Relatório 1 – Fundamentos de Agentes de IA (I)

Marcos Vinicius dos Anjos

Descrição da atividade

1. What are AI Agents?

O vídeo começa com um tom otimista sobre o futuro dos agentes de IA. Em 2024 é relatado pela apresentadora do vídeo o crescimento em adoção e parte das possibilidades que poderão ser alcançadas com essas tecnologias. Dentre os vários termos e conhecimentos mais específicos explanados durante o vídeo, vou trazer os principais pontos na minha opnião de forma enumerada e sequência conforme aparecem no vídeo.

1.1 Monolithic Models vs Compound AI System

Os conceitos não são necessariamente antagônicos, embora, em grande medida possui abordagens contrárias entre si. No contexto do material, há uma narrativa como um *up grade* do modelo monolítico em relação ao sistema composto de IA. Tal argumentação é válida, pois há uma série de benefícios exclusivos ao sistema composto de IA podem e são de grande utilidade prática em soluções customizadas. Para ilustra melhor, trago o diagrama abaixo de forma mais ilustrativa as diferenças entre essas duas abordagens.

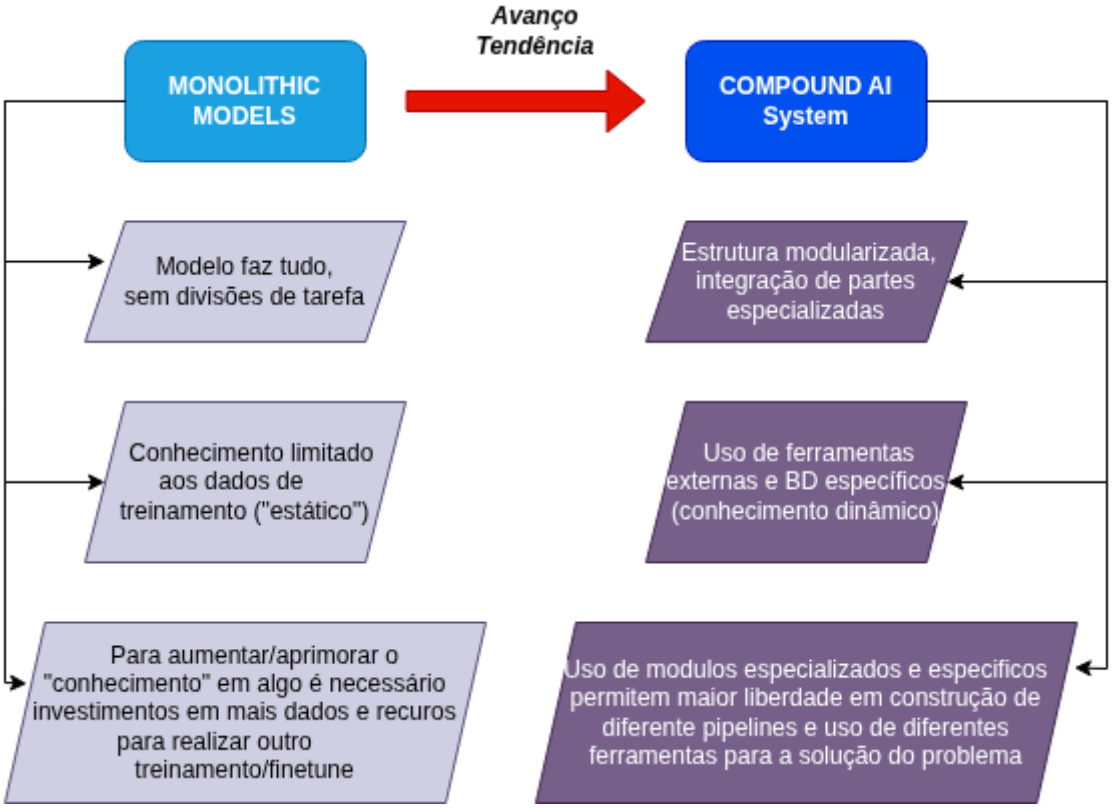

Figura 01: Diagrama Monolithic Models vs Compound AI System

Conforme ilustrado na figura acima, é nítido perceber que os modelos monolíticos possuem uma maior dificuldade de customização em relação ao sistema composto, uma vez que o conhecimento do modelo é limitado aos dados de treinamento e não fontes externas para alimentá-los de forma continuada e para mitigar esse problema seria necessário investimentos em mais dados, re-treino e/ou *finetune*, ou seja, é preciso investir tempo e recursos na maioria das vezes de forma significativa. Em outra mão temos os modelos compostos, nestes há a possibilidade de uma abordagem mais modularizada, permitindo uso de ferramentas especificas para etapas especificas que trabalham de forma integrada para resolver um problema maior.

