How Large Language Models Can Improve Your Search Project

Search Solutions 2023

Alessandro Benedetti, Director @ Sease

21/11/2023
Headquarter in London/distributed
Open-source Enthusiasts
Apache Lucene/Solr experts
Elasticsearch/OpenSearch experts
Community Contributors
Active Researchers
Hot Trends: Neural Search,
Natural Language Processing
Learning To Rank,
Document Similarity,
Search Quality Evaluation,
Relevance Tuning

www.sease.io
WHO AM I?

ALESSANDRO BENEDETTI

- Born in Tarquinia (ancient Etruscan city in Italy)
- R&D Software Engineer
- Director
- Master degree in Computer Science
- PC member for ECIR, SIGIR and Desires
- Apache Lucene/Solr PMC member/committer
- Elasticsearch/OpenSearch expert
- Semantic search, NLP, Machine Learning technologies passionate
- Beach Volleyball player and Snowboarder
AGENDA 1/2

Introduction to Large Language Models (LLM)

The Open Source landscape (and repositories) for LLMs

Selecting the best LLM for your use case

Open Source frameworks and projects to interact with LLMs

Ways of adding LLMs to Search
AGENDA 2/2

Popular Open Source search engines and LLMs

Rabbit Holes

Future Works
AGENDA 1/2

- Introduction to Large Language Models (LLM)
- The Open Source landscape (and repositories) for LLMs
- Selecting the best LLM for your use case
- Open Source frameworks and projects to interact with LLMs
- Ways of adding LLMs to Search
AI, Machine learning and Deep Learning

ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour

MACHINE LEARNING
Subset of AI technique which use statistical methods to enable machines to improve with experience

DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible

GENERATIVE AI

- Text
- Code
- Images
- Video
- Music
- ...

ease
WHAT IS A LARGE LANGUAGE MODEL?

- Transformers
- **Next-token-prediction** and **masked-language-modeling**
- estimate the likelihood of each possible word (in its vocabulary) given the previous sequence
- learn the statistical structure of language
- pre-trained on huge quantities of text

https://towardsdatascience.com/how-chatgpt-works-the-models-behind-the-bot-1ce5fca96286
FINE-TUNED FOR...

- Following Instructions
- Sentence similarity
- Summarizing text
- Creating content
- Translating content
- Classifying/categorizing content
- Rewriting content
- Annotating images
- Synthesizing text to speech
- Correcting spelling
- Detecting fraud
- Generating code
- Doing sentiment analysis
- ...

Ease
Instruct Model: what is it?

- **Generative Pre-training Transformer**
- **Product** capable of generating text in a wide range of styles and for different purposes responding to a prompt
- (based on) generative AI Large Language Models
- e.g. **InstructGPT, GPT 4**...

most of our explanations come from here
Deep Reinforcement Learning

- Input status -> vector
- **Policy network**: A probability for the actions is estimated by a policy (neural network)
- An action is **sampled** from the probability distribution
- the action is performed on the **real** system
- the **reward** is observed
- **Policy Gradients**: the reward is back-propagated to the policy (to affect next probability estimations)

[Link to source](http://karpathy.github.io/2016/05/31/rl/)
Reinforcement Learning from Human Feedback

1. **Supervised fine-tuning step**
   
a pre-trained language model is fine-tuned on a relatively small human-curated dataset, to learn a supervised policy (the SFT model) that generates text from a prompt.

2. **Reward estimation step**
   
a pre-trained language model is fine-tuned on a relatively large human-curated dataset, to learn a reward function that generates a rating from a prompt and a response.

3. **Proximal Policy Optimization (PPO) step**: the reward model is used to fine-tune the SFT model. The outcome of this step is the final model (that can be iteratively improved).

- 2-3 are iteratively repeated
Supervised Fine-Tuning (SFT) Model

- training sample <prompt, text> -> human-curated
  - directly from Human labellers
  - from GPT3 clients
  - 10-15,000 ‘ish samples

- starting from GPT-3.5 series.
  - Presumably the baseline model used is the latest one text-davinci-003, a GPT-3 model which was fine-tuned mostly on programming code.

- expensive -> scale this up is not a solution to improve the model
**Reward model**

- **Scope**: fine-tune a model that estimates a score for \(<\text{prompt, text}>\) pair

- A list of **prompts** is selected and the SFT model generates **multiple outputs** (4…9) for each prompt.

- **Training Set**: Humans rank the outputs. The size of this dataset is approximately 10 times bigger than the dataset used for the SFT model.

