0:00
Markus Brunnermeier: So, nice to see you all. Thanks a lot for coming back and joining us for another webinar organized by Princeton for everyone worldwide. We're very happy to have Sergio Correa with us from the Federal Reserve Board. Hi, Sergio. Hi, how are you doing? Sergio will talk about unlocking economic data with LLMs, so large language models like ChatGPT, and you know they are changing how we do research, in particular empirical research, and hopefully we will learn a lot from Sergio today. And let's start with the poll questions which you gratefully agreed to give us some feedback on. So, the first question Sergio put forward is, will open source LLMs compete with closed LLMs? And to what extent will they compete? And will they catch up? And the answer for yes is almost 60%, 59% thought yes, only partially 30% and not; they will be significantly behind the closed LLMs, the open source LLMs, like META opened up its LLMs and I think GROK was opened up too by Elon Musk. Now the second question was within applied economics, who will benefit most from LLMs as a tool to extract and create data? Will it be large research groups that hire developers or small teams that were previously unable to hire RAs and pre-docs? And the answer is essentially one third large research groups and two thirds, the small teams who are unable to hire RAs – and essentially LLMs are a substitute for RAs. Third question is what's most important for state of the art LLMs in order to improve them further? Is it more training data we need, more papers to feed in, social media and so forth? 28% thought this way. Better models, better transformers, so attention is all you need, 27%; or large computing resources, CPUs and TPUs, 25%, and AI alignment with human preferences. So essentially, it's almost split. It's essentially 28%, 27%, 25%, 20%. So it's essentially evenly split across the answers. And the final question was, when will artificial channel intelligence take over AGI? Is it 2025? 13% thought yes. 2040, 48%, almost half thought in 2040, we will be there. Only in 2100, so in almost 80 years, it's 18%. And never 21%. So almost half thought in 2040, we will be there. Otherwise, it's roughly equally split. And less likely that we will make it in next year 2025. So before I pass on the mic to Sergio, let me just refer back to an earlier webinar we had on ChatGPT just almost a year ago with Kevin Bryan. I think it's still live and it's still on YouTube. I just want to advertise it, I think it's still worth watching. Everything is still very valid even though many things have happened in between. So I pass on the phone or the mic to Sergio and he will talk to us today about the LLMs and how we unlock economic data.

Sergio Correia: Thank you very much. Can you see my slides? Perfect. Great. So first of all, quick disclaimer, everything I say here is of my name only. Any reference, any tools is not going to, doesn't represent an endorsement and of course take any code snippets or examples not at face value. Please review them, if you attempt. Now let's start. There've been many people who have talked about the LLMs, such as Markus here. I want to focus on something that is a small niche, but very important, in my opinion, which is how to create data that can be used in applied economics. And through the talk, by the way, all the images are also created by artificial intelligence tools, like ChatGPT, for instance. So you're going to see a bunch of typos in the images. Let me know if you spot any, like here, where the S is uppercase instead of lowercase. Now, this talk, again, is not general advice on LLMs. There's a very good paper by Anton Korinek on this, where he explains how ChatGPT can be used generally for brainstorming, for proofreading your papers, for coding, et cetera. I'm not going to delve into that. I'm only going to delve into data and research data, not policy at all. And I'm going to, I want you to finish the talk with two main messages.

5:02
I want to transmit two pieces of information. First, I want to tell you why: Why we should care about LLMs for this. And I want to say that LLMs can be quite useful to create data that is novel and can be applied for research. And the other part I want to say is how. Because even if you're convinced, you may think that the barriers of entry, what do I need in terms of coding, et cetera, may be quite high. But actually, they're quite useful, quite simple. A simple Python, an elementary Python user may be able to do this in an afternoon, for instance. Okay, now let's move on to our first fact, the why. As you may probably tell your students, a good empirical paper in economics or in finance needs three things. It needs a question. It needs the economics part. It also needs a clean idea, identification strategy, RDE, instrumental variables, randomized controlled trial, something that allows you to tease out the answer for this question. And lastly, it needs a data set where you can test number one with number two, where you can try to answer the question cleanly. These are the three steps you need for an empirical paper. If you have two of the three, you don't have a paper. So the mistake I remember seeing when I was in the PhD and also afterwards that some students make is they try to write an empirical paper using just standard publicly available data. Think about Compustat or whole reports for banks. And the problem of the data is that many smart people have looked at this data over the years. Think about the chart on the image on the right where the whole terrain is barren. So it's hard to come up with something new with just Compustat, unless you have a new theory, are you testing a novel identification strategy? If you are not doing any of this, it's going to be difficult for you to write a good paper. So this is where these tools enter. So a way to solve this problem is to get a novel data set. And at the Fed, for instance, we often use confidential data sets or people get agreements with companies and get private data sets. Something I do a lot is use historical data sets that haven't been digitized before. And this other third way is to use non-traditional data. Now, this doesn't solve the whole problem. This doesn't solve economics. This doesn't solve the identification strategy. But it allows you to ask better questions that couldn't be answered otherwise. Now, just to show you a few examples of what I mean with non-traditional data. There's this very nice JME from less than a year ago that looks at the faces of Fed chairs when they're giving testimonies. And they control for the words they say. They control for the policy actions, and they focus basically on the facts. Are you nervous? Are you confident? Et cetera. And just using the faces, you can actually derive a right-hand variable that then you can use to run a regression and see what are the effects of these reactions. Another one that is, I feel, incredibly creative. These are also very recent QJE looks at the children's textbooks and tries to infer the race and the gender of the people, the images depicted in these texts. And what they do is they detect the faces first, then extract the colors, and then match these colors with a given race, with a given age and gender. And they use this information to test for their stuff. So again, something that you couldn't have thought otherwise, but you've just created data where you could have imagined that there was data before. Last example, for instance, something that is even more recent, Little and Bybee and co-authors and Jules von

Bismarck have been writing a lot about how to get data from historical newspapers, for instance. So think about the Consumer Sentiment Index, the Michigan Consumer Sentiment Index.

