Sergio Correia

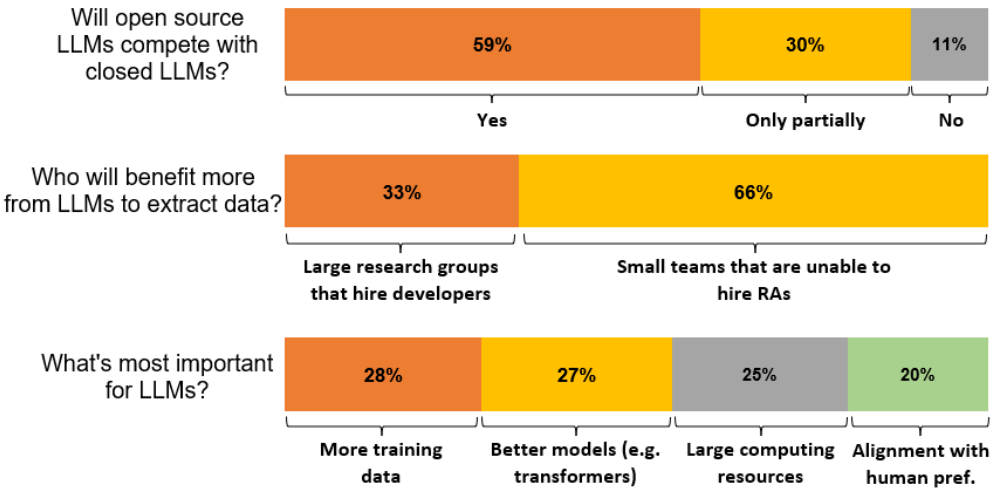
Unlocking Economic Data with LLMs

On Thursday, March 28, Sergio Correia joined Markus' Academy for a conversation on "Unlocking Economic Data with LLMs." Sergio Correia is an economist at the Board of Governors of the Federal Reserve System in the Division of Financial Stability.

A few highlights from the discussion.

- **A summary in four bullets**
  - Although barriers to entry of using Large Language Models (LLMs) appear high, in reality using these tools only requires a few lines of Python, and one need not be a professional coder to do so.
  - The talk covered simple examples that leverage LLMs to summarize publicly available text, extract historical data, or merge datasets. It also covered the basics of embeddings, and showed how these can be useful even without LLMs to classify text or run regressions (for example to predict Amazon product scores based on user reviews)
  - Lastly, the talk covered the nuances around designing a good Retrieval-Augmented Generation (RAG) pipeline to generate data with LLMs
  - All of the code from the talk's examples can be reviewed here and here

- **[0:00] Poll questions:**



- **[3:55] Why should we use AI to create data?**

    - A good empirical econ/finance paper needs 3 things: an important question, a clean identification strategy, and a dataset
    - Too often PhD students try to use only publicly available data (like Compustat), but it is difficult to come up with new ideas using just these
    - The alternative is to use confidential data, historical data, or non-traditional data. Although there is more non-traditional data than tabular data, it is locked inside text, pictures, or audio. We can extract this locked data using AI
    - For examples: Alexopoulos et al. (2023) studies the impact on financial markets of Fed Chairs' faces during congressional hearings, while Adukia et al. (2023) studies the faces in childrens' textbooks. Binsbergen et al. (2024) and Bybee (2023) generate economic sentiment indices from news articles
    - Simple examples of use cases include summarizing publicly available text (like comments on proposed regulations), extracting historical data (like digitizing financial statements), creating historical data (like quantifiable investment measures from qualitative discussions in annual reports), or merging datasets
    - It can often be helpful to digitize text before asking the LLM to perform other tasks like extracting meaning, so using additional tools like OCR technology can improve accuracy
    - A good intern can do anything that an LLM can do, but with AI we can do these tasks at scale. Going through dozens of thousands of PDFs with GPT-3.5 can cost a few hundred dollars (2k with GPT-4). This has the potential to democratize research
    - From Correia's experience in one project, GPT-4 can achieve a 95% success rate when compared to data digitized by humans. Interestingly, half of the errors the model made were due to human errors in how the data had originally been included in the document
    - When extracting data, one can ask the model to also show the snippet of text it took the data came from, making it easier for a human to review the LLM's output and improving its reliability. Since different models hallucinate differently, one can also run multiple models to compare their outputs (ensemble approach)
    - Two months ago OpenAI introduced a random seed parameter, easing replicability. Replicability issues may arise because commercial LLMs are deprecated relatively quickly, so it might be hard to replicate papers many years from now. Using open source models may be helpful to avoid this
    - Hugging Face publishes a leaderboard comparing the efficacy of different embedding models and LLMs. Considering that the models are trained on the same data and that they run on similar architectures, it is not surprising that we are seeing their effectiveness converge (including across open source and closed models)

- **[25:50] The basics of embeddings**
    - Although the barriers to entry appear high, it is actually quite simple to leverage embeddings and LLMs