- The fine-tuned model takes as input a few of the SFT model outputs and ranks them in order of preference. (Learning to Rank, sounds familiar?)

- **easier** for humans to **rate**, rather than write text

- the reward function can be further updated with users’ feedback
Fine-tuning the SFT model via Proximal Policy Optimization (PPO)

- PPO is a **reinforcement learning** algorithm.
- "on-policy"
  PPO is continuously adapting the current policy according to the **actions** that the agent is taking(sampling) and the **rewards** it is receiving.
- PPO uses a **trust region optimization method** - it constrains the change in the policy to be within a certain distance of the previous policy in order to ensure **stability**
Fine-tuning the SFT model via Proximal Policy Optimization (PPO)

- **PPO policy** is initialized from the SFT model
- **value function** is initialized from the reward model.
- The environment presents a **random prompt** and expects a **response**
- Given the prompt and response, it produces a **reward**
- **policy** get updated and the episode ends.
- During the fine-tuning many episodes happen
Problems...

- Biases in generated text
- Incredibly expensive to pre-train (GPUs)
- High operating costs
- Low explainability
- Difficulty troubleshoot due to complexity
- Vulnerability to malicious prompts that could break the system
AGENDA 1/2

- Introduction to Large Language Models (LLM)
- The Open Source landscape (and repositories) for LLMs
- Selecting the best LLM for your use case
- Open Source frameworks and projects to interact with LLMs
- Ways of adding LLMs to Search
• Architecture
• Datasets
  ○ pre-training
  ○ fine-tuning
• Code
  ○ pre-training
  ○ fine-tuning
  ○ inference
  ○ …
**OPEN SOURCE?**


**Opening up ChatGPT: Tracking openness, transparency, and accountability in instruction-tuned text generators**  
Authors: Andreas Liesenfeld, Alianda Lopez, Mark Dingemanse  
CUI '23: Proceedings of the 5th International Conference on Conversational User Interfaces  
July 2023

<table>
<thead>
<tr>
<th>Project</th>
<th>Availability</th>
<th>LLM data</th>
<th>LLM weights</th>
<th>RLHF data</th>
<th>RLHF weights</th>
<th>License</th>
<th>Code</th>
<th>Architecture</th>
<th>Preprint</th>
<th>Paper</th>
<th>Data sheet</th>
<th>Package</th>
<th>Access methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>chatGPT</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td></td>
<td>x</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>❌</td>
<td>❌</td>
<td>X</td>
<td>❌</td>
<td><a href="https://chat.openai.com">https://chat.openai.com</a></td>
</tr>
<tr>
<td>StableVicuna-13B</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>❌</td>
<td>❌</td>
<td>X</td>
<td>✓</td>
<td><a href="https://huggingface.co/StableVicuna-13B">https://huggingface.co/StableVicuna-13B</a></td>
</tr>
<tr>
<td>CarperAI</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>❌</td>
<td>❌</td>
<td>X</td>
<td>✓</td>
<td><a href="https://github.com/Akegarasu/CarperAI">https://github.com/Akegarasu/CarperAI</a></td>
</tr>
<tr>
<td>text-generation-webui</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>❌</td>
<td><a href="https://github.com/Akegarasu/text-generation-webui">https://github.com/Akegarasu/text-generation-webui</a></td>
</tr>
<tr>
<td>oobabooga</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>❌</td>
<td>❌</td>
<td>X</td>
<td>✓</td>
<td><a href="https://github.com/mosaicml/llm-foundry/mpt-7b-instruct">https://github.com/mosaicml/llm-foundry/mpt-7b-instruct</a></td>
</tr>
<tr>
<td>Falcon-40B-Instruct</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>❌</td>
<td><a href="https://huggingface.co/Falcon-40B-instruct">https://huggingface.co/Falcon-40B-instruct</a></td>
</tr>
<tr>
<td>TII</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/tiiuae/Falcon-40B-instruct">https://github.com/tiiuae/Falcon-40B-instruct</a></td>
</tr>
<tr>
<td>minChatGPT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/minChatGPT">https://github.com/minChatGPT</a></td>
</tr>
<tr>
<td>ethanyanjiali</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/ethanyanjiali/minChatGPT">https://github.com/ethanyanjiali/minChatGPT</a></td>
</tr>
<tr>
<td>trlx</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/trlx">https://github.com/trlx</a></td>
</tr>
<tr>
<td>carperai</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/carperai/trlx">https://github.com/carperai/trlx</a></td>
</tr>
<tr>
<td>stanford_alpaca</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/stanford_alpaca">https://github.com/stanford_alpaca</a></td>
</tr>
<tr>
<td>Tatsu labs</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/Tatsu-labs">https://github.com/Tatsu-labs</a></td>
</tr>
<tr>
<td>Cerebras-GPT-111M</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>❌</td>
<td>✓</td>
<td>✓</td>
<td><a href="https://github.com/Cerebras-GPT-111M">https://github.com/Cerebras-GPT-111M</a></td>
</tr>
</tbody>
</table>
False claims?