9:29
It doesn't go back in time 200 years, but you could make it go back in time by forecasting it in the past based on news or what's the journal articles and the sentiment embedded in these articles. And you can also do this locally, for instance, using newspaper articles, specific cities or states. So you can create Consumer Sentiment Indices by state, by city, across time, for other countries, where there was nothing before. Again, this is another way of creating data where before you only had a text, an image, or a picture. So this is basically my first insight. You can use AI to unlock data that you couldn't otherwise. So think about this iceberg, for instance, where the top is the tabular data that we all know, that we don't know from FRED, et cetera. This is the data that can easily fit into Stata. And what we want to do is take this data that is hidden under the water and move it up so we can actually run a regression on it. Now, practical examples of how this happens, for instance, for every regulation that gets proposed and will be taken into action by any of the agencies in the U.S., for instance, and in Europe, the request for comments on different participants and the public have comments on these regulations. And these comments are important because they can tell you potential things that can go wrong with the regulations that we haven't thought of. So they need to be understood and read carefully. If you, as a researcher, want to understand how people react, how interest groups, lobbyists, et cetera, react to these comments, you're going to face a very obvious problem just by looking at this image on the left. You're going to see 33,000 PDFs on a rule on the National Wildlife Refuge System, not even like a rule on something even older with more time for comments, et cetera. This is a very recent proposed rule, and it has 33k comments. How can you understand that? How can you read them? How can you make sense of them? Imagine how many arrays or interns you'll need to pass through this data. But now with a tool like ChatGPT or Kore and any of the others, you can wrap them in a for loop. You send the PDF, like in the example on the right, and then you can start asking questions about this data. Who wrote the comment? What is the opinion of the author of the comment? And what proposals does he have? So with this, you can easily create data again that you can then use to answer an economic question. Like, for instance, how do coalitions form with respect to a rule? How do different interest groups react? And what do they propose as a reaction to the rule? So there's many things you can do with this. Another example that, Markus, you may be more familiar with is employment data for a paper on the German hyperinflation. So we started gathering this employment data before we were familiar with ChatGPT. We had to collect the data by hand. It was extremely painful. But now we can easily, as an alternative, we can easily go back and redo this analysis very straightforward way, just by copy-pasting the OCR text and asking ChatGPT, please tell me the number of employees that you had somewhere in this big PDF from like almost arcane German, and ChatGPT is gonna do a very good job on this, and he's gonna tell you exactly what are the employment numbers, or tell you there's no employment numbers if there's not. This, for instance, allows you to, yes –

Markus Brunnermeier: Can you say a little bit, how big is this time saved if you do it with

ChatGPT relative to traditional ways, and how reliable is ChatGPT? Can you just rely on it? Can you say, oh, that's good enough?

13:38
Sergio Correia: Yes, that's a great point. You save a lot of time. You never completely replace the human work with ChatGPT in this exercise. What you do is you use it as a first step, first step, and then not only you extract the data with numbers, but you extract the snippet of the text where these numbers came from. So then the human reviewer, what it needs to do is review the string and verify that the string makes sense and matches the numbers, and then you are done. But this is much easier. It takes 30 seconds, 10 seconds, compared to the work of going through a very, very long PDF of 2,000 pages that may have typos, and it's very hard to read, which doesn't take 10 seconds. It may take five minutes. So in that sense, you can make more efficient use of people's time, in a way. You're not just replacing people. You are saving them the work of having to go through this manually.

Markus Brunnermeier: But is the OCR of a chat GPT as good as a top OCR model, in a sense of purely reading it in?

Sergio Correia: No. What we do is we run Google Television or Amazon Textract to get the text. You can use two tools in order to minimize errors, and then with this OCR text, we run ChatGPT. It works, but it's not as good as the other tools. It's not specialized for that. So I will, at least as of 2024, I will still use the other tools to digitize the data, and then on the OCR data, run ChatGPT to extract meaning. And this example basically is about how to extract data that is there already, to extract a number. You can also use this not to extract data, but to create data that you were not aware of. And for the same paper, something that we're currently working on is, we have data on basically all the activities of the firm. As of last year, for instance, in 1920, I think this snippet is from 1919, and it describes the recent investments and physical purchases of the firm. And if you want to know not only what happened with employment for the firm, but what physical investments the firm did, it's a bit trickier because you need to classify investments, you need to make sense of them, and it's a little bit difficult. So what we do here is I first, as a first step, ask ChatGPT to extract the list of investments and translate them into English so I can read them and not bother you and Stefan and Tom. And then once we have this in English, we can categorize it, either by a human or, again, by a ChatGPT, to create categories such as investment in factories, in warehouses, in transportation. And then you can use this as a left-hand side variable in your regression to understand how exposure to higher exposure to inflation through leverage can affect the real purchases and investments of firms, not just the accounting investments or the employment. Now, you can also use this, and we use this in our paper, for merging data. Something that happens a lot of times with historical data and even with recent data, you have identifiers based on names. So in this example, for instance, we have names of a national bank on one side, and we want to merge it with another data set, but the names often don't match. So you can have, in this example, the National Bank of Wheeling of West Virginia, and here we have four different spellings depending on the source. How do we match them? What we used to do before was by hand, and that was very costly. Then we switched to fuzzy matching that still requires a lot of handwork and tweaks, but a simple alternative may be to just send it to ChatGPT and tell ChatGPT, give me the number, the charter number that matches, and it doesn't match for you, saving a lot of time.

17:47
Of course, you still want to review it, but it's much easier to review a table that says name on the left table, name on the right table, and then see if these names make sense, than to go through a table that can't have 10,000 banks on one side and 10,000 banks on the other side and try to find a needle in a haystack. So in that sense, it helps a lot to, again, create data, or in this case, to merge data sets. Now, the second insight besides showing that you can use this to extract data is to show you that it works at scale. So none of the examples that I showed before were impossible for a good array or a good internal for a good research. You can do it yourself. The nice thing about ChatGPT is that you can open 10 command lines and run this in parallel and get 10,000 PDFs done in an afternoon, which will have taken months of work to do manually and tens of thousands of dollars in hiring people. So this basically, and I think this is a point that people haven't yet internalized. I believe this democratizes research in a way, because before, if you wanted to do an effort like this, you needed a lot of resources. You needed a lot of money. You needed to be in a university where you could hire researchers or PhD students. But now, even if you're a PhD student and you have a $300 budget, you can do a very similar work and only have to review the output at the end. You don't have to do everything by yourself.

