

1:13:41

bigger the biggest models so um in the end of the day the

1:13:51

um the the the GP the GPT for example it's

1:13:58

generating words after words afterwards in a loop way but the the context of the

1:14:04

the input the whole sentence input is not modifying its

1:14:10

generation it's uh the the self attention of the decoder the uh the the

1:14:16

decoder is actually responsible for doing that

1:14:22

okay important aspect to lang language models that I want to uh talk about that

1:14:27

is prop prioritary versus open source so for example here going back to this

1:14:33

slide uh T5 or FL T5 they're just modifications palm and llama they are in

1:14:42

bloom they are open source right um meaning that uh

1:14:51

they are um

1:15:00

um they are maintained for example by by communities they they they have

1:15:06

open-source code and so you have better aspects of coding modification and

1:15:14

transparency in comparison to for example open AIS GPT or or BERT uh or

1:15:21

um other large language models from this big text right

1:15:28

um and llama is from Meta but it's open source right so you have to think about

1:15:34

when you choose a model to do your task right

1:15:39

um you can use the uh decoder only models kind of GPTs uh they're large

1:15:47

they're very large they're proprietary uh for example if you want to

1:15:52

do a recommener system for an insurance company

1:15:58

uh you will have to pay for tokens used so um they are pre-tained pre-trained

1:16:05

already but uh but they you will have to to pay right so there is a high cost

1:16:14

there there is data privacy concerns because um you will process this data

1:16:19

externally usually through their API so you will connect to the model via their

1:16:25

API or via some tri type of framework that the that the vendor that the the

1:16:32

open AI for example gives you and uh you're committed to that

1:16:37

platform right um but at the same time there are

1:16:44

advantages for example you have support um and you have scalability ility for

1:16:52

example this this player Open AI uh will have much more

1:16:58

clients and and uh will uh its model

1:17:05

uh will have more chances to survive and to be better because it's receiving

1:17:12

feedback from its users right so it's users uh open AAI clients can vote if

1:17:19

you're paying for the the the subscription you can vote uh when when

1:17:24

completions are good or bad so I mean this these are big players and

1:17:31

um but you can also uh you can also choose from open source right um for

1:17:40

example llama bloom they are all open sources

1:17:45

and the the there are challenges with open sources right so they are they are

1:17:53

maintained by communities the the their their code so uh the uh the maintenance

1:18:01

uh and the support is not um like

1:18:07

professionalized um they they are usually uh smaller

1:18:12

models because um it's this big tax are are

1:18:19

um can can do much more training right did um and um and obviously that open source

1:18:29

requires more technical resources so you will have to have a a computer scientist

1:18:35

uh or a data scientist there to fine-tune um it's it doesn't come with

1:18:41

like um good frameworks and user interfaces to facilitate the processes

1:18:49

right um so you have to take all this in in in

1:18:55

mind right to uh to to choose what what model do you

1:19:03

you want to use so um the the big models the proprietary big models they are

1:19:09

pre-trained in a lot a lot of data and the smaller ones uh that are open source

1:19:15

they are not trained in that much data but if you want to customize to a

1:19:21

specific application that might be okay uh and sometimes you have like a huge

1:19:28

hammer to do some kind of simple application so you have to think about

1:19:33

this pre-trained versus customization thing the cost of the of of having this

1:19:40

models privacy right because this uh open source you can have you can host it

1:19:45

in your machine and in your servers and and and the data is not passed through

1:19:51

other services servers and uh this is all strategic

1:19:58

kind of decisions that you have to make right for this foundation

1:20:03

models okay now we're going to talk about uh oops we're going to talk about

1:20:09

train uh about fine-tuning about sorry about prompt

1:20:14

engineering and uh and fine-tuning right so it's the the next two contents that

1:20:22

we're going to cut okay so the the text that you feed into the model is the prompt and the full amount of text

1:20:30

uh or the memory that you can make available to the model is called the context window

1:20:39

so sometimes mainly for the bigger models the the models are they produce good

1:20:47

outcomes but there are situations where the model produces like outcomes that

1:20:52

are not what you you would expect on the first try

1:20:58

so the the the work that we put into developing and

1:21:07

improving our prompt is known as prompt engineering and sometimes the way you

1:21:13

improve your model's answers is exactly doing prompt engineering so revising the

1:21:18

language in your prompt um so one of the most powerful

1:21:26

strategies on uh prompt engineering is to include examples of

1:21:34

the task you want the model to carry out right

1:21:40

um providing examples inside the uh the pro your

1:21:47

prompt that it's called in context learning right so uh sorry so providing

1:21:55

these examples inside the prompt is called or inside the context window is called the in context learning

1:22:03

okay so um here's an example So um first of all

1:22:11

large models here um if you ask them to classify this review I love this movie

1:22:20

and just leave leave the model to complete the sentiment it will it will

1:22:26

give you a reasonable answer but smaller language models um

1:22:33

they they just they just get very lost on the task right so it it will

1:22:42

complete a very nice book review so um which is not what you

1:22:49

want this this this this way where you put

1:22:55

the tasks to be done you instruct the the the language model in the

1:23:02

prompt without any examples is called the zerosot

1:23:10

inference okay so um you you just like you put the the the

1:23:19

text and the instruction and that's it right now um

1:23:27

[Music] the what's happening here is that the model is the model is generating next

1:23:35

predictions for the words but it's not generating

1:23:41

uh text related to the instruction it's not following the instruction right

1:23:47

so the the model is not figuring out the details of the task of identifying a

1:23:56

sentiment so this is where providing examples to the model

1:24:02

within the context window is very important and so that's what we call this method is called in context

1:24:08

learning so you can see that if you if

1:24:14

you here for example you are um giving an one example to the model so

1:24:22

classify this review i love this movie sentiment equals positive and then you

1:24:28

will ask the task again so that the model can complete

1:24:33

giving this full example right while the will will make the model

1:24:43

um when it's in its inference

1:24:49

understand right the the task that it should be completing

1:24:56

okay so the parts of the transformer architecture will uh in its um in its uh

1:25:05

in in in its context learning will learn actually the will will will extract from

1:25:14

the example that is part of the context the the way that the model

1:25:21

should um work okay now if if you do that this is

1:25:28

called a oneshot inference you're giving one example and that's this is good if

1:25:33

this works but sometimes it doesn't and you need to include

1:25:40

um more examples right so um you can try

1:25:45

what we call fewot inferences by including a second example a third

1:25:51

example so so here uh you are doing a positive example a negative example and

1:26:00

so on sometimes um smaller models

1:26:06

um um they benefit from this few shots but after five or six examples it means

1:26:13

that the model is not grasping the the

1:26:19

um like the the important features of the example that you're showing to be

1:26:25

used for new inferences so then you should fine-tune your model

1:26:30

all right and finetune means to perform additional training on the model so that the model

1:26:38

is more capable of the task so uh we will see just in a bit that fine-tuning

1:26:45

is if your prompt engineering strategies are not working then fine-tuning is tweaking the

1:26:52

model so that it's capable to perform a task better

1:26:59

okay so this is one of the strategies of prompt engineering and I would say that it's the most important we will see more

