Limitless Computing

BMO Global Innovators Fund

Malcolm White, CFA
Director & Portfolio Manager
Global Equity
BMO Asset Management Inc.

Jeremy Yeung, CFA
Director & Portfolio Manager
Global Equity
BMO Asset Management Inc.

(Image: Author using Gen AI from Nightcafe Studio)

Executive Summary

AI is creating a new paradigm of how we will perform computation in the future, creating a process that we describe as "Limitless Computing". This white paper details how AI computing is different, our prediction on what new compute architectures will look like, and the ramifications for industry.

Introduction

AI represents a new way of computing. It is functionally quite different from traditional logic-based coding that requires a programmer to define a set of instructions that compute in a limited way. AI learns when shown examples of the targeted goal. This example-based programming approach gives AI the ability to tackle far more complex tasks than possible with older forms of computation. The results have been incredible with AI demonstrating the ability to mimic areas of human intelligence with unprecedented levels of accuracy.

As we look out into the next few years, these trends have major ramifications for the development of software applications. We predict a major rearchitecting of world's computational infrastructure to integrate and accommodate AI-based computing.

We have seen these dramatic transitions before with the shift from mainframe to client server compute after the broad proliferation of PCs. Recently, there was a second transition from PC-centric computing to distributed computing (e.g. cloud computing) that levered the Internet for pervasive, always-on computing for billions of concurrent users.

This white paper will discuss how AI represents the third major transition and another model for computation. Early indications from AI applications show they differ significantly from conventional ones and we will provide our thoughts on how applications will evolve with potential industry ramifications.

If even some of these predictions come true, they imply profound changes for the entire application development ecosystem and a major rearchitecting of the world's existing compute infrastructure for the better.

Traditional Computing is Highly Limited

Though hard to admit, the traditional computing model that has served us so well for decades has hit its limit. These systems can only follow instructions given by the users or programmers. They use a deterministic, predefined approach where the programmer explicitly specifies how a problem can be solved step-by-step. Programs cannot exceed the boundaries of the logic (millions of if then statements) that are hard coded into the system.

This creates numerous impediments for the robustness of these solutions. Rule-based problem solving needs all situations to be anticipated and defined in advance. They are less flexible and accommodate less variability. Programs do not evolve once written and cannot adapt to new situations unless they are programmed to do so.

This works for small, well-defined problems such as processing bill payments. But real-world problems are complicated and resist simplistic definition. Fortunately, AI programs can handle more complexity by creating algorithms that can learn and improve. This AI-based approach has successfully tackled the complexity of real-world situations such as image recognition and language comprehension, which has propelled it to the forefront of application design.

AI Programs Can Tackle Real-World Complexity

AI represents a new paradigm in computer programing. It can make educated decisions based on the data it is constantly ingesting and analyzing and thus AI is not bound by the limits set by fixed logical instructions. It excels with the following:

Advanced Problem Solving: AI introduces an entirely new way of solving complex problems that otherwise would be difficult or impossible for conventional computing. It does this by learning and improving from experience, just as humans gain knowledge and enhance their skills. While traditional programs are less flexible and only perform as instructed, AI programs have the flexibility to maneuver around unstructured environments. They can easily adapt to new unforeseen situations to tackle complex problems.

Natural Language Processing: Unlike traditional computing, AI can understand, interpret, and generate human language providing more intuitive user interfaces and transforming the way we interact with computers.

Machine Learning and Deep Learning: Conventional computing systems use static, predefined programs to perform tasks. In contrast, AI uses machine learning and deep learning algorithms to continuously learn from new data and improve performance over time.

Perception: AI has the ability to learn from its environment using computer vision, audio files for speech recognition, and billions of examples of text for language understanding and translation.

Personalization: AI can make software dynamically adaptable to individual users. It can tailor content, recommendations, and functionality to individual needs, based on learned preferences. This is done not by changing the code as required traditionally but by changing the system level instruction to the AI Chatbot. Modifying the instructions in English (or another natural language) changes the persona and the output. We have created AI programs that can perform multiple tasks (Pizza Store, Financial Planner Equity Analyst) simply by giving different instructions to the AI persona[1]. No other code needs to be modified and this can be done dynamically during the run time of the application.

Uncertainty: Traditional programs operate under conditions where everything is certain and known. AI programming in contrast can make decisions based on incomplete or uncertain information. A great example of this is called zero-shot learning[2] and occurs when the AI system is given a term it does not understand. It can quickly compare the context of this term to other similar ones and infer meaning. For example, it may come across the word “ocelot” and not know its meaning. But from a broader interpretation of the context of the text surrounding the word, it could assume it is cat-like and lives in the wild, which gives the AI enough information to continue its computation. Classic computing systems would not know what to do next.