Markus Brunnemeier: So there are some questions coming in. How reliable is it, really? So let me just put a different spin on it. Is there some evaluation studies, Marcus Leippold asked, where they go, let ChatGPT do it, and then a double check, and how good is it relative to doing it by hand? And is it like, do you think it will improve a lot? So should I, if I want to do a big historical study, let me just wait a year, because the data cleaning will be so much cheaper in one year's time. So should I do it in one year?

Sergio Correai: It's already cheap. I will say for another project, I had to go through about 20,000 PDFs and I spent $200, $300 for that, which is already cheap enough. It's not going to break my budget. So I will say it's already quite cheap, unless it's 3.5. Now, on your question about how good it is, for this other project, we had the luck that there was another group that digitized some of the numbers we wanted by hand, hiring a group of contractors. So then what I did, I benchmarked the output of ChatGPT 3.5 and ChatGPT 4 against this ground truth that people had collected. And what I found is that ChatGPT had about an 87% success rate against this ground truth of human people. And ChargPT 4 had about 95, 96% percent success rate. Now the interesting thing was that when I look at the errors GPT-4 made, maybe half of them were actually errors on the human team, because humans make mistakes as well. So I think it's quite good in general, but you need to be careful because there are several steps required to run this, and if one step is wrong, that basically means the whole thing collapses. So you need to be careful, you need to test your assumptions, look at the output, you cannot blindly believe the output of ChatGPT, which is the same thing for any data in a way.

Markus Brunnermeier: So the other question I have, which comes in from Victor Leonard, is in a sense, is the token pricing, does it make it expensive? And what's about replicability, that I can replicate it, if I want to submit it to a journal and somebody has to verify later on?

21:56
Is there also some error around hallucinations going on and then I can't really replicate it later on if somebody wants to replicate it.

Sergio Correia: This is a very good point and it's active, it's work in progress in many ways. I think two months ago OpenAI introduced a parameter called the seed. So now you can set a random seed and basically as long as you are running the same specific model like GPT-4 version February 1st, you will be able to get the same results with the same seed. One problem you have, however, is that OpenAI and other commercial vendors don't spend too many years, and are going to deprecate this model in two years, sorry, in one year. So you can replicate it tomorrow, you can replicate it in a month, but if you're dealing with a journal with a long lag, you may not be able to replicate it three years from now. Now the local open source projects are better at this, because the code is there. It's not going to change. So as long as you have the model, the weights, you will be able to replicate it. So again, it's a work in progress. Another thing you can do is actually you can run multiple models and see if they agree. For instance, you can run an ensemble on these models. You run ChatGPT, you run Claude, and Llama 2. And if they disagree, you have a human reviewer to inspect the output. A bit slower and more costly, but it gives you better guarantees of success, because the hallucinations are not at the same place. They hallucinate differently.

Markus Brunnermeier: Did you compare some of these different large language models?

Sergio Correia: I compared them by hand, in a way. I didn't run a for loop and compare a thousand observations. I tested a bunch of them. Claude 2 didn't work as well. Claude 3 works quite well now. There was a new model released yesterday by the Databricks team. I haven't released it because there's so many models being released now that it's a bit hard to keep up with this but there has been benchmarks, there's comparisons and there's, I think Hugging Face has a specific page just for benchmarking the output of models, embedding models and LLM models. And you guys can see-

Markus Brunnermeier: What is Hugging Face or what is it? Where do we find this?

Sergio Correai: Let me show you Hugging, wait, let me open.

Markus Brunnermeier: I don't want to disrupt your talk.

Sergio Correa: So this, for instance, the one I'm just opening is the leaderboard on embedding models. So here you can see that the best embedding model right now is Mistral, according to an average of about 56 data sets that had different questions. So people are working a lot on this and there's a lot of ways to find out which is the current best model. Something interesting here is that all of these models are very close.

25:05
And there's some debate about, like, given that all the models are basically trained on the same data and run on the same architectures, it's not that surprising that they're converging. So it's an open question. One of the questions in the survey that you gave, whether the open source models and the commercial fancier models are going to converge, people are debating it. And the answer is maybe because of this data, basically, implication. Are there any more questions on the first half?

Markus Brunnermeier: Yeah, they're coming in. I will just let you go on a little bit. Otherwise, it is up to too much.

Sergio Correia: Perfect. I don't want to go over time for sure. So now that we see that this is useful, I want to explain how exactly ChatGPT works, not from a CS researcher's perspective, but from an economist's perspective, if somebody who just wants to have an overview of this knows enough to be able to apply this. I have a Google collab that I'm going to share later, which basically has, you can actually click on this if you want, but I wouldn't advise it yet. But it has the code that we're going to use for this. So before going through LLMs, I want to go, I want to do step zero, which is basically how to convert text and images into something that can be useful for computers. And this is called tokenization. You have a string, and you make it a number. You create this into tokens. The most simple way to create tokens will be to just create the ASCII representation of every character. So you know, for instance, if I open Python right now, and if I ask for the character for the letter A. And I know it's 97, and the letter B is going to be 98. So in this way, there's a very trivial way to create numbers from strips. But this is not necessarily enough. And what ChatGPT does is something a little bit more fancy. I'm going to show you a little bit here and also do a demo. I took a snippet from your page, from Austan Goolsbee’s most recent talk, and I asked ChatGPT to tokenize it, which basically creates atomic units based on characters and sometimes even words, and also how this looks in terms of their integral representation. You can see here that the word in actually corresponds to the number 644. This is what OpenAI does before even starting the model. It creates numbers, and then it can digest. And to show you a quick demo on this.

Markus Brunnermeier: So by 2023, it just took 202 and then a three as a different thing.