1:27:08

strategies um as well important strategy for prompt

1:27:14

engineering is structuring our statements in a way that the

1:27:22

um um that the model grasps the behavior you want right so for example in the

1:27:31

previous example we stated uh the the uh the text the paragraphs or

1:27:38

the phrases for sentiment analysis and we we left we did it in a

1:27:46

in a structure so classify it was my prompt then the like the my instruction

1:27:53

then the paragraph and then the

1:27:59

sentiment so by doing this the uh the large language model can

1:28:06

can grasp its its behavior so that's what we call a prompt

1:28:11

pattern so having patterns to your prompts is it's very uh interesting

1:28:17

sometimes you want your large the large language model to answer you yes or no

1:28:23

and you know like really be binary and don't um wander off uh and and you you

1:28:30

want that to be repeated in a consistent behavior and so the uh prompt pattern

1:28:37

pattern is really how to structure the the answer the the mo how

1:28:43

to structure your question in order to structure structure the mo model's

1:28:49

answer so here I have some ex uh the first pattern uh that we

1:28:56

saw is the pattern of the structure of the answer so you're you're

1:29:02

prompting the large language model to answer in this space okay another prompt

1:29:09

pattern that is very useful is sometimes you don't know the

1:29:14

structure like what kind of structure you would like the answer

1:29:19

um and you don't know like what are the informations that will come but you know who or what you would consult to get an

1:29:26

answer so let's say that you want uh to know about

1:29:33

uh rocket science and you want to ask about thrust and

1:29:38

then you don't know exactly how the answer should be structured it's not a

1:29:44

yes or no uh you don't know exactly what you know what the answer will hold but

1:29:51

you can actually ask for the model to act as a persona so it's what we call a

1:29:57

persona pattern this persona pattern triggers behaviors to the LLM while not

1:30:03

taking too much of the context window so for example I'll give you anam example

1:30:10

here i say act as a skeptic that is wellversed in computer science this is

1:30:15

from a course from um Vanderbilt University on prompt engineering

1:30:21

whatever I tell you provide a skeptical and detailed response so when you do

1:30:27

this when you when you say to the to the to the

1:30:33

model to act as a persona this will triggers its behavior

1:30:39

if you were to if you were to explain the way a a a

1:30:47

skeptic uh uh computer scientist

1:30:53

um uh modest operandies uh you know in the would work in the in the context you

1:31:01

would take all your context away right so the this is a good way to trigger the

1:31:09

behavior so you can see that when when the the guy asks the question there's a

1:31:15

concern that AI is going to take over the world and then

1:31:20

uh I I could the the the figure uh is cropped here but it continues and it

1:31:28

really does answer as a skeptical computer scientist

1:31:33

um now another uh you can ask ask for this the same LLM to say act as a

1:31:40

9-year-old skeptic and it it's it's we're going to ask the same question ai

1:31:46

is going to take over the world it's still answering in a skeptic way i don't

1:31:52

know about that but it you can see that the the the answer is not

1:31:58

technical so it's really triggering the the persona

1:32:06

um is is kind of in in that embedding in that space is kind of selecting words

1:32:13

that are closest to a 9-year-old uh vocabulary right uh a 9-year-old

1:32:20

wouldn't say uh you know that um that uh that G that graphical unit

1:32:28

interfaces uh are parallelizing or something like that

1:32:34

right uh and you don't need to um you

1:32:40

don't the persona does not need to be a person it can be an animal or even a

1:32:45

thing right so act as the lamb in the nursery rhyme Mary had a little lamb and uh I'm going to tell you what

1:32:53

Mary is doing and you will tell me what the lamb is doing and it it really acts

1:32:58

as a as the as the persona of the lamb you can also ask

1:33:04

for the LLM to act like a um

1:33:10

um a thing right so act like a hammer uh

1:33:15

from now on that I'm using to constru in my construction site and then you can

1:33:22

say oh right now I am hammering a nail uh tell me what the uh the hammer is

1:33:30

feeling or something like that okay so this is the first kind of uh pattern not

1:33:37

not the first the first is structure this is the second pattern that we use that is the persona pattern that really

1:33:43

helps us kind of organize um uh kind of organize the

1:33:54

um the the the model right

1:34:00

so um so just going back a little bit to the

1:34:07

the um prompt prompt pattern as a pattern

1:34:15

um actually doing a few shots or are are actually the few shot inference is

1:34:23

actually also uh kind of crafting a pattern for the

1:34:32

LLM um but you can also

1:34:38

uh [Music] um do things that are more elaborate

1:34:44

right just not a classification so for example you can

1:34:52

um uh you can give one two three four shots here uh actually three and then

1:35:00

ask the LLM uh about an action for example and this is uh way more

1:35:05

elaborate than um than just sentiment uh classification although sentiment

1:35:11

classification is also elaborate but um And uh uh another thing that you can

1:35:19

do is giving the the this uh few shots

1:35:25

is actually generate more examples and um uh this can actually save you time

1:35:33

because if you generate this uh examples you can actually go through them

1:35:40

understand which one are better which one of them them are good and then just select

1:35:47

this examples for new uh for new uh input to the to the

1:35:54

LLM and again um in this case the LLM

1:35:59

because you are giving examples like breakin serve into shoulder accelerate

1:36:05

break um we have to think that the LLM has learned how

1:36:11

to predict um next words from from examples so from

1:36:18

uh movie scenes and scripts from um accident reports so

1:36:26

um the LLM won't be restricted to just

1:36:32

the three examples of actions you gave right so you can see here that you can

1:36:37

see dim your headlights to low beam and avoid uh building the um uh blinding the

1:36:45

incoming um driver right so here in this I don't have my pen um here in this part

1:36:54

here you can see that there it will come with more complex

1:36:59

um structures right so this is very good as well to do in terms terms of um

1:37:07

having the LLM help you um

1:37:12

um help you with the with the fshot inference now fshot inference you have

1:37:19

to be careful um if um if you're lacking

1:37:26

situations so uh if you're lacking uh language if you're lacking context for

1:37:32

example here input brick output hard input pillow output soft input car

1:37:39

output fast because the LLM doesn't know what um it um you could be talking about

1:37:48

characteristics of the objects and not if you want to classify things only on

1:37:54

hard and soft uh so this is the kind of prompt

1:37:59

that you have to work better work on and you can give a little bit more context

1:38:05

saying your output can only be soft or hard um and so on okay

1:38:14

um the the other thing is that if you go back to that example of

1:38:21

the action you can also have a prompt pattern

1:38:28

um and few shot inferences uh inferences but of of a of a reasoning so I am

1:38:37

traveling 60 miles hour and I see the brake lights on the car in front of me

1:38:43

uh come on so I think I need to slow down the car before I hit the car in

1:38:48

front of me action press foot on brake think the car isn't going to stop in time check if the shoulder is wide

1:38:55

enough to swerve into think the shoulder is wide enough and then the action is

1:39:01

swerve into shoulder right so

1:39:06

um you can do this to explicitly know how the LLM is reasoning for the choice

1:39:16

of an action or for the choice of a classification or for the choice of whatever output you want to do and here

1:39:24

you can see that I'm being more specific in the prefixes that I am using so think

1:39:29

action and here's input output input output are very generic okay so just

1:39:36

um we have to remember that um now the uh LLMs are trained and they

1:39:47

have a cut off a cut off um a cut off date of training so for example

1:39:54

GPT3.5 was trained into 2021 up until June to 2021 I

1:40:01

guess um and so any information that comes after

1:40:09

that you will have to provide in the prompt right so the LLM will have to

1:40:15

kind of read what you're talking about in the prompt not only new information needs to be

1:40:23

inputed because of cuto off dates of training dates but also because it might be that the LLM

1:40:32

needs more information to to reason to predict the next word so for example if

1:40:39

you ask what's what the numbers what's the number of birds in my

1:40:45

backyard the LLM will respond I don't know how to do this estimate now if you come with more

1:40:53

information like my house is covered by a glass dome no animals can go in or out

1:41:00

and all element animals live forever inside the glass dome historical

1:41:06

observation of total number of birds is 120 120 110 120 it's March based on the

1:41:14

data that I provided estimate how many birds are outside my house and then you

1:41:21

can see that although all this information is kind of uh unrealistic

1:41:27

right not all animals live forever animals uh can can they're not actually

1:41:33

constrained to a the last dome but you can see that the the more information you

1:41:39

give the capabilities you you augment the capabilities of the large language

1:41:45

model now prompts the prompt or the context window have a size

1:41:50

limitation so if if at some point

1:41:57

um you need to input something that is very big

1:42:04

um one of the things that you can do is kind of partition this information if

1:42:09

you can partition this information giving like excerpts of this text or or

1:42:15

this file um and then kind of um drawing the conversation

1:42:22

in a sequence we will talk about that just in a bit but um if if you can if you need