Um cenário prático do material foi sobre quantos dias de férias a locutora tiraria para viajar, é claro que o modelo monolítico não tem acesso a essa informação pessoal, ele obtinha essa informação no seu treinamento como ele também não tem acesso a fontes externas para acessar. Portanto, ele vai retornar uma resposta errada. Quando comparado a um sistema composto, há a possibilidade de ele checar em um banco de dados do RH da empresa e obter essa informação ao invés de inferi-la.

1.2 Controle Lógico

Foi abordado dois tipos de controles lógicos, o programático que são baseados em regras humanas e o controlado por agentes de LLM (Large Language Models). Ambos possuem as suas valências, o programático tem como característica o estilo *Think fast* e por isso não há checagem ou qualquer natureza de aprimoramento de respostas, ou seja, é um fluxo linear e que não permite iterações para o mesmo *output.* Já o controle lógico realizado por agentes há a possibilidade de iterações incrementais para melhorias do *output,* e há várias metodologias para etapas especificas das iterações como *Chain of Thought* (CoT) e o *Self-Refine* todos muito bem trabalhados no artigo publicado pelo *Madaan (2023),* tanto o CoT como o *Self-Refine* tem como objetivo melhorar o resultado do LLM com instruções mais específicas, porém em momentos diferentes o CoT está associado ao começo da iteração demandando um planejamento um plano de ação das etapas para resolver o problema completo, enquanto o *Self-Refine* ligado ao aprimoramento do resultado após obter a resposta, tendo ciclos de melhorias contínuas por meio de *feedbacks* no papel de crítico (avaliador) e em seguida de refinador (quem implementa as modificações sugeridas pelo crítico), tudo isso acontece no mesmo modelo. O material se aprofunda mais no controle lógico por agente (LLM), por isso, vou focar neste, há três subdivisões interessantes que vamos abordar.

Reason: O Agente (LLM) por meio do pensamento “lento” permite definir um plano de solução de forma prévia, quero dizer, antes mesmo de atacar o problema ele já mapeia como ele vai segmentar o problema, bem como a ordem de cada etapa assim como a estratégia de CoT. Portanto, nesta etapa, o modelo não só pode, mas deve orquestrar o fluxo como as chamadas das ferramentas externas, o que pode potencializar a customização, especialização, todavia, em contrapartida pode gerar muita latência e muito tempo investido entre etapas que podem prejudicar a experiência do cliente final, e neste cenário de *tradeoff* clássico o artigo escrito por Gao (2025) sugere uma abordagem de pipeline, e dividir tarefas e paralelizar quando possível, enquanto uma LLM está baixando ou coletando uma informação, já se inicia outros processos que podem ter o seu início de forma independente na medida do possível como carregar o modelo e deixar ele pronto para receber a entrada do passo anterior. Portanto, essa etapa de controle de LLM é uma área muito explorada em estudos, pois há grande potencial de melhorias nesta parte estratégica.

Act: Após traçar todo o fluxo de trabalho, agora vamos chamar as ferramentas, o *Act* é ação que permite a customização/especialização e a modularidade do sistema composto de IA. Pois, por meio dele o sistema vai poder alimentar a LLM orquestradora com informações que estão fora do escopo do treinamento dela, tais como banco de dados específicos, dados atualizados (dados obtidos após o treinamento e que vão substituir dados defasados, aumentando a qualidade e veracidade do resultado) e APIs especificas (geolocalização, buscador web, etc...). Ou seja, mesmo que a LLM tenha conhecimento amplo, em geral, a mesma