Sergio Correia: That's an amazing. That was my example, actually. Awesome. Let me show you the, and this is one of the reasons, for instance, so actually, this is the one I had right here. So LLMs tend to be mad with math, for instance. So OpenAI, if you ask it to compute some stuff with three digits, it works. But it was chosen to, without having too many numbers, to stop at three digits, which kind of works for sometimes telephone numbers, for instance. But it means that a number with three digits only uses one token, and a number with four digits uses two tokens. And the way OpenAI chose this is to minimize, like, the more tokens you have feeding into the model, the more likely the model is to forget or make mistakes. Think of the tokens as the information you are sending the model.

29:05
If you can compress this and optimize this and send fewer tokens, then it's going to be faster, cheaper, and the computer is going to run, it's going to make fewer mistakes. So here, if you go into-

Markus Brunnermeier: But if Open-AI did this way, why did Google and Claude do the same thing?

Sergio Correia: All of them do the same thing. I think this is called a bit, a bi-pair encoding. They try to encode strings that appear together very often. So for instance, the word hello is its own token because it's very common. But probably this weird character is like eight tokens because it's very rare. And this tokenization, somebody says it's the root of all evil because it's behind many of the mistakes LLMs make, for instance. Let me translate this string into two languages. Please translate the following text into Spanish and Japanese. So I want to translate. OK, it doesn't work. So if I use this token, and if I tokenize this string, I'm going to find out that this text uses 58 tokens. It's very, very optimized. Even if it uses 300 characters, it's able to compress this into only 58 tokens. But now look at Spanish. In Spanish, it goes and uses.

Markus Brunnermeier: I don't see any Spanish.

Sergio Correia: Sorry, do you only see my slides?

Markus Brunnermeier: Yes, it says background tokenization.

Sergio Correia: OK, let me fix this then. Share my whole screen, not just my PowerPoint. Can you see it now? Yes. OK, so what I did that was not a thing is. I opened the tokenizer page in OpenAI. I copied this text and asked ChatGPT to translate it into Spanish and English. You could have used Google Translate, it's the same. And if I use this, if I count the number of tokens of the English version, it only uses 58 tokens. So it's very compact. If I ask for the Spanish version, it uses about 50% or 40% more tokens. It's less compressed. It's going to be more difficult for the LLM to work on this. And if you use a Japanese version, you're going to use 137 tokens. So more than twice the amount of tokens, because it's not trained on as much Japanese data as Spanish or English data. So it's not going to perform as well.

Markus Brunnermeier: It's all a matter of being trained well, so you can use fewer tokens if it's trained better or what?

Sergio Correia: So the algorithm that does the tokenization sees that the word hello appears a lot of time in the corpus of text it was trained on. So it says this is going to be token number one. But more difficult words are not as likely to appear. So if you see, for instance, Sergio, it's not very common, so it's three tokens. But I'm guessing John is only one token, because it's very common in the data, basically.

32:32
And this is just an input, but this is what may explain some of the mistakes the LLMs make, both OpenAI, LLAMA, all of them are. Now, one problem, we can easily create those.

Markus Brunnermeier: There's a question from Connor. If you look at the text, the in is different. So for 2023, we saw virtual data, and in a Phillips curve, this “in” has different tokens assigned to it?

Sergio Correia: Yes, because this is a great question. Look at this. If you see this string, you can see that space matters. So here, the word in has many different tokens, depending on whether it has a leading space, a training space, et cetera. So the word in, at the beginning, is going to have a different meaning that the word in, in the middle of a paragraph.

Markus Brunnermeier: I see. But does it make logical, it does make a logical distinction whether it's in a certain time dimension, like a year, or in a certain building, like in a location. It cannot differentiate between these two things, logically.

Sergio Correia: No. And that's what comes next. So this is useful, because now you have a vector of numbers. But this vector of numbers has two problems. Every string has an underlying different length, so you don't have equal-sized vectors. And they don't represent any deeper meaning. They don't represent context or anything. They're just pure translations. Now for that, we're going to go into embeddings. And embeddings are something that are not really well understood, I think, but they're super cool. So an embedding, you can think of an embedding of a way to represent meaning from these tokens. You compress the dimension into something of a fixed length, 100 rows, 1,000 rows, et cetera. And they represent something like meaning in terms of a latent space or an embedding space, kind of like principal component analysis that many economists know about. And this is what then gets used in the LLMs. Basically, the first step that the LLMs does when reading text is converting these tokens into an embedding. Now, I took for instance, some of the titles in your past talks.

Markus Brunnermeier: Yeah, I noticed the titles from the webinars, yes. Exactly.

Sergio Correia: And here I created, I am showing the first 15 rows of each embedding. This embedding has 3,000 rows because it's an OpenAI embedding. So here, what you're going to see is that you can see, for instance, some patterns. I wouldn't trust them too much. But for instance, when you're talking about Covid, it seems that the token number 4, like row number 4, is higher. When you're talking about emerging countries like Brazil, Argentina, maybe row 1 activates. Remember, Python starts counting by 0. So the first row is a 0 row in Python. But basically, you can start thinking about meaning with embeddings. And this is the first step you do, even for LLMs. I want to show you.

Markus Brunnermeier: essentially a factor, it's a component analysis, a principle.

36:04
Sergio Correia: Yes, a bigger factor. It's not three factors, it's a thousand factors. And very nonlinear because PCA is linear in a way. This is extremely nonlinear and kind of like what you do when you run random forest or something like that. So embeddings can do many things. And the economical example for an embedding is once you have vectors, you can add them up. You can multiply, you can divide them, average them. So for instance, there's a canonical paper from 10 years ago, it feels like ages ago, called Word2Vec. The key example here is that you have the word king, and you move everything into the embedding space. And then you can say, oh, king plus woman minus man equals queen. So you can start doing these things when you're thinking in terms of embeddings. You can also compute distances between texts. You can say, oh, I have this text that talks about Covid and this other text that talks about something else. Are they closely related or not? You can also classify the title, strings, pictures, et cetera, based on this. You can run PCA, which is what all economists know. You can do clustering. Or you can do fancier things like UMAP, which is basically state of the art. And you can run it in one line of Python.

Markus Brunnermeier: What is UMAP? So I didn't get this.

Sergio Correia: UMAP is uniformly. You know what? I don't remember. Uniform manifold approximation and projection for dimension reduction is a state of the art way to approximate.