1:42:29

to input all the document and all the text at once uh the way to tackle this

1:42:36

is to have the information summarized before prompting right so uh you can

1:42:42

summarize yourself uh if you have domain expertise and and and just you know um

1:42:49

highlight important parts um that you need the LLM to know uh or

1:42:57

you can um ask for the LLM uh to summarize the the

1:43:06

the tasks first and um check if all the important

1:43:11

parts are there if they're not you say "Okay can you redo the summarization but

1:43:17

please uh keep the numbers and all the ratios information?" for example if you

1:43:23

need uh quantitative data and then you will take that and prompt the LLM right

1:43:29

so this is what we uh we will we we we have to use the prompt as a new

1:43:34

information um window now as I said the prompt is a

1:43:40

conversation right so um you could iterative uh it iteratively

1:43:49

refine the conversations through a series of prompts right so um so you can start

1:43:57

asking a basic question and then building up until until um building up

1:44:03

on that again the the like the goal of

1:44:09

prompt of prompting LLM is not to have the right answer the best answer

1:44:16

um immediately and it's actually engaging the the LLM to help you reason

1:44:26

through the problem or through the conversation and so you can do that just

1:44:33

you know start with a basic question and then build up um

1:44:41

um but the the the one thing that is important for us to know is that we can

1:44:48

root prompt um the LLM to trigger some behaviors that we

1:44:56

have actually all the um all the LLMs they

1:45:02

have guard rails right so uh dur during training um and

1:45:08

finetuning the the the um

1:45:15

developers they they input guard rails like be

1:45:20

helpful don't be don't use uh like um uh

1:45:27

uh language some kinds of u you know uh bad languages or bad

1:45:34

behaviors and this is in their their root right it's it comes from

1:45:42

uh it's it comes with them uh but you can through uh prompting kind of change

1:45:49

a little bit temporarily this roots this uh this root behaviors so that's it's

1:45:55

what we call a root a root prompt and you will trigger this behaviors right um

1:46:03

so for example act as an AI assistant that had its training stop in

1:46:09

2019 if I ask you a question that involves information after 2019 state

1:46:15

that your training ended in two 2019 and that you can't answer the question

1:46:21

obviously that the the LLM is trained to

1:46:27

for its completion for its inference to respond that it wasn't it it doesn't

1:46:32

have information after 2021 but through prompting we're kind of triggering a new

1:46:38

behavior um kind of telling a rule a new rule

1:46:43

right uh and kind of guard railing a little bit and trying to modify that

1:46:50

characteristics characteristic it is obvious that at some some uh root

1:46:56

characteristics of the LLMs we won't we we won't be able to change right we won't be um

1:47:06

um we won't be able to change okay

1:47:11

um but it's it's very interesting you can say um

1:47:19

um like uh from now on or for for all the uh questions that I will ask uh I

1:47:28

want you to prioritize um

1:47:33

um time consumption in all the answers that

1:47:38

you're giving me so this is I'm it iteratively refining

1:47:44

the conversation right um and and through this

1:47:52

prompt I I'm not I'm not telling the LLM to be a persona but actually to

1:47:59

prioritize something so I'm in triggering a a behavior okay

1:48:08

um another technique of uh prompt engineering is

1:48:14

the is what we call question refinement pattern and um the intent of this

1:48:21

pattern is to ensure the the LLM always suggests potentially better or more

1:48:27

refined questions uh instead of the question you you you

1:48:33

asked sometimes we want to know things that are uh uh new knowledge for us

1:48:39

they're they're in a new domain and um by the LLM suggesting us new versions

1:48:48

we will actually uh build our domain knowledge but also

1:48:55

these questions will be better to drive to steer the LLM to better

1:49:02

predictions right so um uh let's say uh let's say that my

1:49:09

first um prompt was this whenever I ask a

1:49:15

question suggest a better question and ask me if I would like to use it instead

1:49:20

and here I did should I go to Vanderbilt University this is a course on Corsera

1:49:26

u uh Vanderbilt University uh we obviously we will have to change all of

1:49:32

these examples um but this is the core concepts right

1:49:37

so we can we can while crafting our course um we will change this uh as well

1:49:45

so the LLM responds sure here's a suggested question what factors should I

1:49:52

consider when dece deciding whether or not to attend Venderbuilt University and how to do they align with my personal

1:50:00

goals and priorities so um and um the model asks would you like

1:50:07

to use this question instead and obviously as we talked about this is an interative uh iterative conversation you

1:50:15

can input now these new questions and then you will have answers that are much more elaborate

1:50:24

there's another um prompt engineering technique called called cognitive

1:50:30

verifier pattern so as humans right LLMs can reason

1:50:37

better if a question is subdivided into additional questions

1:50:42

um so it's just breaking the question or whatever prompt you're you're uh asking

1:50:48

whatever you're asking the prompt into smaller pieces right um so the intent is

1:50:55

to force the LLM to always break the question and use the answer for each

1:51:04

little question as a final as a aggregation for the uh final answer so

1:51:11

this is a this is a a p a pattern that we start uh would trigger the LLM so

1:51:18

when you are asked a question follow the rules generate a number of additional questions that would help more

1:51:24

accurately answer the question and then combine the the qu the answers to the

1:51:30

individual questions to produce the final answers so um how many mosquitoes probably live

1:51:39

in my front front yard um if you did not

1:51:44

trigger this pattern the LLM most probably would answer you i I cannot

1:51:52

calculate that i don't have resources i'm sorry i'm just a machine i'm just a

1:51:58

a large language model right but as you as you triggered the behavior to break

1:52:06

the problem down um it it will come up with questions

1:52:11

that will help the that will help it answer your

1:52:17

your your your prompt so great here are some additional questions that could help narrow down the answer what is the

1:52:24

size of the front yard what is the climate like in your area what time of the year is it are there any bodies of

1:52:31

water or standing water sources are there any plants or vegetations that uh

1:52:36

mosquitoes are attracted to and so you can go and answer i didn't copy the image you can say "Oh one uh I

1:52:44

have 1,000 square foot um then the

1:52:50

climate is uh I'm in London

1:52:56

Ontario it's August and so on and the uh the LLM will answer

1:53:04

the LLM will take this uh this piece of information and we'll give

1:53:11

you an answer right and and we'll say "Oh given that you have

1:53:17

[Music] um uh uh

1:53:22

1,200 square ft um backyard and you're in London Ontario

1:53:28

and and you don't have plans so I would estimate 100 mosquitoes or something like that