pode não ter a mesma qualidade que modelos menores, mas que são especialista em tarefas especificas, exemplo disso foi observado por Patil (2024) onde foi criado um *framework* especialista na utilização de APIs, ele foi treinado para esse propósito de forma robusta, buscando a documentação mais atualizada além de avaliar a qualidade do documento e então sugerir mudanças nos códigos se necessários, algo muito importante principalmente se determinada ferramenta vive mudando argumentos e métodos. Neste cenário, o GORILLA a ferramenta proposta no artigo levou vantagem para essa tarefa especifica mesmo competindo com modelo muito maiores como o GPT-4. Portanto, nem sempre modelos maiores são sinônimos de melhoria, ainda mais quando a tarefa é muito específica ou em constante modificação.

Obs: O ReAct (*Reasoning + Acting*) é o framework que une os conceitos apresentados, combinando raciocínio (*Reasoning*) e ação (*Acting*) de forma iterativa. Enquanto os parágrafos anteriores descrevem o planejamento do fluxo de trabalho e o Act representa a execução prática desse planejamento através de ferramentas externas. No ciclo ReAct, a LLM orquestradora primeiro raciocina sobre qual ação tomar (ex: "preciso buscar dados atualizados"), depois executa a ação chamando a ferramenta apropriada (ex: API de busca), observa o resultado retornado e decide se precisa de mais ações ou se pode gerar a resposta final. Esse loop de "pensar → agir → observar → pensar novamente" permite que sistemas de IA compostos sejam mais robustos e precisos do que modelos monolíticos, pois podem validar informações, corrigir erros e adaptar sua estratégia dinamicamente com base nos resultados intermediários obtidos pelas ferramentas especializadas.

Acess memory: O nome é bem explicito, essa parte diz que o agente pode acessar memórias externas para validar, compor ou substituir informações do LLM. Ou seja, no cenário de uma empresa onde precisamos disparar alertas de pagamento em atrasos, as informações e a lista dos credores devem estar no seu banco de dados e não em fontes públicas para pesquisa e muito menos nos dados de treinamento do LLM (a menos que houve algum *finetune* para esse propósito específico). Portanto ao ter acesso ao banco de dados específicos, sejam locais ou em nuvens, dá ao agente grande poder de customização e criação de novos produtos que agregam grande valor. No exemplo dos devedores, pode ser criado uma automação para se obter a lista e então enviar de forma automática uma mensagem até personalizável com nome, valor devido e novo vencimento. Trabalho que se fosse manualmente feito por humano levaria minutos para cada usuário e por agentes pode ser realizado, em milissegundos, talvez em nanossegundos.

2. The Rise Of AI Agents And Agentic Reasoning

O material de apoio começa defendendo que a IA é de uso de papel geral assim como a energia elétrica e a sua aplicação podem ser quase que infinitas, e dentre as várias possibilidades os agentes de IA fazem parte de um tema muito aquecido e repleto de oportunidades. Agora vou trazer os pontos que mais me chamaram a atenção durante o vídeo.

2.1 Camadas da IA

A IA no conceito no senso comum pode ser representando em grande medida aos modelos generativos como GPT e o Gemini, contudo, apesar de não está incorreto esta abordagem está incompleta. Na verdade, para chegarmos aos grandes modelos, bem como as suas aplicações há outros níveis são fundamentais que suportam os modelos que são a ponta do *iceberg*. Neste sentido o apresentador mostra as camadas necessárias para que se disponibilize um grande modelo em produção.

A base de tudo são os chips e semicondutores, a tecnologia de *hardware* que avança de forma acelerada, permitiu um maior poder de processamentos, tanto por GPUs, CPUs mais modernas e os elementos que os compões nano chips, barramentos, etc...

O segundo pilar são as estruturas de *cloud infrastructure,* essa tecnologia permite um crescimento em escala e dinâmico, serviços integrados como GCP, Azure e outras grandes, fornecem a “oficina” com ferramentas para que os modelos possam trabalhar.

Por fim, temos os modelos generativos, como GPT, Gemini, etc. Porém, apesar de serem ferramentas fantásticas segundo Andrew elas não são o fim, mas marcam o começo de vários novos produtos, pois, o maior valor e que de certa forma paga todas as camadas anteriores são as aplicações, ou seja, o produto final, o *software, o programa,* a plataforma. Portanto, oq resolve um problema para o usuário/cliente.