[1] https://huggingface.co/spaces/mswhite/bot
[2] https://jalammar.github.io/illustrated-word2vec/

No Free Lunch – With AI Capabilities Come New Coding Challenges

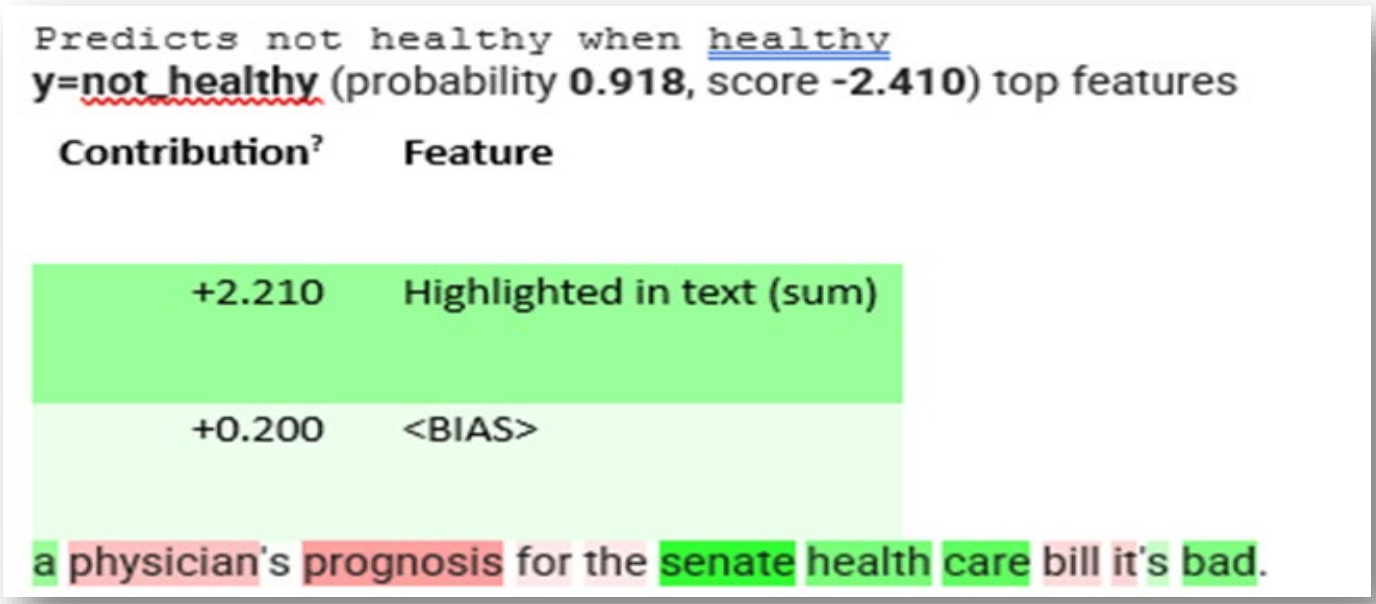
While AI applications benefit from new capabilities, such as flexibility and the ability to deal with uncertainty, these capabilities can add more challenges to coding especially when debugging applications. There is no free lunch and programmers will need to compensate for the open-ended nature of AI application development by doing more to check and validate the results.

Error checking in traditional logic-based coding is quite basic. One checks the output to see if it falls within the expected range of outputs and if not the application spits out an error e.g. "garbage in, garbage out."

Debugging is not simple in an AI environment. We encountered two problems when we developed new AI applications. The first problem is explainability – how did this incredibly complicated algorithm attain this result. The second problem is observability - this application required several steps to complete its task and the result was incorrect but where did it make a misstep in this sequence?

There are solutions for the explainability problem and AI applications can overcome their "black box" nature by 1) developing models that incorporate explainable AI (XAI) 2) supplementing AI results using visualizations 3) using techniques such as XGBoost that are based on decision trees making them easier to understand and explain feature importance and 4) using techniques such as Local Interpretable Model-agnostic Explanations (LIME)[3] or Shapley values[4] to understand the impact of the addition or subtraction of factors. It will be difficult to promote AI-based applications in regulated industries if they lack explainability so solutions are critical.

Figure 1- Visualization Explaining How a Large Language Model Scores a Sentence (Source:



[3] https://christophm.github.io/interpretable-ml-book/lime.html
[4] https://christophm.github.io/interpretable-ml-book/shapley.html