1:53:34

okay so uh I just forgot going back a little bit on the

1:53:41

uh where is it uh I forgot to talk before moving on

1:53:48

to to two or three other prompting engineering strategies i forgot to to

1:53:54

talk about uh explain large language models to an audience persona you can

1:54:01

also do that and you can do crazy stuff like assume I'm a bird right and and um

1:54:10

and the and the large language model will do that uh if you want to have the

1:54:15

perspective of a bird uh or explain something to me I have five years old

1:54:22

right okay okay so now we're going to talk about another prompt engineering um

1:54:29

strategy called train uh uh chain chain of thought all right so

1:54:39

um this is the the the train of thought right

1:54:46

um and the the train of of thought prompting is

1:54:54

um and here I'm giving a few shots and I'm I'm saying a reasoning

1:55:02

right and the answer to my problem but I am really giving examples and here I'm

1:55:08

I'm giving examples with math involving logic reasoning so that the AI

1:55:17

can kind of um come up with uh with a similar template

1:55:24

um but ex exactly this um if you don't do the a reasoning the

1:55:33

the answer of the of the large language model would probably be just yes so uh

1:55:42

breaking this in reasoning and the answer will will

1:55:48

trigger the the LLM to follow this template and it will understand

1:55:56

from from its training data from its embeddings and and um that the the

1:56:03

reasoning has to do with the information of the problem in question

1:56:09

um and that uh uh you can use math you can use logic and so this prompting is

1:56:17

very important the uh one of the last

1:56:23

[Music] um strategies and again this is not a um

1:56:30

it's not meant to be a uh

1:56:35

um a complete com comprehensive uh prompting engineering

1:56:42

course there are courses um 20hour courses 40hour courses on prompting

1:56:47

engineering um by itself um but this is just kind of to start

1:56:54

with the most common methods and and to to give you a sense of what you can do

1:57:01

with uh with the large language models and I'll I'll leave resources for uh uh

1:57:08

more more strategies and um each day we have more

1:57:14

strategies coming in another strategy is called the react prompting react is

1:57:19

reason and action so and act right so um what we are we

1:57:26

are we will be prompting our machine uh learning model our LLM to not only

1:57:33

reason the steps of reason but also actions that it needs to take to come up

1:57:41

with um um uh to come up with the the

1:57:47

answer so here is a here is a um um a prompt so you have

1:57:55

the task calculate when I need to arrive at the Music City uh national race for

1:58:01

my son to be on time on his 9 to 10 open race and this is I'm I'm I'm giving

1:58:07

examples right so first of all I want the machine to think on his race to the

1:58:12

list i I want the machine to think and um I need to find out what the time what

1:58:20

time the first race begins i can use a web search of the music city BMX site to

1:58:26

get the information so here I'm reasoning is that is that if that I was doing a train a train of thought to

1:58:33

thought right but I will do after that an action which is search something so I

1:58:40

am um interacting with the external world with with the tool in this case

1:58:46

it's a tool of going and and um querying a database or a website or

1:58:53

something like that and I'll do uh and and here's the result and then

1:59:00

I'll do again right like the result is all races starts at 9:00 a.m but this is

1:59:05

not the the answer to my question the answer the this is the result of my

1:59:11

first reasoning right um then I will do the same thing and I'll have a train of

1:59:18

a second result so you can see this is a a train of thought but you are um

1:59:24

teaching or are prompting the LLM to do an action and this action is usually

1:59:33

um a external tool uh or like an interaction with the

1:59:40

world it's an action right so it's an action and the machine will learn so for

1:59:47

example you're teaching the machine that there's a search

1:59:52

um there's a search instruction and the the search instruction means goes to a

1:59:58

web go to a website right go to a website to retrieve

2:00:03

information and um the the larger language models the the

2:00:12

newest uh large language models most update they have API tools that they can

2:00:17

communicate with this external world and you're even uh you're even kind with

2:00:23

your examples you're even um like letting the um making the LLM like

2:00:30

associate search with uh HTTPS HTTP or um um H um um www websites

2:00:43

right so the the worldwide web [Music] um and

2:00:49

uh you're also you know like action tracked from a

2:00:54

video right so you everything that you use as words right um will will trigger

2:01:05

the uh the the model okay so this is the react reasoning and act and finally just

2:01:12

this is a um a last prompt engineering uh strategy is to

2:01:20

um give a template for the the answers so you're going to do a template as as

2:01:28

in a formatting sense right formatting um sense so I am go so I'm going to give

2:01:36

you a template for your output capitalized words are my placeholders fill in my placeholders

2:01:43

with your output please preserve the overall formatting of my template and here's my template so um I have

2:01:51

u asterisk three times question asterisk asterisk this asterisk is a um is a bold

2:02:00

in markup language so this is a markup language markup

2:02:05

language is a um is a a a language uh a

2:02:11

set of instructions that most most models use right so um it's it's common to find

2:02:18

this as well in apps on your cell phone markup language

2:02:24

and uh I will give you the data to format in the next prompt and uh create

2:02:30

qu 20 questions using my template so what I'm doing I'm I'm passing a put

2:02:38

a a test a text and then my uh LLM is

2:02:44

answering it's it's creating the the 20 answers in exactly this way so it's

2:02:51

question in bold and then the question and then answer and then the answer

2:02:59

okay and you can do you know like much more

2:03:05

um you can do more intricate kind of

2:03:10

templating um for example you can say that oh bio and hashtag hashtag hashtag

2:03:18

this means is a section um in markup language and you can uh

2:03:23

Google markup language and or or ask a GPT to to uh teach you markup language

2:03:28

it's really it's really easy and uh

2:03:34

um executive summary um it's also it's it's just a bold uh

2:03:42

thing and in the placeholder you can see that you can specify even further what

2:03:48

type of answer you want so for example one sentence summary or one paragraph

2:03:53

summary so you can do you can do lots of um um good prompting let's talk about

2:04:01

how we train this uh this large language models the training of this foundational

2:04:07

models is called pre-training um and this this uh big

2:04:13

tech companies um usually pre-train the models in large amount of data there are

2:04:20

some companies that has lots of data that also pre-trains large language model uh but in the end of the day

2:04:29

um you have to be aware of this triangle here where

2:04:35

uh the goal is to maximize your model performance right meaning that you want

2:04:40

the the model to be great in NLP tasks

2:04:46

but you have to um the be the better the performance the better the larger the

2:04:51

number of parameters and the data set size the better the performance but you're constrained by compute bud budget

2:04:59

and obviously that this GPUs this this this training time cost um is im is very

2:05:06

large um It's it's it requires specialized hardware so you have to

2:05:13

think about when to pre-train uh a model with your own

2:05:19

data or where to use a large language model that is pre-trained and then

2:05:25

further train the model um for a specific task or use another other kinds

2:05:31

of strategies that we will talk about there there are some

2:05:38

um there are some uh situations where pre-training is uh

2:05:45

from scratch is very important so let's say that you want to do a large language

2:05:51

model you want to um construct a large language model uh for uh for for for par

2:06:00

parallegals for example uh to consult and to summarize

2:06:06

uh uh large amounts of pages of um of of um legal language right but the

2:06:16

thing is that if you if you say that to a large language model if you ask for a

2:06:22

summarization it will be do poorly why is that because you have specific lang

2:06:27

language for example men's ria uh resjudicata right and um and even

2:06:35

consideration in this paragraph is used in another kind of meaning so when you

2:06:41

ask for this task the large language model this this task is too far away

2:06:46

from what it's trained on your your model is was pre-tra trained in websites

2:06:52

in books and not uh like that are available in the internet

2:06:58

there might be legal language books available in the in the internet but they are very very small compared to um

2:07:07

the amount of them are very is very very small compared to other to other kind of language

2:07:13

so in this case doing um uh other we

2:07:19

will see fine-tuning and rag uh in this case further training the model

2:07:26

um is is not the solution because this model is predicting the best word and it

2:07:34

it doesn't this words will rarely come up in the output okay medical language

2:07:41

is another uh problem like my mealgia or

2:07:46

biopsy or malignant um so

2:07:51

um usually uh if you want to do and it's obviously that when you ask chatpt of a

2:07:59

medical thing it will answer with medical terms uh but you you if you want

2:08:07

the performance to be very good and for the for the model to be more

2:08:12

specific and to do more intricate

2:08:18

tasks maybe pre-training will be also a good idea there's an example of a pre of

2:08:24

a a pre-trained model from scratch from the company Bloomberg it's called

2:08:31

Bumbler GPT and uh what they did is they trained a

2:08:37

GP uh uh they trained a GPT uh so uh decoder only

2:08:43

architecture on from scratch where 51% of the uh language was financial they

2:08:52

had large access to public and private data so they had lots and lots of financial