Markus Brunnermeier: So if I look at this UMAP on webinar titles, what can you tell me on this color coding and the cloud?

Sergio Correia: Yeah, I did this basically to be sure I was able to run UMAP and it was not too difficult. So the blue titles are about COVID and the brown titles are about inflation. And basically I wanted to see if they were actually grouped together or if UMAP was giving me nonsense based on the titles. And there seems to be a common pattern. Now, the interesting thing is that the titles about inflation, COVID, that are kind of in the middle are because you're talking about, oh, COVID and Brazil, COVID and something else. So they're talking about multiple points. And of course, you could do a better job if you go beyond titles and you go into the transcripts of the webinars, for instance. But this tells me, for instance, that it works. And the other example I have here is a subtle example. I picked a list of words from every language for the word water in blue and the word hello in red. About 100 languages and how they are represented. And I ran a principal component analysis on this. And I wanted to see if embeddings with PCA were able to distinguish the words. And indeed, it's able to distinguish that on blue, basically, it's able to separate the words for water and the words for hello, even if they're all in different languages. Because OpenAI is trained on multiple languages. If you run this on other models, it may not work. Now, basically, what I'm saying here is this is very useful, even without running an LLM. And this is very easy to run and embed. And something you can even do is you can take these vectors, and you can copy and paste them into Stata, or your pool of choice, like R, and then you can run –

39:47
You can run any modern tool, like Elastic Net, Random Forest, and operate directly on the embeddings, like variables x1 to x1,000 as right-hand variables, and then try to understand something else. It can happen.

Markus Brunnermeier: So how do you interpret my coefficient then? I mean, it has a meaning.

Sergio Correia: You don't, but you basically can say, can we predict Amazon scores that go from one to five based on the comments they give? Can you predict stock performance based on the answers that CFOs give around earning goals? You don't go into the details of the exact answers but you want to see if there's some level of prediction, for instance.

Markus Brunnermeier: What about causality?

Sergio Correia: Not in this, not with embeddings at this level. You can use causality when you basically add extra layers, saying some sort of temporal dimension, for instance. Now, let me, at the risk of going too much in time, this is a very simple example about embeddings done by Andrej Karpathy. He just took the summary of every movie from Wikipedia, built an embedding, and then for every movie, it creates a list of closely related movies. So here, for instance, let me see if there's something fun, the movie Volcano. I have no idea about this movie, but it's going to tell me other movies about volcanoes, but at the same time, all other weird romantic comedies. You can ask Jurassic Park, and it's going to tell you other movies, not just related to Jurassic Park, but to dinosaurs, or things like that. So basically, it's a very simple way to do a quick approximation. Think about Netflix recommendations. You can do this in five lines of Python.

Markus Brunnermeier: Netflix recommender systems, all recommender systems could be built on that, in a sense.

Sergio Correia: Yes, but they're also built on what people like and what people watch. Yes, yeah. From this, and yeah. OK. Now, let me show you a quick demo. Now here, how do you get an embedding? So this is Python. I'm going to run a quick Python script. I'm going to import OpenAI, the package to be able to call OpenAI and get embeddings. And I'm going to import the standard data manipulation tools, NumPy and Pandas. These are very standard. First, I write a quick function called getEmbeddings that I send some strings of text, and it's going to return me the vector embeddings. And I'm going to run this for the words employment, inflation, stocks, and bubbles, for instance. And here, I'm going to run this and see what happens. And then I'm going to get, for the first word, the different numbers I get from minus 1 to plus 1. And I'm going to see that it's an embedding that has 3,000 rows. This is not particularly helpful to use it yet. But then I can do something else. I can compute the distances. So for instance here, I create a cosine similarity, basically the similarity across two vectors, is the dot product of their normalized, of the vectors normalized, and then I can use this similarity, do 1 minus it, and get the distance between the two vectors.

43:25
So now I can get the distance between these four words, for instance. And I can do this quickly, and I'm going to see that the word most closely related to bubbles is inflation here. So I thought it was stocks, but anyways. So stocks and inflation are mostly closely related, and then employment is not as related to the word bubbles.

Markus Brunnermeier: : So what do you use for that, that uses all of OpenAI, all or just from the webinar series, or what?

Sergio Correia: Oh, this just calls OpenAI and gets their embedding. It's just the pure word on this. And you can do any word, and it's going to get the embedding of the words. Now, of course, this is not perfect. I'm going to skip this part. But basically, there's many dimensions in an embedding. Like, it represents the language. It represents the tone, optimist, negative, future, past. There's many ways there. So if you want to compute a distance between a sentence in Spanish about the future and a sentence in English about the past, it's not clear which of the dimensions is going to get picked by the embeddings. So this is basically – you need to be careful with these embeddings. Because otherwise, you might get distances across the wrong dimensions.

Markus Brunnermeier: Just to be sure, if I do this with OpenAI and I do it with Google Gemini, will I get similar things or will it be very different?

Sergio Correia: I tried with a more simpler model called Bart, or Sentence Embeddings. And it was quite similar, but I haven't tried with Gemini. I don't even know if Gemini allows us to get embeddings. I'll need to check. I did something else, for instance, just to have a sense. I opened the list of ECB speeches, like a personalized data set about the ECB speeches over the last years, and copied a bunch of statements about inflation. Raises the probability that we are also underestimating inflation today. Inflation will return to 2%, et cetera. So given these statements, suppose you want to classify them into statements about inflation being high or inflation being low. The way you will do it is basically – the way I do it at least – is I create two benchmark statements saying inflation will be high, inflation will be low. below and then ask ChatGPT to tell me which of these five statements is closer to higher low. And I can run this and it's going to tell me, after doing all these employment inflation stocks, it's going to tell me that, for instance, raises the probability that we are also underestimating inflation is high, because maybe inflation is going to be high, but inflation will return to 2% represents a low expectation. And this is something that you can actually even do in ChatGPT. If I open ChatGPT and ask this question, classify these statements into high or low, copy paste these statements, I'm going to get the same answer. So just with embeddings, I'm able to replicate some of the stuff that you do with ChatGPT. So in that sense, it's me telling you embeddings are still useful, even before going to ChatGPT. Now, moving away from embeddings and into the last 10 minutes in the part that is more interesting, let's go into talk about LLMs.