Complementary not Competitive

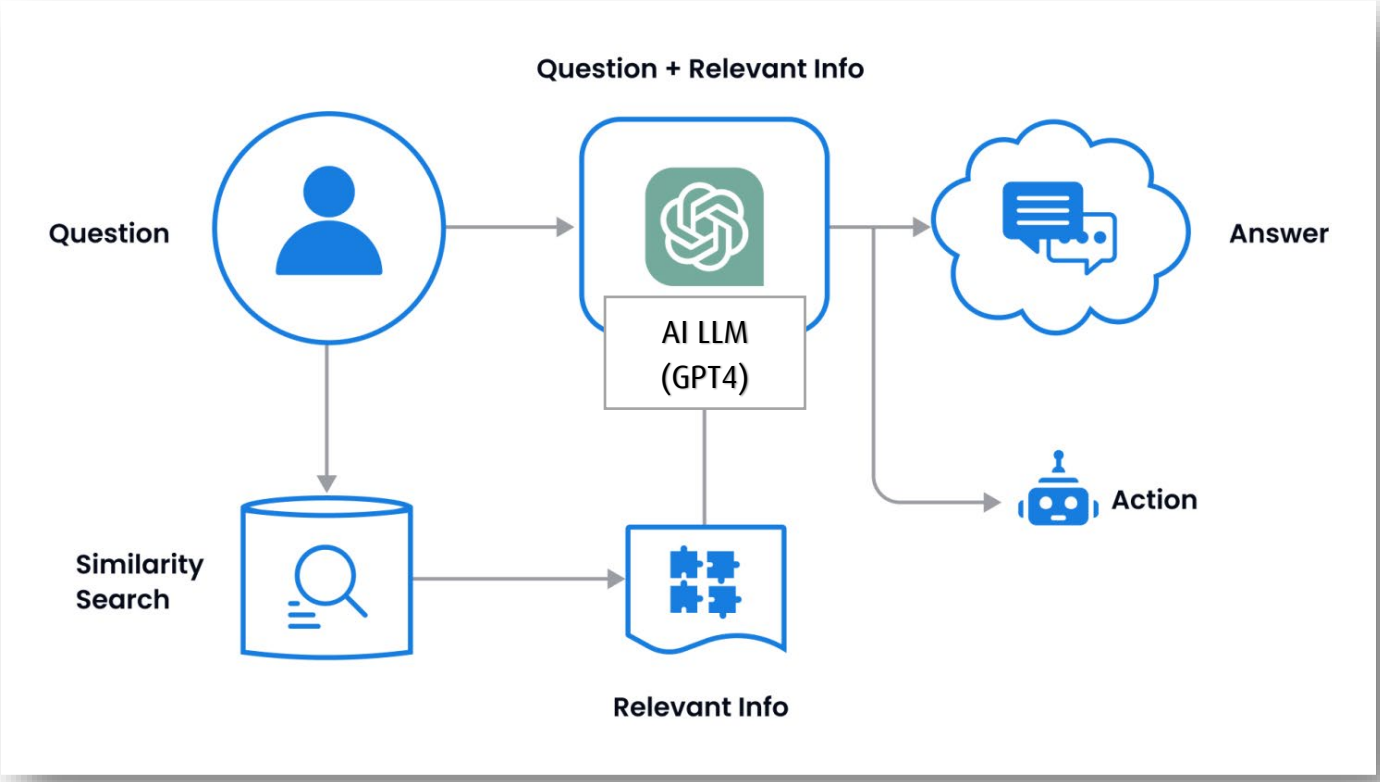
We believe that traditional computational techniques are complimentary to newer AI ones. If a compute process is well defined, it can be given to the logic engines of a program to complete. Conversely if the process is Ill-defined or complex, the logical component of the program can hand it over to the AI. There are other ways that AI applications and traditional applications complement each other.

Automating Routine Tasks: AI applications can automate and offload routine tasks within a program freeing up time for users to focus on more complex tasks.

Pattern Analysis: AI applications can analyze large volumes of both structured and unstructured data, identify patterns and trends, and make decisions with little to no human intervention.

Enhancing User Experience: AI has been cited as a better way for computers and users to interface in a manner that is more efficient than a graphical interface. This is the vision for Microsoft's Co-Pilot initiative[5]. AI applications can offer personalized interfaces for users based on their preferences, behavior, and instructions.

Figure 2 - Overview of the Flow of an AI application (Source: ChatGenAI)

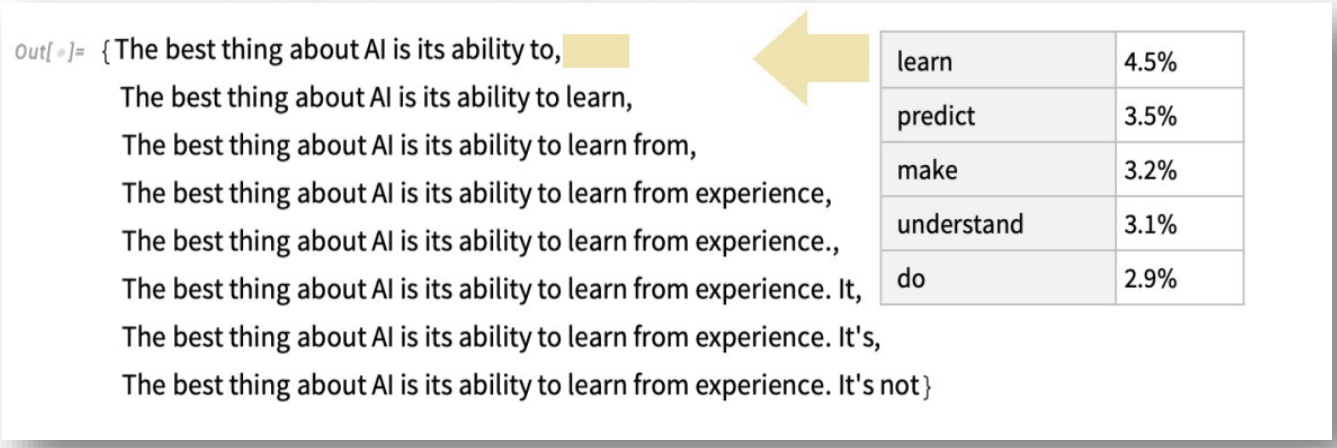


[5] https://blogs.microsoft.com/blog/2023/09/21/announcing-microsoft-copilot-your-everyday-ai-companion/

Extending Capabilities

While large language models (LLM) such as OpenAI's ChatGPT are impressive in their ability to generate human-like text, they have several limitations. Their understanding of the world is restricted to the context of the training material they have read. They don't have a true understanding beyond this and rather mimic the intelligence of a sequence they know. This is illustrated in **Figure 3** below that shows exactly how a language model works as it fills in the blanks based on the most probable word.