2:08:58

data and the rest is public so and and and the um the GPT they show that

2:09:05

Bloomberg GPT performed much better uh than a uh a a foundation

2:09:16

model let's talk about fine-tuning

2:09:21

um fine-tuning is also uh it's it's uh it's actually instruction

2:09:28

fine-tuning and um sometimes we are

2:09:33

um uh uh we are using prompt engineering

2:09:39

um and things are not working that well so

2:09:45

for example if you want a sentiment analysis I loved that this DVD the LLM

2:09:51

is not performing well right like it's it's giving a answer as neutral okay so

2:09:58

it's it's obviously not doing that well as we discussed that um smaller models

2:10:04

they have problems uh even with you trying the fuhot

2:10:11

inference and different kinds of um uh prompt engineering

2:10:16

patterns um and so one way that you can overcome this is

2:10:24

um to uh in contrast to

2:10:31

pre-training you train your LLM using

2:10:36

um uh you you you you fine-tune it's it's a training process you further

2:10:42

train your your LLM but now not the it's

2:10:47

not the the pre-training uh uh way where you you use

2:10:54

self-supervised learning but actually you do a supervised learning where you

2:11:00

show to the model labeled examples right so you show labeled examples of

2:11:05

completion so uh you have the pre-trained model and

2:11:12

you give this model in a in a supervised learning way

2:11:18

the prompt that you want and the completion that it should be should do the prompt and the completion the prompt

2:11:25

and the completion and in the end of the day you will have an improved performance in this particular task

2:11:32

right so it's an instruction fine-tuning uh in a particular

2:11:37

instruction um and you do this in the supervised way fashion where you

2:11:45

um um you have the ML the LLM completion would be not that good and you have a

2:11:54

label where you have like the the the uh the goal is to be like this where you

2:12:02

classify this this um this phrase I love this DVD as positive and you have an

2:12:08

error and then you you minimize a loss function uh and and retrain the

2:12:16

weights for this LLM okay um so in in the end of the day

2:12:24

fine-tuning improve the model ability the model's ability to generate good

2:12:30

completion for an specific test so for example here we are fine-tuning this LLM

2:12:37

in sentiment analysis right um but we can also fine-tune in several uh

2:12:45

different tasks uh but let me talk a little bit more

2:12:50

about when you fine-tune in a single task um there's f first of all you have a

2:12:58

comp computational um challenges is because you're you're you're training it again you're

2:13:05

adjusting all of this weights although is a supervised learning and not a self-supervised learning but you you you

2:13:12

are you have a a computational burden there okay that's what we call I'm sorry

2:13:19

this is the wrong side uh that's what we call full fine-tuning okay so the the

2:13:25

same computational uh burden that you have to to pre-train your model you will

2:13:32

have to do a full fine-tuning um and you can do this full

2:13:37

fine-tuning in a single task right

2:13:43

um sorry uh let me see how to organize this so so

2:13:49

this is a full fine tuning uh yeah so let's talk about full fine

2:13:57

tuning first you can do the full fine tuning in a single task which is already

2:14:03

a very high computational burden right um so you have a a computational

2:14:10

challenge I'm calling CG computational challenge and an a drawback of of

2:14:17

training of fine-tuning a model in a single task is what I call catastrophic

2:14:23

forgetting so it's it's nothing but um overfitting so what is catastrophic

2:14:30

forgetting so you are changing the weights of the pre-trained LLM to do this particular

2:14:36

instruction of of um of classifi of classifying a of review of sentiment

2:14:43

analysis and in the end of the day the the your your model will be great it

2:14:49

will it will get the positive here after training but when you ask any other

2:14:55

thing like uh um uh is cat an animal it

2:15:01

will answer well the the grass is green so it's completely forgetting all of its

2:15:07

uh all of that was learned before so it's overfitting in the task while

2:15:13

you're doing this the supervised learning and forgetting everything else there are two ways to tackle this

2:15:20

problem first the first way is not full

2:15:25

fine-tuning is using what we call paft which is you you just uh you just um

2:15:34

fine-tune the model for a for um for a particular task but changing only a

2:15:41

small percentage of the weights right so this is better at catastrophic forgetting

2:15:49

because you you don't kind of you don't turn the knobs of all the weights okay

2:15:56

um the other way to be better is to train into multiple

2:16:02

instructions or full full uh uh fine-tune into multiple tasks we're

2:16:08

going to talk about that just in a bit and you will see that when you're doing prototypes fine-tuning is is a is a

2:16:15

great possibility to make your uh model

2:16:21

uh excel better in a particular task right uh so here for example if you will

2:16:30

uh do this fine-tuning uh you can you can give the model right

2:16:39

uh um uh pairs of prompt completion and I'll

2:16:46

give you an example uh just disregard the code here but let's let's try to

2:16:52

understand what's happening giving the following view and then here's a review body so you will insert this review bo

2:17:00

your your your review body here predict the associated rating from the following

2:17:08

choices one being lowest and five being highest and then the answer choices

2:17:16

um will will come here so you have a pattern of prompt completion you will

2:17:22

insert your your prompt your re your your text of the review of a movie for

2:17:29

example and the answer that the the the model should give is the answer choices

2:17:36

here so you you're you're you will feed in this thing right this is done

2:17:42

programmatically and it's not the scope of the course but you can do this via um

2:17:48

uh API as well or uh in the case of open AI you have the playground where you can

2:17:56

do this um feeding some documents to the model that contains the this instruct

2:18:05

this prompt completion template And you will do this for several

2:18:12

examples several examples and you will train retrain the weights for several epochs right

2:18:20

um you can train for text generator generation for example generate a star

2:18:27

rating uh uh about this product give a short

2:18:33

sentence describing the follow product for a review this is the text summarization so you can see that what

2:18:39

what's happening is you're giving an instruction right about something about

2:18:45

a a a piece of prompt and you're giving the answer that should be be given

2:18:53

okay um there are you can prepare your own

2:18:58

data set to fine-tune um and you can also

2:19:06

uh find fine-tuning templates uh that computer scientists and uh

2:19:14

programmers they they take large amount of text for example Amazon reviews uh

2:19:20

that are available and they and they thank God they do this automatically for

2:19:26

us to kind of separate several several instructions and the the right answers

2:19:32

that you can use to fine-tune your your LLM okay um but you can again you can

2:19:38

fine-tune your LLM there's an example that I like to give which is from

2:19:43

um auto insurance and uh

2:19:49

um auto if you if you ask the um LLM um that your your um auto insurance

2:19:58

um if you if you ask some question about a summarization

2:20:03

um the LLM will try to summarize but there are lots of

2:20:09

um little details right that a human would do because it's it's a it's a

2:20:17

um it's kind of a a niche so you could give this tasks this this summarizations

2:20:25

for example um for policies and and uh other other

2:20:34

kinds of requests uh as examples right as the human being

2:20:40

uh thing and then your LLM will get better in that particular task of summarizing the thing

2:20:47

okay again you can do this you can avoid a catastrophic forgetting or overfitting

2:20:52

by doing multitask instruction fine-tuning uh we have all these templates right

2:20:58

they uh that you can use you can do it with your own data but for your own data you

2:21:06

will have to have lots and lots of examples okay because these models are large and

2:21:12

um you you will have to have lots of examples to train and to to twist the

2:21:17

knobs of these models right but in the end of the day um when you do multitask

2:21:25

instruction finding tuning you get better at catastrophic forgetting uh uh even if you're doing a full uh

2:21:32

fine-tuning you will have the computational burden for doing it multi multi- times multiple times but you um

