GROWTH ACCELERATION PARTNERS

AVOID THESE PROBLEMS WHEN PUTTING LLMS INTO PRODUCTION

GAP Expertise Removed the Trial and Error from App Development

LARGE LANGUAGE MODELS

Large Language Model (LLM) applications right now are hotter than a Nevada summer. But they present real challenges for companies related to data privacy, security and legal/regulatory compliance, resource intensity, fine-tuning complexity, bias and ethical use.

Striking a balance between utility and data protection is crucial, while training and deploying LLMs require expert resources to achieve desired performance. And deploying LLMs at scale, across various platforms and environments, requires careful optimization and project management.

Growth Acceleration Partners has this LLM expertise, and is actively helping customers overcome these challenges with precision and speed.

It's Cool, But is it Capable?

Chip Huyen, author of the influential book Designing Machine Learning Systems, summarizes the issues companies face when building LLM applications succinctly. "It's easy to make something cool with LLMs, but very hard to make something production-ready with them," Huyen explained in an April 2023 blog.

The promise of utilizing LLMs in application development remains compelling. The explosion in use cases, interest and deployments of generative AI has LLMs at their heart. ChatGPT, from OpenAI, is the most well-known generative system. Not to be left behind, Google's Bard and Meta's LLaMa are likewise able to use deep learning techniques with massive data sets to understand, predict and "generate" new content, whether images, computer code, or a high-school book report on Ulysses.

The jaw-dropping potential for how the enterprise can leverage it, but also the variety of use cases, is key.

Why is it hard to get projects into production? Huyen points to several reasons, going from the practical elements of cost and latency, to the more theoretical "ambiguity of natural languages."

Consider prompt engineering: Originally the only way a machine learning model could be adapted was through fine tuning, in some instances featuring brute force tactics that were computationally intensive. Taking that path can result in unpleasant surprises contained in your AWS bills.

Prompt engineering is a faster, more affordable alternative to brute force. But an accidental error in prompt engineering is what Huyen calls a "silent failure." The prompt will still run, but the output may be wildly incorrect. Output validation is absolutely required before relying on a generative system.

Another issue or roadblock comes when creating LLM applications in-house. If a third party has developed the models, then you will not know how the tool was built or how to best utilize it. And it is simply prohibitive to build and develop an in-house large language model anyway. As recent developments have shown, the rate of progress in different iterations of these models — such as GPT 4 compared with 3.5 — is less about the architectural knowledge required, and more about the budget and compute time needed to feed and train the model.

GAP's Experience Pays Off

What does this look like in practice? Because creating a pipeline around the model, or placing it into an existing pipeline, is hard. Rudimentary knowledge — such as downloading a model and asking questions of it, or drawing on example code and getting a couple of trials to work — is one thing. Getting it to a state where you can release it into the wild, is quite another.

With off-the-shelf models, you can get a solution that is 90% of the way you would want. Yet it is very hard to get it through to 100%, or even 95%. Why? Because like "normal" software development, the last 10% is the most challenging and expensive part of the project.

Here's one example: a current GAP project showcases all of the required attributes: technical acumen, innovation, and understanding which artificial intelligence technology fits where. This customer projects uses large language models to automate certain aspects of pre-processing data before it can be used in a more traditional machine learning pipeline.

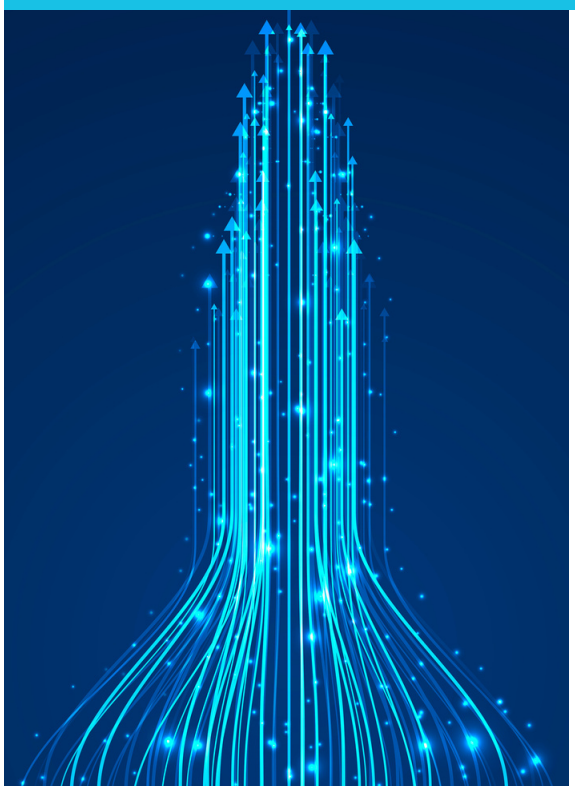
One GAP Client — an Arkansas-based database marketing company —

Had a problem related to data enhancement. They were building myriad classification models across a huge number of attributes from data such as customer surveys; and they wanted to take in information from another data provider, merge it and compare it with their universe of data. However, with data from so many sources, standardization becomes a problem. With the client's current pipeline, even though columns are semantically linked, it takes a human to manually — and inefficiently — look through it all to figure out which columns map where.

The solution was to begin experimenting with and implementing LLMs, with an already-built “understanding” of language through semantic linking, to make that mapping an automatic process. To put this project together, the code was written with PySpark, an interface for Apache Spark in Python, to enable writing Python and SQL-like commands to manipulate and analyze data. The libraries and models used were from Hugging Face, and the models were run in Databricks, which is itself linked very closely to Apache Spark. One of the key Hugging Face libraries used was Sentence Transformers, a Python framework that offers more than 500 models and can be used to embed sentences into a vector space, which can be useful for text classification, semantic similarity, and other natural language processing (NLP) tasks.

It was something of a trial-and-error process to work out which type of language model classifier worked best, as well as figuring out the correct input for the models. This is because you get a lot more than column names; you also get distinct values the columns may have, which are frequently classified into greater hierarchies. This can include demographic data and purchase data. Therefore, understanding the right data to fit into the LLM classifier was key.

Through the stack that GAP compiled, the project lead was able to in effect download a copy of the code that comprised the LLM and deploy it locally. This was another important issue for the client: to ensure their proprietary data did not end up with a third-party operator.



Relying on a trusted strategic technology solutions partner, such as GAP, to help you make the most of your LLM project, pays off. Those just beginning their LLM journey often make a couple of big mistakes:

- Some companies apply LLM solutions to problems that could be easily solved with classical machine learning.
- Others apply LLMs to problems that can be solved with straight-ahead software engineering or regular programming.

Yet the value of investing in a technology partner is not just the expertise in understanding when to use an LLM and why, but also to execute and develop a solution for clients.

EXPERIMENTATION AND THE PACE OF EVOLUTION

Perhaps the key attributes an organization needs to make LLM applications work is an appreciation of experimentation. This is common with many data science projects, where you might try different algorithms to solve classification problems, and be unsure as to which algorithm works with which parameter. Yet with minimal effort required to swap parameters and models in LLMs, the first combination is rarely the best one.

The other thing to consider — and this is the real kicker for working with a partner like GAP — is the pace of evolution with LLM applications is staggeringly fast. As Huyen observes, APIs are changing daily; new applications are discovered and new terminologies introduced. So while it may be easy to get started with large language models, the finish line might have moved without you even realizing it.

It therefore makes sense to have a trusted hand alongside to help you on your way. Wherever you are in the process, GAP can help.

09202023

To find out more, please visit WeAreGAP.com

company/growth-acceleration-partners/

@GrowthAccelerationPartners

@GAPapps