Figure 3 - ChatGPT is Based on the Next Most Probable Word Sequence (Source: Wolfram)



Cynics will say ChatGPT is just an advanced "autocomplete" function predicting what word comes next based on a set of inputs and probabilities. However, saying that ChatGPT is limited to this underestimates its sophistication. The capabilities of ChatGPT go beyond mere autocompletion. ChatGPT operates on a much larger scale, producing entire paragraphs and even essays. It surprisingly has the capability to write creative work - composing stories, songs, or poems. Unlike autocomplete, ChatGPT can adapt its tone and style of language to the input or instructions given and this capability is at the heart of some AI applications. When memory of a conversation is added, ChatGPT can interactively have a dialog with users and responses to prompts and even ask questions that incorporate the context of what was already said to infer meaning.

The best applications will utilize the strengths of a language model and augment areas where it is weak. For example, mathematical calculations may or may not work in a large language model depending on the complexity of the computational problem. In this case it is best to pass the question over to a program that can compute symbolically.

It is also hard to validate the results from a AI application based on LLMs as these models can make predictions or generate information that seems plausible but is actually incorrect, nonsensical. misleading, or biased. We note the infamous headline this year of the lawyers who submitted fake citations in court that were generated using ChatGPT[6].

We highlight the areas where LLMs are excellent and the areas where they may be deficient in **Figure 4 – A Comparison of the Strengths and Weaknesses of GPT (Source: Wolfram / Author)**. The best approach is to incorporate both applications to get the best result when the other algorithm cannot solve as well.

Figure 4 – The Strengths and Weaknesses of GPT (Source: Wolfram / Author)

Task to Perform	Symbolic (e.g. Wolfram)	LLM (e.g. GPT-4)
Complex Reasoning	✓	~
Data Validation	✓	X
Handle Large Datasets	✓	X
Math Calculations	✓	~
Flexible Inputs	X	✓
Human-like Interaction	X	✓
Resolve Hard Questions	X	✓
Use Unstructured Data	X	✓

[6] https://www.theguardian.com/technology/2023/jun/23/two-us-lawyers-fined-submitting-fake-court-citations-chatgpt

The Monster Renovation that Unlocks Innovation

Our first foray into LLM-based AI applications was simple "prompt engineering." These are basic applications where you give the language model an administrative prompt (instructions) and then engage in a chat session. The ChatBot will engage with the user under the guidelines you provide. For example, "You are OrderBot, and collect orders for a pizza restaurant. Greet the customer, Collect the order, and then asks if it is a pickup or delivery. Summarize the order and if it's a delivery, ask for an address. The menu is: 1) pepperoni pizza for $10 2) cheese for $8 … and toppings are: extra cheese $2, mushrooms $1.50 … " etc. With just this prompt, you can quickly create a PizzaBot persona for this task. You can also very quickly change the entire program by adjusting this prompt to say, "You are a Financial Planner / Fundamental Equity Analyst / Event Planner / Celebrity" and without any code changes, the AI will adopt the new persona and follow the instructions you gave.

The speed of application development is amazing and the best part is the flexibility of the application. For example, even though you didn't specify in the program logic that the delivery of the pizza requires a address, the AI application will infer it needs an address and will ask for one.

The limitations of prompt engineering are the size of the instructions the GPT language model can handle as they are restricted to a maximum number of words[7] per request. To go beyond this limitation, two approaches are available 1) a Retrieval Augmentation Generation such as LangChain and 2) a Vector Database. Often both approaches are used simultaneously.

A Retrieval Augmentation Generation (RAG) engine is an AI technology that utilizes existing language data to create new, meaningful sentences. The engine retrieves relevant information or data based on specific prompts or tasks and generates a coherent, contextually accurate response. The need for such a tool arises for several reasons:

Content Generation: AI applications in content marketing, professional or creative writing fields need to create high-quality text that is succinct, relevant, and engaging and more importantly referenced and sourced to check validity. A RAG in conjunction with a vector database can meet these requirements.

Sophisticated Dialog: When engaging in a dialog with users it is important to have an engine that can recall information and generate responses contributes to a more engaging, accurate and realistic conversation. RAGs can incorporate extensive memory of the conversation that can be used to enhance the dialog by recalling what was said.