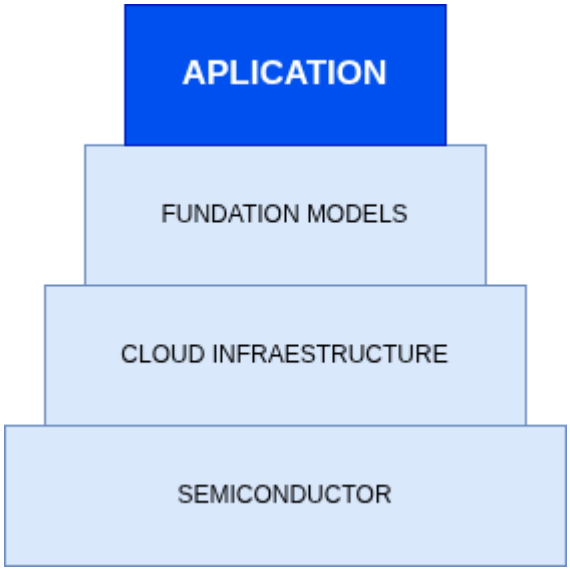

Figura 02: Camadas da IA.

Andrew fecha esse bloco reforçando que a velocidade em se gerar novos produtos, protótipos está cada vez mais eficiente, fazendo mais em menos tempo. Ele cita um exemplo que as vezes é viável realizar 20 protótipos/MVPs e que apenas 2 são validados do que ter que esperar e investir muito tempo em único produto, antes se demandava meses para trabalhos com IA supervisionadas e que hoje muitas delas podem ser implementados em semanas.

2.2 Agentic Reasoning Desing Patterns

Com tantas possibilidades de implementação e liberdade criativa que a área de desenvolvimento permite, é importante estabelecer padrões que sugerem boas práticas para a comunidade e nesse sentido a fonte traz um padrão no desenvolvimento de agentes que é divido em 4 partes e que vamos trabalhar individualmente.

Reflection: É onde tudo começa, é definido o propósito da tarefa, exemplo, automatizar uma planilha que busca informações do sistema. É o momento que há a contextualização do problema e o que se espera de resultado.

TOOL use: Assim como na sessão Act da atividade anterior, essa parte está relacionada a utilização de ferramentas externas. Portanto, as caracteríscicas como modularidade e utilização de modelos especializados vale aqui também. Ou seja, o principal benefício além de partes separadas que podem ser substiuídas, aprimoradas ou incrementadas por conta da modularização, podemos tbm ter modelos treinados para determinadas tarefas, ou APIs que buscam dados dados para fora do escopo da LLM, no qual o propósito principal é gerar valor, resolver um problema complexo dividindo-o em problemas menores e essas ferramentas contribuem de forma organizada e sistemática para solução do problema maior.

Planning/Reasoning: Assim como já falamos do CoT, esta etapa está associada ao planejamento prévio do modelo antes de executar todo o fluxo de trabalho. Nesse sentido, é importante que o modelo não só consiga desenhar o fluxograma, bem como chamar as ferramentas necessárias de forma ordenada e correta. Como característica de sistemas compostos, aqui já deve ser mapeado as iterações e até mesmo as condições de paradas. Pois segundo o artigo “SELF-REFINE:Iterative Refinement with Self-Feedback" é dito que iterações desnecessárias podem depreciar o resultado, termo chamado de over-refining, por isso é importante definir as condições de parada, podem ser iterações fixas, como 4 ciclos, ou há uma marcação no modelo crítico que vai sinalizar que a resposta já está boa ao refinador e então encerrar o ciclo.

Colaboração multiagente: É onde ocorre a integração, nela é definido os papéis internos que o LLM poderá estabelecer, metologias como o self-refine sugere que se tenha o crítico (“quem” vai procurar melhorias nas respostas que ele obteve como entrada), o refinador (“quem” vai processar a reposta original e junto ao feedback do crítico, então escrever a nova resposta). No artigo de self-refine (2023) aponta que o crítico é que desempenha o papel principal nesta metologia, pois encontrar o erro e como resolvê-lo é a principal matéria-prima para aprimorar o resultado. Nesta etapa também é definido as chamadas e integrações com as ferramentas externas.