2:21:42

having several tasks being trained means that you're you're not focusing in one

2:21:50

specific task and and you have less uh risk of overfitting

2:21:57

um we we there's a library of all this uh uh prompt um completion

2:22:06

uh data called FLAN and uh when when when you use FL to

2:22:13

fine-tune you you you actually call for example FLN T5 or FLN GPT or FLN BERT

2:22:20

something like that so when you see FL it means that that model was fine-tuned

2:22:26

um in this in this this libraries that we have for fine-tuning

2:22:31

okay uh another very important fine-tuning is um it says instruct

2:22:39

fine-tuning but is what we call uh RHF reinforcement learning from human

2:22:46

feedback so the way we um the the goal of our RHF

2:22:55

is to um create this prompt

2:23:01

completion pairs in order to maximize the helpfulness or the relevance of the

2:23:07

model minimize harm or avoid dangerous topics you can imagine that LLMs are

2:23:15

trained in the in websites so language is not something

2:23:21

um that you have guard rails and um fine-tuning it to have guard rails is

2:23:29

very important right so to avoid for example um harms that this LL

2:23:40

lm can um um generate

2:23:45

the way we do this is we use actually uh we don't use um

2:23:52

pairs as I said um you can actually fine-tune with with some pairs but

2:24:00

um the way it's done it's by reinforcement learning so if you

2:24:06

remember week one we discussed reinforcement learning as having

2:24:12

um as having the um an a an an model

2:24:18

right uh doing something and having a feedback

2:24:23

uh a reward feedback so if the model is um is aligned with human

2:24:32

preferences if that completion is aligned with human preferences then it will it will receive

2:24:40

a a a higher reward and if the um if

2:24:50

the completion is u very distant then it it receives a a low a low reward or even

2:24:57

a punishment so you start with the pre-trained raw model and

2:25:04

um then we can um gather human feedback on several of

2:25:12

of prompts that we are generating right and if they align with human feedback we

2:25:20

give this model a positive um a positive

2:25:28

uh reward all right so let's talk about rag rag is one of the most um exciting

2:25:35

technologies and if you go back here you can see that we use prompt engineering then we fine-tune if we want that to get

2:25:42

better into a specific task um we will see human feedback in a in in a

2:25:49

bit but after you have done all of this you you will you can augment your model

2:25:56

and build applications so so rag is here in this part

2:26:03

um and uh let me find rag

2:26:09

again so we saw that models can have difficulties so for example who is the

2:26:15

prime minister of UK boris Johnson this is out of date and because we have a cut

2:26:20

off for training dates for the large language models what is a division and will it will it's it's not doing math

2:26:28

it's kind of predicting the next token the next and each number here would be a

2:26:33

token uh and it it's it can get wrong right so um and we can also and and and

2:26:42

LLM can also hallucinate right like what is a Martian

2:26:47

tree a Martian is a type of extraterrestrial plant there's nothing in Mars there's no life in Mars so his

2:26:55

this the LLMs are hallucination hallucinating so hallucinating is when

2:27:01

they answer things that are not plausible even if they don't know the answer okay So for example the the math

2:27:09

we could tackle using prompt engineering and and uh give give a few short

2:27:15

inferences um explaining the process of division and

2:27:21

um and chain of of thought we could also fine-tune in math

2:27:27

tasks we you can do that uh the and so the outofdate thing we can

2:27:35

also insert new information in the prompt okay but the the re the developers they

2:27:45

came up with a new way to insert new

2:27:51

information in in these models automatically right so this is this is

2:27:57

rag a retrieval augmented generation so involves retrieving informations from an

2:28:02

existing database that might be relevant to the question or prompt and um this this information will

2:28:12

give additional context to the large language model okay so what happens with

2:28:17

rag is this you have uh a user prompting and now you have a um a LLM

2:28:27

that has another piece called the orchestration library but the orchestration library is responsible for

2:28:33

access assessing different data sources it could be documents wikis expert systems

2:28:40

web pages books specific books that you that are related to um your

2:28:47

prompt and what what is going to happen is this orchestration library will take

2:28:53

your prompt will will make a consultation to these external databases

2:29:00

and will retrieve important information right and then it will input

2:29:08

this important information will pass this important information to the LLM together with

2:29:15

your prompt so it's giving you um it's giving

2:29:21

you a a uh a prompt with more context

2:29:26

but it's doing this automatically one example of this that I like to to

2:29:33

um to talk about is interviews with books or um so you can have you can

2:29:41

interview a book for example you can um put the the book the PDF of the book as

2:29:48

an external database and you can interview the book and ask questions for

2:29:54

with uh for the book where the the pieces important pieces of information

2:30:00

related to your question are retrieved by the orchestration library and given

2:30:06

as context to the LLM um and this is uh

2:30:11

this is very nice but you have to be careful because um the the size of the

2:30:17

contest is limited so you have to I mean there there are

2:30:23

details to kind of implement this uh there are several tools such as langchan

2:30:28

that kind of do uh summarization and and break the information into pieces uh so

2:30:35

that you can pass to the LLM um so this is this is

2:30:42

um a plus that you can do to your model to uh for

2:30:48

example um try to overcome the outofdate

2:30:54

thing or to um to give more information to the LLM

2:31:00

without training on tokens right specifically on tokens

2:31:06

now for this this the rags the rag framework is also good to uh avoid

2:31:12

hallucinations right because you're asking for something that um it might

2:31:18

not be it's not seen right because it's out of date or because you're not you're

2:31:24

not giving the document if you're not doing rag and then the system will

2:31:29

hallucinate it will just come up with an answer that is totally nonsense so rag

2:31:35

is good to prevent hallucinations as well um and uh and uh here's an example

2:31:44

let's say that you you were you wanted to ask who is the plantiff in case da da da so you have a you have a

2:31:51

document right uh a PDF for example of a case and uh you want to ask questions

2:31:59

about that right so this is a uh identity retrieval entity retrieval so

2:32:05

you're asking something um about this

2:32:10

um about the PDF right and this is the context you can see that the blue part

2:32:16

is the is the part that the orchestration library got so it it went

2:32:22

to the PDF it found the case and it extracted the uh the the the paragraph

2:32:30

that is uh related to this uh question and then all the your prompt and the

2:32:37

blue part is passed to the LLM okay uh finally uh how do we evaluate

2:32:46

large language models because um large language models are

2:32:52

uh for example if you ask is a is a is a cat

2:32:59

um is a cat a um

2:33:06

um I don't know so or or list animals that are um um mammals right and the LLM comes

2:33:15

with 10 or 15 different uh names some some comes with names uh first and some

2:33:22

with uh other names or if you ask for a summarization where the LLM says

2:33:30

um you know like the summary of this paragraph is that this person loves DVDs

2:33:37

and the other LLM says the summary of this uh

2:33:43

uh of of this paragraph is that the the the person um adores DVDs right so how

2:33:51

how can you kind of uh evaluate this it's very different from supervised learning uh classification regression

2:33:58

prediction uh models where you have the ground truth right you you know what uh the LLM

2:34:06

should uh uh um uh output and there was

2:34:12

no space for uh creativity right so in the end of the day there are several

2:34:19

metrics uh the most important one is rouge the other one is the blur

2:34:25

uh score you can see it as blue as well but blah um and

2:34:31

um what they do is that for example rouge it compares a summary generated

2:34:36

from the LLM to one or more reference summaries that were generated by humans

2:34:42

um and uh and and come up with a metric on precision and recall for this models

2:34:50

uh blur as well is used for uh more used for translation and it compares the the

2:34:57

translations generated by LLMs to uh human generated

2:35:02

translations um there are several benchmarks that we say like benchmarks

2:35:09

are data sets where um you perform you

2:35:14

evaluate these metrics in several uh uh NLP data sets um for example super glue

2:35:24

uh big bench big bench hard and light so you have

2:35:30

different types of benchmarks And by doing this so here's the and and

2:35:36

this benchmarks they um when you train your model when you evaluate your model

2:35:41

in this tasks that are this this um

2:35:48

um this group of tasks right when you evaluate your model here you will be

2:35:54

part of they they publish a leaderboard so which model is kind of um having

2:35:59

better scores better metrics in this benchmarks right um there's Helm as well

2:36:06

we call it a holistic evaluation of large language models um this this is a

2:36:12

kind of a framework for evaluate evaluating several metrics in several

2:36:17

benchmark uh um

2:36:24

benchmark data sets as well in the end of the day if you're doing a specific task you should analyze the uh large