    - The first step to process text/images/audio is to convert them into vectors. In the process of doing so, the model will turn words (or parts of words) into tokens.
    - The higher the number of tokens the model has to vectorize, the more likely it is that the model will make mistakes
    - Tokenizing text in languages other than English typically results in less compact representations (i.e. requiring more tokens), which may lead the model to perform less well. This is primarily because the model was trained with mainly English data
    - Vectorizing language however has two problems. Depending on the size of the original text the resulting vectors will have different lengths, while they will not preserve meaning or context. To overcome these we use embeddings
    - Embeddings are low-dimensional semantic representations of tokens, designed to capture meaning within a specific context. These representations ensure that the resulting vectors have consistent dimensions
    - One can think of them with an analogy to Principal Component Analysis (PCA), a common tool in economics for dimensionality reduction. In PCA, the latent space condenses data while retaining the essential information. The embedding space in language models serves a similar role, condensing linguistic information without losing semantic nuances
    - Although one can compute embeddings with PCA, but there are many other non-linear and state-of-the-art ways to do so, like k-means clustering or UMAP (Uniform Manifold Approximation and Projection)
    - Embeddings can be useful even without leveraging LLMs. The seminal "Word2vec" paper (Mikolov et al. 2013) introduced the idea that we could use linear algebra operations on the resulting vectors to study meaning. For example, taking the vector of "King," adding that of "Woman," and subtracting that of "Man" will yield the vector for "Queen"
    - We can also compute the cosine similarity of two vectors: computing the dot product of the normalized (i.e. unit length) vectors. This measure allows us to assess the similarity between the two underlying texts
    - Just with these operations embeddings can allow us to classify text. For example, to classify whether each individual sentence in an ECB statement assesses that inflation might be "high" or "low" in the future
    - We can also run regressions on the embeddings (including Lasso regressions, random forests or elastic nets). For example to predict amazon scores based on the online comments or to predict a stock's performance based on CFO earnings calls. These regressions will not yield conclusions about causality

- **[47:02] LLMs and GPT**
    - LLMs can be thought of as the successor to embeddings. They are like an extremely sophisticated version of the auto-predict feature of messaging apps. GPT-4 has 1.8 trillion parameters (simply put: 1.8 trillion coefficients in the PCA) and are trained on 13 trillion tokens (effectively all the public data that exists)

    - When using them to generate new data, the key problem is that the model does not know about the specific data you aim to generate. Because of this it is helpful to give it context, but it should be short because there is a limit on how many tokens you can prompt the model with, while doing so will also minimize costs
    - As a side note, if you are “nicer” to the model (e.g. saying “please”) you might get better answers, since in the training data better quality questions are tied to better quality answers
    - If you want to prompt the LLM on a very long text (like a book) before doing so you will have to use embeddings to find the relevant sections and give only these to the model (details below)
    - There is no definite theory on how to best carry out each of these steps, so it is best to test different things. Each step is also based on open areas of research. A recent paper from Netflix argued that we lose a lot of information when we normalize vectors to compute their cosine similarity (Steck 2024)

- **[58:15] Designing a good RAG is difficult**
    - To recap, using LLMs to generate data involves: (1) digitizing documents and splitting them into chunks, (2) compute the embedding for each chunk, (3) given what you want to search compute the embedding for the search query, (4) based on the two previous steps, select the chunks most similar to the query in the embedding space (this will identify the relevant chunks of the document), (5) prompt the model sending only the relevant chunks and ask it to give a numerical/coded answer
    - This is called a Retrieval Augmented Generation (RAG) process. Designing a good RAG is difficult, and one should be careful with each of the steps
    - There are many ways to digitize PDFs, and many PDF parsers will give you text that is hard to read
    - There are many different chunking strategies. How many chunks is it optimal to have? Should they be of a fixed length? Or should they be based on semantic breaks like paragraphs? It is unclear whether we will need chunking in the future, but today it outperforms digesting a full PDF and is cheaper
    - One can also choose different embedders (OpenAI offers 3). There are also different ways to search for the relevant chunks (e.g. approximate k-nearest neighbors, metadata filtering, or reranking)
    - How you ask the model a question matters a lot: optimal prompting is an entire research area
    - Never trust the output blindly. For robustness try running the RAG with different prompts and temperature parameters (which governs the model’s creativity) for a small subset of observations and see how it performs. Ultimately we still need humans for validation
    - Some say that Python is eating the world, and indeed Python + LLMs is a powerful combo. One need not be a professional coder to use LLMs to generate data

- In the future the technology will continue improving, but perhaps not as much as many believe. The most recent improvements have been in improving the training data and in reducing the costs of training a model

Timestamps:
[0:00] Introduction and poll questions
[3:55] Why should we use AI to create data?
[25:50] The basics of embeddings
[47:02] LLMs and GPT
[58:15] Designing a good RAG is difficult