Dificuldades

1. **Material em inglês:** Assisti duas vezes pelo menos, uma em inglês e outra com legendas em português e mesmo com a legenda traduzida, quando tinha dúvidas pesquisava o termo específico.

2. **Termos técnicos:** Não conhecia muitos termos, portanto, tive que parar muitas fezes para pesquisar. Procurava assistir a primeira vez sem muitas pausas, já na segunda vez quando estava em português só avançava quando entendia minimamente oq o apresentador estava propondo.

Conclusões

Com base nos materiais fontes obrigatórias, é fácil afirmar que a IA é uma ferramenta de propósito geral e que pode ser implementada e integrada em uma enorme gama de funcionalidades. Os agentes de IA possuem protagonismo dentre as metodologias e ferramentas de IA, pois, ao contrário de um modelo isolado como, por exemplo, o GPT ou Gemini disponível no app do celular, esses agentes podem ser especializados sob condições específicas gerando muito valor ao usuário final. Portanto, se grandes modelos como os já citados (GPT, Gemini e outros) são capazes de fazer coisas incríveis sendo eles próprios o fim em si mesmos, imagine quando eles são apenas parte de um sistema ainda maior. Nesse cenário, é fácil perceber que a solução será algo ainda mais incrível ao somar forças com outras ferramentas especializadas para um único propósito em comum. Por exemplo, um agente que combina uma LLM para compreensão de linguagem natural, uma API de busca web para dados atualizados, um modelo especializado para análise de dados e ferramentas de automação para executar ações práticas, pode criar relatórios analíticos completos de forma autônoma, algo impossível para um modelo isolado.

Referencias

IBM TECHNOLOGY. **What are AI Agents?**. 15 jul. 2024. 1 vídeo (12 min). Publicado pelo canal IBM Technology. Disponível em: https://www.youtube.com/watch?v=F8NKVhkZZWI. Acesso em: 05 fev. 2026.

SNOWFLAKE INC. **Andrew Ng Explores The Rise Of AI Agents And Agentic Reasoning | BUILD 2024 Keynote**. 19 nov. 2024. 1 vídeo (26 min). Publicado pelo canal Snowflake Inc. Disponível em: https://www.youtube.com/watch?v=KrRD7r7y7NY. Acesso em: 05 fev. 2026.

MADAAN, Aman; TANDON, Niket; GUPTA, Prakhar; HALLINAN, Skyler; GAO, Luyu; WIEGREFFE, Sarah; ALON, Uri; DZIRI, Nouha; PRABHUMOYE, Shrimai; YANG, Yiming; GUPTA, Shashank; MAJUMDER, Bodhisattwa Prasad; HERMANN, Katherine; WELLECK, Sean; YAZDANBAKHSH, Amir; CLARK, Peter. **Self-Refine: Iterative Refinement with Self-Feedback**. *Advances in Neural Information Processing Systems (NeurIPS)*, v. 36, 2023. Disponível em: https://arxiv.org/abs/2303.17651.

GOU, Zhibin; SHAO, Zhihong; GONG, Yeyun; SHEN, Yelong; YANG, Yujiu; DUAN, Nan; CHEN, Weizhu. **CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing**. In: *International Conference on Learning Representations (ICLR)*, 2024. Disponível em: https://arxiv.org/abs/2305.11738.

PATIL, Shishir G.; ZHANG, Tianjun; WANG, Xin; GONZALEZ, Joseph E. **Gorilla: Large Language Model Connected with Massive APIs**. *arXiv preprint arXiv:2305.15334*, 2023. Disponível em: https://arxiv.org/abs/2305.15334.

GAO, Silin; DWIVEDI-YU, Jane; YU, Ping; TAN, Xiaoqing Ellen; PASUNURU, Ramakanth; GOLOVNEVA, Olga; SINHA, Koustuv; CELIKYILMAZ, Asli; BOSSELUT, Antoine; WANG, Tianlu. **Efficient Tool Use with Chain-of-Abstraction Reasoning**. *arXiv preprint arXiv:2401.17464*, 2024. Disponível em: https://arxiv.org/abs/2401.17464.