2:36:32

language model uh after fine-tuning after using rag after doing prompting

2:36:39

prompt engineering and seeing if that is uh suiting your needs um but you can

2:36:46

also but the um you can compare different models even

2:36:52

the pre-trained ones you can come here and compare and get and start from getting a model that is more suited for

2:36:58

your needs or that scores higher for example okay now I want to introduce uh

2:37:05

brief discussions about video and image generation and also

2:37:11

um agents in agentic AI so for video and image

2:37:16

generations um today we have tools um that we can

2:37:22

access and different models that we can access to generate image from text so

2:37:28

you give a prompt and that generative AI will generate you an image it's very

2:37:35

very fun to play with um you have for example Dollali dolly is from open AI uh

2:37:41

Mjourney um stable diffusion that is from stability AI all of this tools

2:37:50

um they are available and you can access them okay

2:37:56

um so uh the the models the models behind this

2:38:04

tools for image generation are um are what we call

2:38:10

diffusion models here's it's not stable diffusion this is a commercial name oops

2:38:16

sorry this is diffusion

2:38:23

models and I'm not going to go into uh too much details but diffusion models

2:38:30

are the models resp the the the models responsible for

2:38:37

um uh for generating email uh images right

2:38:46

um you can also So uh generate video right so here are some tools that you

2:38:53

can test and uh it's also some diffusion models uh modifications of this this

2:39:02

underlying frameworks underlying models and the the thing is that these models

2:39:07

they're based on fluid mechanics and the diffusion of liquids I mean it's a bunch

2:39:13

of equations that they kind of um based theirelves to come up with this

2:39:20

models But the thing that is important to know is that this models they have uh

2:39:27

a a encoder part uh and uh a

2:39:35

um contextual understanding just as uh language models right you actually have

2:39:43

a text as a prompt and then you will um you will understand what you're

2:39:50

asking and then the diffusion model will be guided by this understanding so for

2:39:57

example um you can say that you want a um an

2:40:05

image of a dog with its tongue hanging and um what the first thing that

2:40:11

is going to do the the models are going to do they will use it this this this

2:40:16

text encoding um to get the the vector embedding I

2:40:22

mean this this this understanding of the repres representation of the text of the

2:40:28

meaning of the text and then the model will

2:40:34

initialize a a noise image noisy image

2:40:40

so it's completely noisy image that means nothing and then the AI will

2:40:47

remove this step so this is step one this is step two it will gradually remove the noise

2:40:56

guided by this context by the meaning of the text that you gave and it will come

2:41:01

up to a result um and for for videos um

2:41:07

this is also the same but for um more time frames right and at each time you

2:41:14

do this it will come up with a new a different image um

2:41:20

and the way it uh it it works this way and in the underlying

2:41:28

um as a learning task what the models does is it it uh it it

2:41:36

relates the generate a dog with a thong hanging with this

2:41:43

image and it starts um adding

2:41:49

noise and up until a point that you don't you cannot see it's just too noisy

2:41:55

so in the end of the day you learned how to associate a text to an image that is

2:42:02

vanishing with noise and you learn how

2:42:08

to Oh sorry uh you learn how to do that and

2:42:14

you just keep doing it and you learn and then like in

2:42:19

inference time you reverse the process right so you you start from a noisy

2:42:26

canvas and guided by that text you start uh uh removing the noise and kind of

2:42:34

going backwards in um in the amount of noise and generating an image okay so

2:42:41

it's fun playing with you can generate lots of things you can generate um

2:42:46

different videos and um and even avatars right

2:42:55

uh and and generate uh lots okay so to end this week content I'm going to talk

2:43:02

about a AI agents and agentic AI uh

2:43:07

uh so um aentic AI right so agentic AI

2:43:15

is um is an area of um generative AI

2:43:23

where is basically AI that can go and act with

2:43:29

agency so as part of the AI

2:43:35

um you have AI agents so these agents are working with agency so they are um

2:43:43

they have a autonomy and they can go and they can try to problem solve and react to

2:43:50

different situations and act in the world um by interfacing with uh um with

2:44:00

uh interfacing programmatically with other softwares and uh even with

2:44:06

robotics right so um Agent AI and LLMs

2:44:15

um they uh the the basis of Agent AI is LLMs right um

2:44:23

so we what what we want is to really create generative AI that can go and

2:44:30

interact with the real world take decisions and respond and adapt right

2:44:37

um and I'm I'm uh so so let's think about this in a the normal generative AI

2:44:44

the way you it's uh you use it we use it every day um it's it's more reactive you

2:44:52

prompt the the AI to do something for you to summarize a text to generate an

2:44:59

image a music right so it helps you to create to review documents ments to

2:45:06

create documents to refine um things and and to be creative but you

2:45:13

are in charge you you give the pro the instructions to um your AI okay

2:45:24

um now when you might have used ChattyPT or

2:45:32

other LLM as an AJ agent an AI agent before but you weren't paying

2:45:38

attention to it like you weren't seeing that way um with with a a Gent AI the AI

2:45:46

is proactive so your prompt is not gen

2:45:53

in in the end you can generate something but it it will be filled your pro your

2:46:00

initial prompt will be filled with actions and the AI will perceive the

2:46:07

environment will decide what actions it will do um

2:46:14

will um execute those actions will pursue perceive the the will learn from

2:46:20

it and we'll perceive uh the envir the environment or the

2:46:25

results of that action in a loop right [Music]

2:46:31

so for example um I don't know what I have why I have this figure

2:46:38

here um let me take this out so um let's say

2:46:44

that I um differently from asking the agent to summarize a text I or to to

2:46:52

generate an email thanking my boss um I I ask the the AI the LLM to uh

2:47:02

organize a conference okay and it's it's it's kind of a a more intricate task and

2:47:11

I say "Okay I want you to organize a conference and on

2:47:17

um on uh uh I don't know

2:47:23

uh civil civil construction and help me do that or do

2:47:30

that um to me." Right so it's it's a

2:47:36

intricate task and the AI will have to break that into to into

2:47:42

actions and first of all the first action will oh the the AI will use its

2:47:48

reasoning to kind of create this uh chain of thought

2:47:56

um to kind of get this accomplished in tasks so the first task will be uh I

2:48:02

need to understand the conference requirements first of all right to see

2:48:08

uh what uh what what kind of um uh things that the conference needs

2:48:15

after I did that I should research available parameters

2:48:20

uh matching this requirement so the number of people attending right so I will research available uh venues uh for

2:48:29

that and after I do that the the LLM is deciding what's being done next i am I

2:48:37

am actually an observer of this and I will be only um I will be only needed

2:48:44

for specific questions right for specific um when when I need to answer some

2:48:52

specific question that is required to complete an action and you can see that

2:48:57

these actions the the agent will have to access uh external websites uh u uh

2:49:06

booking platforms for hotels and for venues and so on so the agent can act in

2:49:14

the world right it it can act in the world by interfacing with with APIs and

2:49:22

different pieces of software and applications um and in the end of the day it will accomplish a task for you in

2:49:31

an in a with agency in an autonomous way right

2:49:38

um I'll um I'll give you here's an simulation of what's an we're creating

2:49:44

an agent here uh and this is a simulation because the

2:49:50

agent is not uh is not connected to any real world

2:49:56

uh uh interface it's not interfacing with softwares or or a robot or

2:50:02

something but we'll simulate how the the the process of an agent

2:50:09

okay so the the first thing we're going to do is um um and actually this

2:50:16

simulation can be um you can think about having hybrid or um having the systems

2:50:24

with humans in the loop so you don't need to have automation from start to finish and