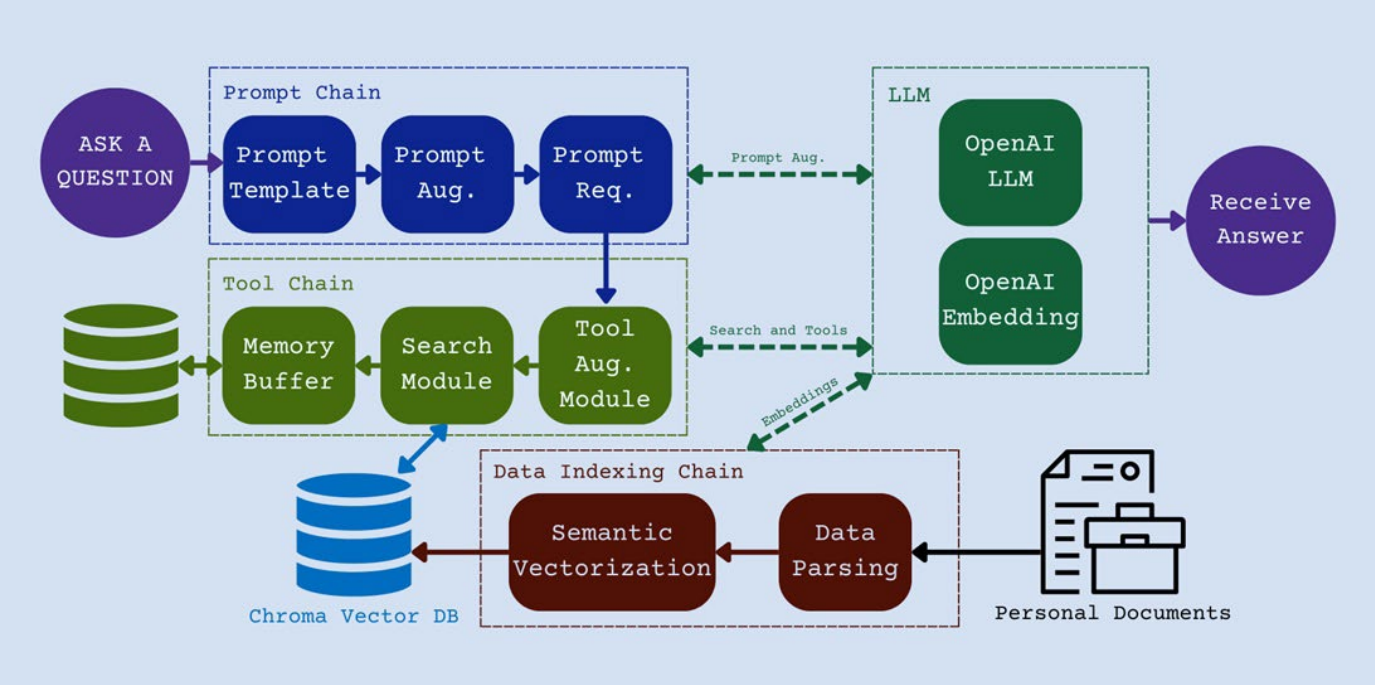
Accuracy and Efficiency: A RAG improves quality and accuracy of relevant information from multiple sources. If one source cannot answer the question, the RAG can ask for a response from another source to generate the desired output. This reduces the chance of generating irrelevant responses and improves efficiency by returning valid results on the first attempt.

[7] Technically tokens are similar to parts of words and one token is approximately four characters. Token count limits will vary by model and could be 4000 to 8000 tokens approximately per request.

A vector database is needed to handle high-dimensional data used in language models. They are needed because the language model doesn't use "words" but rather it converts a word into a multidimensional representation of a word that encodes more information (see Word2vec[8]). This vectorization of words was one of the techniques that led to a better understanding of language models ultimately to the state-of-the-art transformer models such as ChatGPT we use today. Other benefits of vector databases include 1) faster and more accurate search based on how semantically similar words are 2) scalability as these databases are designed to handle massive amounts of data scales efficiently and 3) quick recall and classification of information.

Applied together with a RAG, a vector database forms an AI application as shown in the following figure.

Figure 5- The New Architecture Required for an AI Application (Source: FourthBrain)



We have several conclusions here: 1) developers will want to create AI applications because of their flexibility and power and 2) new tools are required to build these applications properly. If enterprises want to use this innovation, they will need to "renovate" their existing infrastructure to facilitate these new AI processes that are quite different from conventional ones.