47:13
So LLMs, basically, you can think about them as the next part of an embedding, like the natural succession of an embedding. And they are kind of like the auto-prediction you have in Gmail, in WhatsApp. They're going to predict the next token and auto-complete it. But doing that is a bit of a disservice because they're extremely sophisticated. And it's hard to explain how much data there is there. So just as an example, GPT-4 appears to have about 1.8 trillion parameters. Think about 1.8 gigas in a regression. That's how much data, the parameters there are in the 120 layers of GPT-4. And it was trained on about per. 13 trillion tokens, about 13 trillion sealers. So it's basically using everything that you could use. It's using movie subtitles, books, papers, everything you can find online. So we're kind of getting close to the limit of what, of the available data that a model can use to learn. In that sense, data is becoming a first order concern for these models, but they have a limitation. They know all the public data, but they don't know about your data. And this is basically the reason why you need to have something on top of ChatGPT or an LLM. And this example, for instance, if I want, if I ask ChatGPT how many mergers happened between national banks two years ago, it's going to tell you an answer. But if you ask what's the number of mergers from 120 years ago, it's not going to know because it's not available on the internet. So it has no idea. Now, this is where we enter. And the way we do this is we are going to talk to ChatGPT and add context to it. And think about this prompt, for instance. And this is the prompt, I think, one of the earlier prompts I actually used to get the data on real assets. First, you have to prime the ChatGPT. For some reason, it seems like if you ask please and you're polite, you tend to get better quality answers because the training data has better quality answers for better quality questions. So you say, oh, you are tasked with classic information from an old German handbook. After the word context, I'm going to paste all this text that is relevant. Please tell me the answer to my following questions. And basically, the part on the left is what is called the prompt. And the part on the right is what's going to change for every PDF or for every form and what's going to change, is also going to make the data change, the answer change. And this is basically the entire game. Now, one thing you need to be aware of is you cannot just space the whole book, because it's either going to fail, because it doesn't have the ability to parse a big book, or it's going to be extremely expensive, because it's going to charge you by the word, or it's going to give you poor quality answers. So the game is, if you have a book and you want to have a ChatGPT answer questions about that book, you need to find the relevant paragraphs, the relevant pages, and copy paste these paragraphs into ChatGPT within Python and use that. Now, let me show you how it works. Let me move this. OK. Again, I'm going to start with my embedding function, my cosine similarity, and my distance. And I'm going to add a new function that calls ChatGPT. I'm going to call GPT4, the January version, 0125. And I'm going to ask a quick question. I'm going to copy-paste five paragraphs from the recent financial stability report that the Fed released. I'm going to split these five paragraphs into five chunks of text, compute the embeddings of these five paragraphs, and then I'm going to ask the question, how has household debt evolved recently or lately? And I want to extract the paragraph that is more relevant, and then if I do the right job and get the paragraph that is more relevant, which in this case is the one talking about household borrowing, what's

going to happen is I'm going to send this paragraph to ChatGPT and use it to answer my question.

51:50
And this is basically, including the paragraphs and everything, this is about 80 lines of text. And I'm going to run this now. And here, at first, I split the text into five chunks, compute the five embeddings, and then I ask the Python, tell me the paragraph more relevant to my question. And it's going to tell me, look, the one with the smallest distance is borrowing by businesses and households, blah, blah, blah. That's the one we care about. And then we're going to send the question to ChatGPT, and I can actually print the question, which is sometimes useful. I can run this again, and I print the question so people can see what my question is. You have been tasked to extract information from this report, et cetera, et cetera. If you copy-paste this into ChatGPT, it's going to give you the same answer that it's giving here. Now, for the purpose of time, I'm going to skip the next example where I am asking a specific question. We can take a little bit more extra time if it's needed. Here, one problem is that ChatGPT is giving me an answer in terms of text. I cannot use that in Stata. So what you can ask ChatGPT is a small different question, which is ask for a JSON output, which is the language that computers use to talk with each other in websites, and provide your response using JSON fields, where you're going to give it two answers. I want you to tell me – let me write this text so you can see. I want you to tell me your response about how has household debt evolved lately in a number from 1 to 5, kind of like Amazon ratings, where 1 is great and 5 is terrible. So I ask ChatGPT this, I send the paragraph that is relevant, and ChatGPT is going to tell me a number that I can use in my regression. And here is what I'm going to do, it is going to tell me that the answer is 2, which is relatively closer to great than to terrible. So I copy this answer into Stata automatically. I create a CSV file or something like that, and then I can continue. And you can see here, this is about 100 lines of code, including the prompt, the strings, and everything. And just with that, you already have the answer.

Markus Brunnermeier: But my wonder is, I can manipulate this a little bit this way, and then something which was not significant suddenly is significant. There's so much freedom of how to manipulate this data to nudge the data in one direction or the other. Is this an issue or not?

Sergio Correia: Well, I mean, if you have 10,000 PDFs and then you want to play with a different prompt, the problem is going to be very expensive. You cannot, it's not like when you run a regression where you include a control and then change the result. Here, you have to run a lot of code, wait an afternoon, spend $200, then get an output. And it's not as easy to manipulate, but of course, anything economics can be manipulated. That's why we want replicability in a way.

Markus Brunnermeier: Because it could be how you split it, split it in five components or in two components or whatever. It matters a lot in a sense.

55:12

Sergio Correia: Yeah, I think that's true. And here we need to have like a, like, we need to see the code, we need to be able to replicate. These are the usual questions we have, which is the same thing, even when you're running a survey, how you ask a survey matters for instance. But yeah, but this is definitely a big concern, especially when you're doing research, where you care about being able to get the right answer and not just like some cooked up answer.

Markus Brunnermeier: There's one question in the audience, among the audience members who said, oh, sometimes research is about a construct, multiple dimensions instead of one word. What is your suggestion in that case?