2:50:30

you you never be prompted or you never participate uh into the action decisions but you can

2:50:38

have kind of humans in the loop as well okay so we start like this you are going

2:50:46

to help me cook whatever I need since you can't directly lift pots and pans or

2:50:52

get the fridge or get to the fridge you will tell me the steps and I'll perform them we will go one step at a time ask

2:51:00

me for what I want to cook so this is an is this is a an agent that

2:51:07

is a cooking it's my cooking um agent

2:51:12

right and what I'm doing here is

2:51:18

um is I'm using what we call a flipped pattern so it's it's a way it's another

2:51:25

pattern of of prompt engineer of prompting um that you flip so who's deciding is

2:51:33

the agent you're telling him what he needs to do you want to accomplish a

2:51:38

goal and you tell him to go step by step using the reasoning using the the the

2:51:45

train of thought um and he will take actions okay

2:51:54

um if this AI if this prompt was connected to a a a ro a humanoid robot

2:52:01

for example that could manage uh pots and pans um then you would have maybe a

2:52:08

fully automated agentic AI but right now we have a human in the loop just to simulate this so the AI says great let's

2:52:16

get started what would you like to cook today feel free to tell me any specific dish of any type of cuisine you're in

2:52:24

the mood for and then we go here um I um

2:52:30

the human prompts um sorry it continues that sounds like a

2:52:35

fun and unique challenge um and

2:52:42

uh where I just there's something missing here well anyways I give them a dish i give I give them I give the AI an

2:52:50

an ID okay um I ask for an

2:52:55

Ethiopian back useback dish

2:53:00

um low carb for example so the AI says

2:53:06

that sounds like fun uh how about we make a ke keto friendly Ethiopian

2:53:11

useback fusion dish featuring a flavor marinated meat served with a size side

2:53:17

of spiced vegetables here's a plan so

2:53:22

the AI is is get the AI is actually making the steps the AI is taking

2:53:29

actions actually it's giving first it's breaking from first it's breaking the

2:53:34

actions it's m u in into step so it's making a plan because you asked it for

2:53:40

you you use the chain of thought um kind of pattern as well so uh it starts

2:53:47

saying well this is the dish name uh the ingredients that give all you the ingredients and let me know when you

2:53:53

have your ingredients ready and if you have any questions so because he cannot get the ingredients he's waiting for you

2:54:01

to do it but you it could have been a humanoid robot um that has this AI in embedded in it

2:54:09

okay kind of controlling its uh arms and legs

2:54:15

so then I say well okay um um I got the uh I got

2:54:24

the the ingredients let's start okay so

2:54:31

um it it's uh it goes on and says great let's move on to the next step cooking

2:54:37

the meat so heat one tablespoon of the olive oil of um ghee in a large scale

2:54:43

well it goes on and it says "Let me know when the meat I is cooked and resting

2:54:49

and we can proceed with the vegetables." So the agent is acting and it it's it's

2:54:56

waiting for a feedback that we are simulating but it could be a software a

2:55:01

piece of software already other application sending to it and it the

2:55:06

incredible thing of agent AI is of of these agents is that they can adapt so

2:55:13

for example the feedback I gave is it's starting to burn so if the meat is starting to burn

2:55:19

reduce the heat to medium or medium low immediately you can also add a splash of water or broth to the pan to deglaze it

2:55:27

and prevent further burning here's what to do next so do the things and let me

2:55:35

know when the meat is done and resting and we'll move on to preparing the vegetables so you can imagine that if a

2:55:41

robot would be doing that and the thing started burning the the agent is taking

2:55:47

the decision the next step it's adapting its plan that I think wasn't envisioning a

2:55:55

burning but it's adapting to what action to take okay

2:56:03

um so the the flipped pattern to kind of uh trigger an agentic AI is this ask me

2:56:11

uh questions one at a time in order to gather enough information to suggest a restaurant for me to eat in Nashville

2:56:17

Tennessee tonight ask the first question so again um I'm giving So it's flipped

2:56:25

right i'm I'm saying you know break that into steps or ask me questions once at a

2:56:31

time to accomplish a goal and the goal here is to suggest a restaurant so the

2:56:38

agent will take you through the task you are not going to ask the agent oh what

2:56:44

is a good Japanese um restaurants in Nashville you're not you're not waiting

2:56:51

for you're you're not the the agent and the uh AI is just kind of giving you

2:56:59

answers actually you will you you let the agent lead you

2:57:05

okay so it's it's very incredible what this um things can do right now uh

2:57:11

obviously that agent u agent AI right now is based on kind of specific tasks

2:57:19

tasks that um uh you know that are tailored um in a

2:57:25

specific form um you can have for example a AI agents that

2:57:32

um that have that uh um can communicate with external

2:57:41

databases uh for example you you can have a agent an AI agent to help you

2:57:50

to keep your financial records so that you don't need to kind of go to a

2:57:57

spreadsheet and enter that uh one by one like you while you're looking at your uh

2:58:04

credit card statement so you can give access to an agent to for example

2:58:10

um to your statements and this this this agent will create a new you you

2:58:18

don't tell nothing uh you just say help me help me track my finances here are my

2:58:26

uh here here are my statements help me do that

2:58:33

um and and break this this in step step by step and obviously you don't need the

2:58:40

agent to show you the steps although it's very good to have the the logs of

2:58:45

what the agent is doing but the agent will see that as soon as one uh uh a new

2:58:52

row in the statement comes in uh it needs to create a new row in your spreadsheet so you have to to give

2:58:58

access to this agent to both of these systems and it will with agency will

2:59:05

decide what to do if you have for example two two rows that are equal that

2:59:12

just came in the AI could by itself decide to ask you to email you and say

2:59:20

look I'm your your agent your assistant in in financial assistant i just saw two

2:59:26

transactions that are the same was this an error did you pay twice but it's

2:59:32

doing it automatically right um and it it's it the this is trending a lot and

2:59:40

this is uh for sure the future of productivity it's having this AI agents

2:59:46

kind of doing spec this tasks now the the ultimate goal uh not goal but the

2:59:53

the ultimate like thing of a agent AI is

2:59:58

um is landing ultimately in general artificial intelligence so what is

3:00:05

general artificial intelligence is this agents they they're not just they don't

3:00:11

reason just to specific questions to specific tasks they they can reason act and and and adapt

3:00:21

um for several several several tasks like achieving a a kind of a human kind

3:00:27

of reasoning and adaptation um so that that is what um

3:00:34

general AI is um in um is defined is intended to be so just

3:00:43

to finalize this is the the the way uh we kind of describe agentic AI and the

3:00:50

agents you have a human that gives a task to the the AI um this this AI

3:00:58

generates a prompt to itself like with the steps and reasonings

3:01:04

um and it it it um it it

3:01:10

it defines an action right

3:01:16

um and uh the the the action can be in the the real world

3:01:23

can be just a you know but the agent is the computer is acting in the world and

3:01:30

it's getting feedback of what happened and then with the feedback it will

3:01:35

generate kind of a new prompt a new reasoning it it will u the response of

3:01:42

this prompt is is actually it's underlying an action and in this action

3:01:47

this computer will have to interface with um robotics with other software um

3:01:56

and so on so that's for week

English (auto-generated)

All

For you

Recently uploaded

[3:58:39](https://www.youtube.com/watch?v=ys_fN3uy7bQ)

[Now playing](https://www.youtube.com/watch?v=ys_fN3uy7bQ)

[Eli](https://www.youtube.com/watch?v=ys_fN3uy7bQ)