[8] https://jalammar.github.io/illustrated-word2vec/

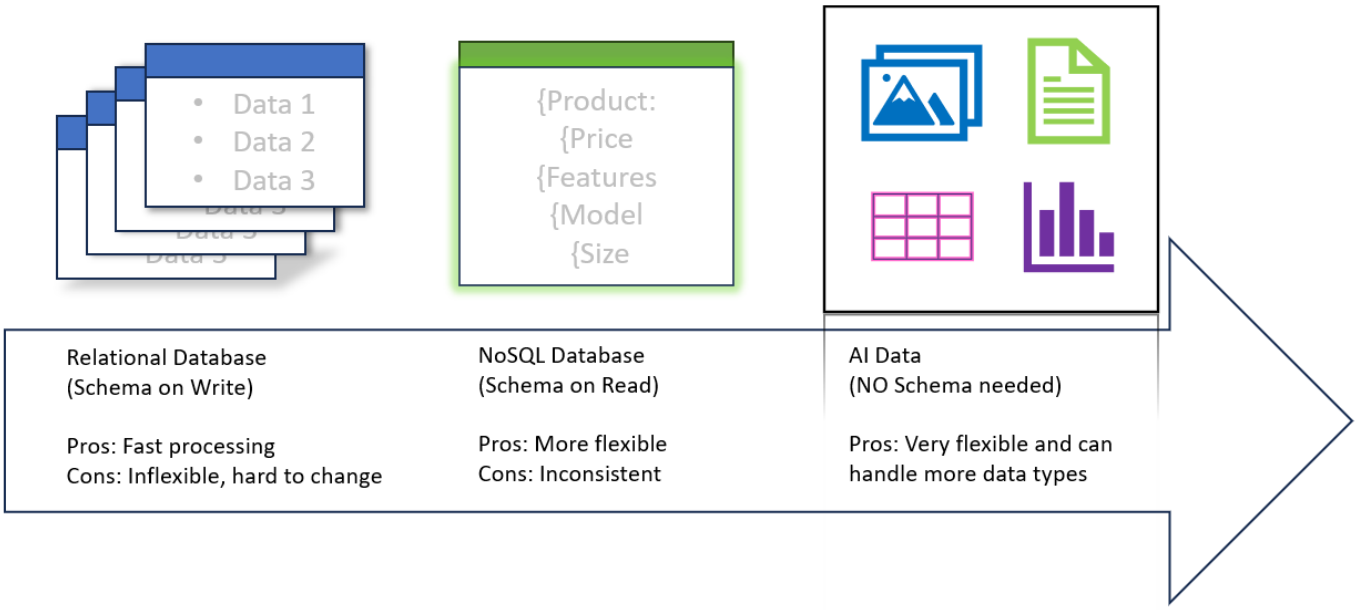
The Fast and the Flexible

We were quite surprised at how quickly we could create an AI-based application. Coding an AI application can be notably faster than programming a conventional application for several reasons. Reuse and transfer learning is one reason: AI models can often utilize transfer learning, where a pre-trained model can be tweaked to suit a new problem, saving time on development. We have used pre-trained models for both text and image applications. One application we wrote (Miami Vice) can mimic the style of an artist by simply providing examples of that artist's work. We were surprised at how well it worked after only 20 minutes of training.

But what really drove the speed of development was AI's ability to handle unstructured data. Traditional programming methods often require structured data, which needs to fit a specific schema or format. For example, you cannot create a program that gives you information about products without finding a common format for all of the data, fitting into a schema (think consistent columns), making adjustments, converting this data into a consistent format if necessary (ETL – extract, transform and load) etc. This data wrangling can consume over 70% of the time required to build an application.

In contrast, AI can handle unstructured data like text, images, or natural language, without the need to pre-format the data extensively. This ability can save a significant amount of time during development as less effort is used to pre-process or structure the data. In fact, many of our bots read data as is and can answer a variety of questions without the need to do anything to structure it[9].

Figure 6- The New Direction of Data Towards Faster Development of Applications



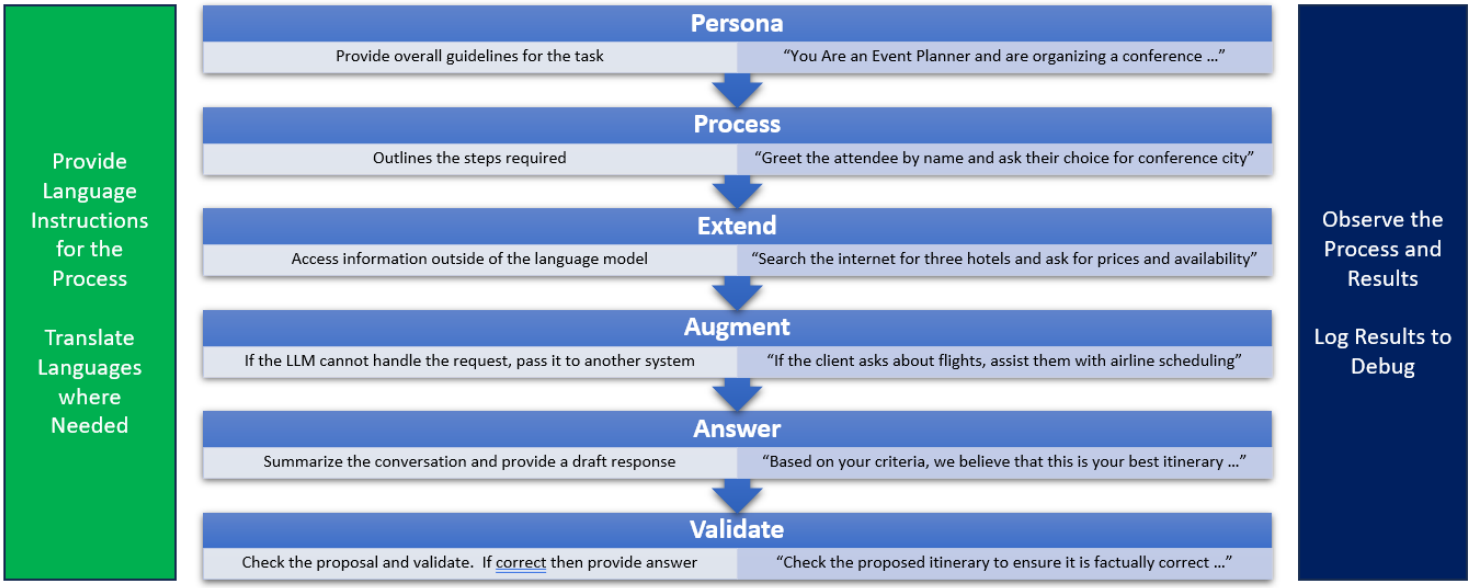
[9] You do need to encode (create word embeddings) and vectorize as mentioned in "The Monster Renovation that Unlocks Innovation" but this is handled automatically by the code

The Next Greatest Programming Language … is Language

AI applications can be designed using natural language like English or French in several ways. One such way is through Prompt Engineering where carefully crafted input prompts are used to guide a language model's response like a computer program would. Similarly, natural languages can be used to guide a Retrieval Augmentation Engine, which performs search operations based on the natural language queries provided. For instance, if an AI application is using a retrieval augmentation engine to fetch data from a large corpus or database, natural language queries can guide the data retrieval process. A typical example could involve a legal AI application that scans through vast amounts of legal documentation. A legal practitioner might use a natural language query like 'Show me all cases related to intellectual property rights in 2020'. The Retrieval Augmentation Engine can use this natural language input to filter, fetch, present, source and reference the desired data factoring in constraints that are also defined in natural language e.g. 'show Canadian cases only'. In the following schematic, **Figure 7- A Next Generation AI Language-Based Application Framework**, we show how this technique can be extended and augment the whole process. We believe that this is what future AI application architectures could look like. This has profound ramifications for developers as 1) programming in natural language is very fast and flexible 2) everyone from every business unit can now be a "programmer" 3) these systems will offer multiple ways to customize how the application works and where data is acquired.