Sergio Correia: I guess it puts a component... Well, you can ask multiple, you can ask a GPT to give multiple answers across multiple dimensions. So yes, you can run it multiple ways, or you can ask for multiple questions in the same procedure. Rachel asked a question about whether embeddings are convertible to text. This is only possible in some types of models. There's encoder-decoder models that encode the text into embeddings and then decode them into text. But LLMs and the ones we use by OpenAI are called encoder-only models, so they only go one way.

Markus Brunnermeier: But can I think of, you have all these complicated things, and embeddings, you reduce the dimensionality to 3,000. Is this always 3,000 something, or?

Sergio Correia: It used to be fewer, but it's a lot. And actually, I remember reading a paper that says that most human communication can be, like, once you skip it and compress it a lot, it can be made quite compact. So there's people who think that 3,000 is too much, is more than enough. It's, in fact, too much, in a way, for this. But this is an open area of research. It's not clear exactly how to compress embeddings into PCA or other tools. It's not clear exactly. There's many open areas of research. In fact, there's a paper from a few weeks ago from the Netflix folks who said that, who were arguing, for instance, that this cosine similarity is wrong. We actually shouldn't normalize this. We should just return the dot product. Because when you normalize it, you're losing a lot of, and they make, they create a toy model, a toy embedding model, where this matters, for instance, where this changes the results. And the right way is without normalization.

Markus Brunnermeier: But what does normalization just do?

Sergio Correia: Makes the vectors equal size. But that's not always what you want. In a way, here, the techniques are leaving the theory a little bit. So sometimes you do something because it works, but you're not sure why it works. So here, maybe with some normalization works better. It's probably something that you need to test empirically. Now, we're done with all the code. But just to recap, what I just showed you is called RAG, or Retrieval Augmented Generation. Basically, you read a document. You split it into paragraphs or chunks. Then you create these embeddings of every paragraph that you can use to search the document. Once you know what are the relevant paragraphs that you want to use for the LLM, you ask a question to the LLM, and then you ask for an answer, a numerical answer, a dummy variable, something like that.

58:48
Markus Brunnermeier; So do you think we will do this, splitting them into chunks in five years too, or just the power will be so powerful that we don't have to split it into chunks anymore?

Sergio Correia: I don't know. The new open AI models can work with a PDF that is 100 pages. But there's been some pushback against this idea because it's not just whether you can do it or not, it's whether you can do it efficiently. So at least now, all this chunking still outperforms digesting a big PDF. And it's way cheaper as well. So if you see the cost of this for open AI, it costs 50 or 100 times as much for every word in LLMs that every word processed in embeddings.

Markus Brunnermeier: But is it because it also saves on computing power putting it in chunks? Or is it because the pricing of open AI is somehow strange?

Sergio Correia: It saves a lot of computing power because when you're running an LLM, basically the LLM is reading the text and trying to make sense of the text in context. When you're chunking, you're just working one paragraph at a time. It's way cheaper. But we are not sure whether this is going to be a thing in a year or not. This is the way you will do it today. Maybe will, maybe won't. People way smarter than me have no idea.

Markus Brunnermeier: So to be aware, can you explain this in a practice, so that sounds a little bit scary, but I don't know what it means.

Sergio Correia: I just show you pure examples that use pure Python. So this is basically pure Python code, nothing besides the OpenAI API or LLAMA API. A lot of people are doing this within something called a framework. Basically we think of a tool that does the chunking by yourself, the PDF reading, the vectors, everything for you. The problem is that for research, you don't always want that. You want to be able to tweak the results, to peek under the hood and see what's going on. And this hood is so simple, like a few lines of code, that you don't need a framework for something that is going to be 10 lines of code. So my message here is, if you're doing this for research, work on first principles. Don't just rely on a framework that will handicap you and may create problems in the future. So just to wrap up, I have one last message I wanted to give. Writing a RAG that is trivial is easy, but doing a good quality RAG is problematic. It's not problematic, it's difficult. You need to be careful about how you extract the data from the PDFs, about adding enough paragraphs to the question you're asking to ChatGPT. All these details matter. If you do it wrong, the whole answer is going to be wrong. So you need to be aware of these details. Now, I'm going to skip this because this is what we talked about in a way. Another disclaimer, another warning is that sometimes-

Markus Brunnermeier: Oh, sorry, this was too fast for me. What, easy or hard? Can you go through this?

1:02:00
Sergio Correia: Yes. Oh, yeah. If you want to do a good quality rack that's going to parse 10,000 PDFs and do it right, you need to be careful when you're extracting data from the PDFs. That's not trivial. There's many ways to do so, and most of them are going to give you a text that is hard to read. There's many ways to split the text sentences, paragraphs, and fixed-length text. There's many embedders. Even within OpenAI, you have five possible embedders. How to search? Are you going to search for every possible paragraph and find the one closely related, or are you going to use approximate methods? How many paragraphs are you sending? How do you build the prompt? There's a whole field about how you ask a question to ChatGPT, which is called prompt engineering. And how you ask the question matters a lot. And there's another variable called the temperature, where you tell the ChatGPT to be creative, for instance. You say, oh, look, be creative. Pick stuff where you won't comment the answer. For research, that's the opposite of what you want. If you're just asking for a nice picture, you want that. You want creativity. But for research, you want the boring answer. But it also allows you to do more robustness. You can run multiple times the same prompt and ask for different temperatures and see the results change. And that way, you can make your results more robust to the randomness of ChatGPT. Also, this is new for these breaks. When I was writing the first example of this, I couldn't run the whole thing because ChatGPT had a lot of failures in this case. So this can happen to you. I was a little bit worried, for instance, that ChatGPT would fail. GPT may fail when I was doing the demo. Luckily, it worked, but there's always a chance. Because it's a new tool, and we don't have enough computing power to run what everyone's running right now. This is slightly expensive. For me, $60, which is what I spend on some PDFs, it's a lot. But this $60 represents, I don't know if you can read, but 60 million GPT 3.5 tokens, and almost 200 million embedding tokens. And that's basically, think about how much money it would have cost to ask an intern to read 200 million syllables or words. It would have been way more than $60. So it's not free, but it's quite cheap for that, given that they're not. And now, for me.

