1.8 Transcript

Hi Professor Johar, I have a question on ChatGPT and OpenAI's token limits, and how we can set our project well within those limits.

I guess my specific question would be, we know that, OpenAI has a prompt limit, and we also have an upper limit.

So how do we know the large, document and, you know, generate a large document by working around those limits. Okay, I guess you know, the first thing is that the limits are pretty big.

So that's 64,000, 32000. Yeah, that's pretty big.

You need big documents to go beyond that. And then you also want to consider the fact that these limits will keep increasing.

You're looking at the state of the world now. And but we know that six months from now you're going to be having much larger limits.

So you can put larger documents anyway. But even beyond that, when you're using the API, what you can do is you can take your very large document.

Let's say you want to read the entire contents of War and Peace.

That's a huge document. And then you chunk it into smaller chunks and these chunks, and we're going to see this in the course as well. The chunks are fairly small.

They're 512 or 1024. You know those kind of lids. You chunk it, and then you have an array of many, many chunks and as many chunks as you want.

And then you can feed these into, you're LLM within the limits, multiple times, and in a sense, get your entire data into it.

I see so. And when I want to curate some specific parts of it, I can just focus on the really big chunks. So we see this too, we build rags, for example.

We'll be taking a subset of these chunks. You have a question. And you find the chunks that are the most, relevant to you, to the answer.

And you send those chunks, which is going to be a very small set to the LLM.

So you're not going to be constrained by the, the much larger limit that we have on them. So we see all this in the course.

So there are many ways of, getting around this or even I think getting around is probably not the right way to phrase it, but you can do a lot with these models.

So, for example, if you want to build an LLM that does nothing but answer questions on War and Peace,

Yeah. Then you can fine tune the model with War and Peace.

Yeah, I guess, and we will see this later. We will see this later on as well.

So you mean the current limit is actually not very limiting to any sort of application?

Yeah. We have, like, smart ways to get around this, not get around this.

We have a ways. to just that works already. So we don't have a problem. Yeah. I mean it's not really a limit because you're not going to be giving everything in one shot.

Thank you for the answer. I think is very clear. And then I know that the limit is actually not very limiting how we are going to learn a lot of efficient ways to work with large language models and program it.

Yeah. In the next chapter of this course, not just more efficient, but also more natural.

If you think about it. Because we have seen this in past, as the prompts get more detailed and as you give it more context, as you specify the output formatting here, the more detail it gives you.

It seems that the large language model performs better in the sense that it gives you the output that you want in a specific format, and more accurate based on the contents you give. But in practice, how do we well navigate conversion hearing?

Because to more detail, the problems are going to be that means the more tokens we are going to use and there's a cost factor on that.

So to think there is a tradeoff or.

So it's a good question. First of all, the cost is very low per token.

So if you were asking a question and you're doing it only once, then it makes, for a single use. Then it makes sense to make your prompt as detailed as possible so that you get the best answer immediately.

And you don't have to keep asking it, but it's not going to be very expensive if you it's like, you know, the cost is so low, you you barely hit a cent after ten minutes of questioning or 20 minutes of questioning or something like that.

So it's very low cost. But even if you are using one that has rate built in, you're always better off asking a more detailed question.

Then you don't ask follow up questions, and your total token usage is probably going to be lower. And you have to think about this most often when your, prompts are embedded inside an application that millions of users are using.

So a single prompt doesn't cost you anything. It's going to be super cheap. But if you have millions of people asking the same question and this is that's a multiplier effect, you're going to end up paying.

It's a lot of money. And so at that point, what you want to do is you want to make sure that your prompts are accurate and accuracy means detailed enough to give you the best answer possible as quickly as possible.

Okay. And it should be also concise, it should be to bare to just to the bare bone, though that's not the function, but also no like repetitive tokens.

Kind of. Yeah, I guess that's that's a good way to look at it.

You want to be, you want to be as brief as possible, but make sure that you don't miss out on any detail that is important.

So and, I guess to follow up on that, are there no applications or programs? Two can estimate the tokens that's going to be used in that application.

I should have mentioned that. It's a good point.

There are many applications, many programs that will help you figure out what the cost is going to look like.

So in terms of the number of tokens that you use, you know, the because the cost is divided up into the tokens for the prompt and the tokens for the completion for the response.

And usually the prompt tokens are cheaper than the completion tokens.

So you want an application that can figure out, estimate from your prompt that you can see the prompt and fill the tokens,what the completion will look like and what the cost will be. And they are applications that do that.

I see. So I guess that answer my question. So first the tokens won't cost that much. And even though cost management is important to tokens they don't cover much.

So detailed tokens a detailed context. It just improves your model and eliminates the need to second ask it for anything.

And there are applications to help me manage the cost for my large language application. So that should, you know, keep that trouble out of my mind.

And it's good to point out that if you are giving a prompt that is has all the details that you want, you're going to get a better answer. Yes. So irrespective of the cost, you want the, you know, the prompt to be able to predict anyway.