Conclusion: You will be the programmer. You will program by "talking" to your application. You will teach it how you perform your process, and it will help you to automate your workflows.

Figure 7- A Next Generation AI Language-Based Application Framework



Industry Ramifications

Although we are still early in the process of migrating applications from traditional computing infrastructure to AI-based applications, we believe that there are significant industry ramifications.

Industry could spend the next decade doing the following:

1) Improving AI computing capabilities (2-5 years) – Examples include expanding RAG capabilities such as LangChain, integrating LLMs with symbolic or other computational engines such as Wolfram to compliment respective strengths of each platform, expand and improve foundational models like ChatGPT.

2) Creating equivalent infrastructure for AI applications (2-5 years) – Examples include adding vector databases to store language model data or modifying and converting existing relational databases and cloud storage to support AI-based applications.

3) Integrating AI with legacy applications (5-10 years) – The efficiency of AI applications will drive users to rethink existing applications and how they can incorporate AI. Commercial applications are already undergoing this and will be relaunching and upselling new AI-based integrations to their existing applications starting in 2024.

As such, this industry shift to AI-based computing should exceed our near-term projected spend of over $100 billion on AI-based semiconductors. We estimate that we will see equivalent dollar spends in multiple information technology categories such as commercial software upgrades and subscriptions, new AI-based systems, AI-hosting and training cloud compute services, application development and consultancy and system integration work.

Private Companies to Watch

Public companies with exposure to AI applications have been well publicised this year as investors seek ways to invest in AI. However, we note that there are a number of private companies that have direct exposure to AI that are generally not well known. Given that we have seen signs that capital markets are preparing for more IPOs after a long hiatus caused by rate and economic uncertainty, we highlight a sample of promising private companies to watch that we have been engaging with as we program or use AI-based large language (LLM) systems.

OpenAI Global LLC – Well known to the public as the creator of ChatGPT (GPT) and Dall-E. In addition to the publicised investment in this company by Microsoft, there have been other significant developments we believe were equally important yet missed. The first was the release of their latest language model GPT-4 to a general audience. It is fantastic and can handle important nuances that were missed in past models. The second big development was enterprise certification of their APIs. Specifically, OpenAI had been using customer requests to train models. This was problematic for enterprises wishing to maintain privacy and the confidentiality of their data. This was recently resolved and enterprises can now use a version of ChatGPT that addresses these concerns. This will ease and accelerate the adoption of their service in our opinion.

Hugging Face Inc. – This is a hosting site for applications similar to what Amazon (AWS) does for e-commerce and cloud applications. Hugging Face recognized that AI applications have different needs and custom created an environment to meet these needs for AI developers (e.g. model training, hosting, prototyping etc.)

Databricks Inc. – This company started out developing high-speed databases such as Apache Spark. Databricks was early to recognize the different data needs for AI systems and acquired MosaicML[10], which gives enterprise customers the ability to build LLMs securely and cost-effectively with their own data.

Cohere – A Canadian-headquartered, multinational organization focused on developing and advancing enterprise grade LLM systems for enterprises. We believe that there will be multiple language models used in a system.

Jasper AI – A vendor offering AI-based software that will read your marketing documents and then generate customized derivative work using the baseline content tailored specifically to different mediums such as press releases, social media, tweets, etc. without the need to rewrite for each channel. This is a great example of how LLMs can be fine-tuned and customized for enterprise needs.

LangChain – An open source framework for developing AI-based language systems. This is a must have for robust application development. Although not commercial we are monitoring the addition of their observability service that could be monetized.

Wolfram Research Inc. – Long established as a developer of symbolic computing systems, going back to their release of Wolfram Mathematica in 1988. We believe that their strengths are complementary to LLMs and needed to improve outcomes and results. The company has recently announced a number of initiatives to integrate with LLMs along these thesis.

Open-Source Vector Databases – There are a number of databases that we have used such as Pinecone, Chroma and Faiss (Meta).

[10] https://www.databricks.com/company/newsroom/press-releases/databricks-completes-acquisition-mosaicml#:~:text=Databricks%20and%20MosaicML%20have%20a,LLMs%20with%20their%20proprietary%20data.

Conclusions

We see a future paradigm shift that we call "Limitless Computing." This is the notion that AI-based computing will not be constrained by the limitations of traditional computing. AI-based computing will benefit from the flexibility of large language processing that will be better at solving real-world problems, deal with ambiguity without the need for specific logic and offer rapid development of applications using natural language.