Markus Brunnermeier: So just some rough idea, if I want to do an empirical project, a student approaches me, he wants to do an empirical project, he needs some money to run a project in that space, what's the range it will ultimately cost? It's like when you buy a data set, he says, oh, I need to buy a data set, it will cost me $10,000. If I need these resources, computing resources, is it, what is it, what's the range of cost?

Sergio Correia: That I did when I passed for another project, about 20k PDFs, 100 pages, it cost me about $300 total. If you wanted to use GPT-4, which is better, it would have cost me $2,000. So if you want to do it right and carefully and with robustness checks, $1,000 may be good for a project that is not extremely large. But it's smaller, it's less than a data set, buying a commercial data product is way less than a Bloomberg license. So that's why I'm saying, in a way, this helps a little bit smaller research teams compared to larger teams. Now, we talked about this a little bit, but I just want to double emphasize, you need to be careful about the output it gives. Never trust it blindly. One way of doing this is, on your own, run this and build your own ground rules.

Like, basically, go through 100 observations, do it by yourself, and then see how well it performs, and see if you can tweak the prompts or improve the chunking or all these other different hyper parameters to get it to work better.

1:06:51
Never trust it blindly. Not when you're training it, but also not when you are, like, verifying. Always, basically, always verify the output, which is going to be way easier than creating the data by yourself. And now, just to recap, my main message is this can be very useful to unlock data. And sometimes you don't even know there's data somewhere. You see a PDF, you see a face in a children's textbook, but there's data hidden there that you can unlock with these tools. Again, I think this makes it easier for PhD students, junior researchers, et cetera, to access data that they couldn't have otherwise, researchers in developing countries, for instance. But you still need to do human steps. And basically, something I found is that combining Python with an LLM API, either closed source or open source, is a very, very powerful combo. It allows you to get answers quite quickly. And I think there's a lot of people who are saying Python, in a way, it's eating the world, because everyone is using it more compared to R, Stata, et cetera. And at least for this purpose, at least for spectral analysis, for LLMs, et cetera, I think that's true. And I hope, in the audience, I hope people, if you have any questions, let me know. But the code is already open. I'm going to make the other data set open, or the other script open. But I really encourage people to try this with LLAMA, OpenAI, et cetera, because I think we can actually get new research, creative research. We can start answering questions that we're not able to answer before.

Markus Brunnermeier: So I have two questions before the end. So you described earlier what's the current state of the art. If you were to project it now five years into the future, is it possible to project it five years in the future? How do you think the technology will change and how should we do it similarly in five years? If I'm starting a PhD now I know I have to do two years of course work and then in three years I will start to play with data. Should I learn this or should I just say no in two years it will be totally different? I have to just wait, let me focus on my…

Sergio Correia: I think it will keep improving but not as much as some people believe. So the original paper that led to this is called the attention is all you need paper this was from 2017. So ChatGPT who I think was from 2020 and basically the big improvement I think that has happened in the recent iterations is data and also a little bit like scale of like how much money you spend training these models you can like you can spend 10, 20 million dollars to train this model and you can get far. And now we're reaching a problem where we don't have more data. We have exhausted every possible data that's available. In fact, there's a famous data called the pile, where everything's people have scrapped basically everything on the normal web, the dark web, everything, and put a lot of books, papers, etc. And that's where in a way like every movie subtitle is there, etc. So how can you get more data? It's not clear to me how you get not just more data, but good quality data.

Markus Brunnermeier: So when you answer your own poll questions, you would say extra additional data is not there.

1:10:47
Sergio Correia: I think that's the most pressing concern, additional good quality data.

Markus Brunnermeier: So let me ask, finally, thanks a lot. It was fantastic. If you were to pick a PhD student who would think so of starting a new PhD and say, what qualities should I look at to pick the best PhD student who will be a fantastic researcher in 10 years time? What qualities should I look at? I probably should not look, oh, is the best in real analysis or something? Or is creativity the best? Or is it best to be a good coder? He has a good creative mind in terms of talking to computers the best? What should I look for?

Sergio Correia: Let me go back to this slide. I think three, you don't need to be that good at Python. And all students, even people who do theory, are going to take an empirical course. They're going to learn Stata. They can learn Python during a spring break. So I don't think three is going to be what differentiates people as much. At least, I mean, this year, yes. And for something fancy like a video, perhaps, yes. But ideally, what's going to matter is number one and also number two. I'm partly guilty of, like, I see a lot of papers where people just run fixed effect models. That's particular to that, but I think there's a lot of nice identification strategies, RD, basically, bunching, et cetera, that are not used as much as I would have liked to see in papers and PhD candidates. So I think the most important thing is still, one, like having a good question, and two, being creative for your identification strategy. And creativity also helps a lot for number three, to find data.

Markus Brunnermeier: So finally, sorry for picking up again. So what you see in the profession, there's a shift towards, you know, some researchers set up labs and they have 10 predocs working for them and, you know, and so do you see more as what's in the sciences, you see this lab structure, there's one professor and a lot of others underneath will do all the data cleaning. To see this business model, this evolving business model changing now, we've just moved into that. Do you think it will be much smaller groups, a very agile researcher with one or two, or just on his own, he can do as well as somebody who has 10 pre-docs working for him?

Sergio Correia: That's what I hope. That's why I'm not a hundred percent sure, but that's what I hope. I think that's true already. I have a paper with Stefan where we digitize a hundred thousand balance sheets and we basically had no RA support or other things, but thanks to advances in OCR and machine learning, we're able to do so. So that's the first time I said, oh, wow, look, you can really scale up with high quality, like improvements of this magnitude. So I hope that we are not going to have an army of people digitizing, because that's not the best use of a researcher's time. I would like these pre-docs to be doing something more creative than just validating data, actually.

Markus Brunnermeier: So I think we all should include something like this in our PhD programs to get young researchers exposed to these new tools. I'm very grateful, Sergio. We stay in touch and, of course, talk about more research. And I hope everybody of you found it useful as well. And perhaps we have to do a new one, another one, in every one year's time, because things are changing radically all the time. Yes. Thank you very much again. See you soon. Bye bye.