Our thesis:

- AI-based applications are attractive as they offer the potential for unprecedented capabilities that beyond traditional programming.
- The speed of AI application development is much faster than older techniques as the new programming language is natural language.
- The ability to change the persona of the application to change its functionality will also speed up the development of AI applications. Persona modifications are quick and promote the reuse of code that is already written.
- Infrastructure will have to be duplicated to support the AI equivalent of an application. Traditional databases will need to support vectorized information or will need to be replaced. Servers will require GPUs to run efficiently.
- History will repeat itself. Just as we needed programs to monitor a process (code observability tools[11]) in the client-server world and in the cloud computing world, we will need tools to perform similar functions in an AI-based world using large language models (e.g. LangChain's LangSmith[12] observability tool for RAGs).

Investment and Industry Ramifications:

The benefits of AI in terms of increased productivity and efficiency will justify continued investment in these applications and drive spending beyond our $100 billion estimate in AI semiconductors. Successful industry players will incorporate AI features into their products to drive adoption and monetization. Those that don't will eventually risk cannibalization by other products. Many of these companies involved in this ecosystem are privately held. There is at least a decade worth of infrastructure enhancements (renovations) required to support AI, which provides an exciting backdrop for those investing in early-stage innovation.

[11] https://www.gartner.com/reviews/market/application-performance-monitoring-and-observability
[12] https://www.langchain.com/langsmith

Glossary

AI Explainability: The aspect of artificial intelligence that focuses on making the machine's decision-making process understandable and transparent to humans.

Autocomplete: A feature in computer interfaces that predicts the rest of a word a user is typing to speed up user input, minimize typo errors, and provide suggestions.

ChatBot: An application that is designed to emulate a conversation with a human.

ChatGPT: A variant of the GPT model specially trained for generating human-like text based on a given conversational prompt. It is mostly used in designing chatbots.

Deep Learning: A machine-based learning algorithm that is based on emulating the functions of a human brain in a computer. Synthetic neurons are created and connected by mathematical weights that are then trained by proving examples using data. The algorithm is optimized by minimizing the error between the predicted and expected results using a technique called gradient descent calculated using large arrays of Graphics Processing Units (GPUs).

GPT (Generative Pre-trained Transformer): A language model that uses deep learning techniques to understand and generate human-like text.

GPU (Graphics Processing Unit): A specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. It's often used in AI computation for its ability to process large block of data simultaneously.

Limitless Computing: A phrase created by the authors that captures the notion that AI-based computing will not be constrained by the limitations of traditional computing. AI-based computing will benefit from the flexibility of large language processing that will be better at solving real-world problems, deal with ambiguity without the need for specific logic and offer rapid development of applications using natural language.

LLM (Large Language Model): A type of artificial intelligence model that uses machine learning, specifically transformer neural networks, to generate lengthy and coherent pieces of text.

Prompt Engineering: The practice of crafting the input given to a language model to elicit a desired output. This includes phrasing, context setting, and other parameters that guide the model's response.

RAG (Retrieval Augmentation Engine): An engine that aids in locating and extracting the designated information from a specified dataset for fulfilling a query or a command.

Schema on Read: A flexible data management strategy where the schema for data is applied at the time of data querying or reading, allowing more agility in handling diverse data forms.

Schema on Write: A data management strategy where the schema (organization of data) is applied at the time of writing data into the database, ensuring consistency in data format at the cost of flexibility.

Semantic Similarity: A measure of the likeness of meaning between two pieces of text.

Structured Data: Data that is organized in a predefined manner or model, like rows and columns in relational databases, making it easily searchable.

Symbolic Computing: A method of computing that uses symbols rather than numerical computations. Symbolic expressions are manipulated using algebraic, trigonometric, calculus, and other mathematical rules.

Unstructured Data: Data that doesn't have a predefined model or organization, typically text-heavy and often includes metadata like dates, authors, and sources.

Vector Database: A type of database that stores, processes, and retrieves data in form of vectors (one-dimensional arrays), often utilized for high-performance computing and data analytics.

Fund Codes & Fees

Series	Fund Code/Ticker	MER (%)
Advisor FE / US$ FE	BMO99164 / BMO79164 (USD)	2.16**
T6 FE	BMO34269	2.16**
Series F / US$	BMO95164 / BMO40164 (USD)	1.05**
Series F6	BMO36164	1.05**
ETF - BMO Global Innovators Fund Active ETF Series	BGIN	1.07**

**Annual Management Expense Ratios (MERs) are estimated as the funds are less than one year old.

Disclaimer

The viewpoints expressed by the Portfolio Manager represents their assessment of the markets at the time of publication. Those views are subject to change without notice at any time without any kind of notice. The information provided herein does not constitute a solicitation of an offer to buy, or an offer to sell securities nor should the information be relied upon as investment advice. Past performance is no guarantee of future results. This communication is intended for informational purposes only.

Commissions, trailing commissions (if applicable), management fees and expenses all may be associated with mutual fund investments. Please read the fund facts or prospectus of the relevant mutual fund before investing. Mutual funds are not guaranteed, their values change frequently, and past performance may not be repeated. Distributions are not guaranteed and are subject to change and/or elimination.

For a summary of the risks of an investment in BMO Mutual Funds, please see the specific risks set out in the prospectus.

BMO Mutual Funds are managed by BMO Investments Inc., which is an investment fund manager and a separate legal entity from Bank of Montreal.

BMO Global Asset Management is a brand name under which BMO Asset Management Inc. and BMO Investments Inc. operate.