- **Llama**
  - Fully Open Source?
  - doesn’t actually use an OSI approved license [Check the license]
  - OSI-approved licenses may fall short of certain needs of AI models
  - not sharing training data
  - not sharing training code
- Not only LLAMA

https://spectrum.ieee.org/open-source-llm-not-open

https://www.theverge.com/2023/10/30/23935587/meta-generative-ai-models-open-source
Looking to fine-tune your open source LLM? Try LoRA.

Microsoft offers the open sourced LoRA (Low-Rank Adaptation of Large Language Models) project on GitHub:

- LoRA is a training method that uses a mathematical trick to decompose large metrics into smaller ones. This leads to fewer parameters and more storage efficiency, resulting in quicker processing time.
- Techniques like LoRA can help you deploy LLMs to many customers, since it only requires saving small matrices.
- Other techniques for fine-tuning LLMs include hard tuning, soft tuning, and prefix tuning.

OPEN SOURCE Large Language Models


- Generalists
  - MPT-Series
  - Falcon-Series
  - Bloom
  - T5
  - Mistral
  - LLaMA 2
    - alpaca
    - vicuna
  … many others!
**OPEN SOURCE** Large Language Models

- **Fundamental Biomedicine Science**
  - MoLFormer - Chemical molecules
  - Nucleotide Transformer - DNA sequences
  - Evolutionary Scale Modeling - Proteins
- **Biomedical - Clinical Healthcare Support**
  - BioGPT
  - BioMedLM (previously known as PubMedGPT)
  - GatorTron
- **Finance**
  - FinBERT
- **Legal**
  - LEGAL-BERT
A list of Awesome LLM's to select the best for your business.

What best represents your Search use case? Product Search or News Search or?

Select top 3 LLMs ask:
- Use LLM out-of-box?
- LLM requires Fine-Tuning?
- Feature Engineering?

#1. Awesome-LLM - general list with corresponding papers, tutorials
   Pre-training / Instruction Tuning / Alignment
   https://github.com/Hannibal046/Awesome-LLM

#2. Awesome-LLM-Large-Language-Models-Notes
   https://github.com/kyaiooiayk/Awesome-LLM-Large-Language-Models-Notes

#3. awesome-decentralized-llm (effectively "open source" with "commercial license")?
   Collection of LLM resources that can be used to build products you can "own" or to perform reproducible research.
   https://github.com/imaurer/awesome-decentralized-llm

HuggingFace's Open LLM Leaderboard
https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard
AGENDA 1/2

- Introduction to Large Language Models (LLM)
- The Open Source landscape (and repositories) for LLMs
- Selecting the best LLM for your use case
- Open Source frameworks and projects to interact with LLMs
- Ways of adding LLMs to Search
Selecting the best LLM

What is the best LLM to use with your Business Content?

1. A list of Awesome LLM’s to select the best for your business.

2. What best represents your Search use case? Product Search or News Search or?

3. Select top 3 LLMs ask:
   - Use LLM out-of-box?
   - LLM requires Fine-Tuning?
   - Feature Engineering?
What can I use?

What best represents your Business Search Use Case?
Product Search? Or News Search? or Research Paper Search?

1. In regards to an Open LLM, can I legally use it for my commercial business?
   Real world Open LLM examples: Dolly, DALL-E, Falcon.

2. Using a list of Open LLM's that I can legally use, was it pre-trained on similar content?
   Real world Content examples: Product Search (Kaggle Walmart?), Legal (LexisNexis) or ?

3. Using a list of Open LLM's that are highly relevant to my business content and use case
   Do I have a version fine-tuned for my task (sentence similarity, instruct, summarization, etc?)

4. Given these questions as a starting point and list of Awesome Github Resources, ask you team:
   Can we use the LLM out-of-box? Will the LLM requires Fine-Tuning? Any additional Feature Engineering?
Starting point

https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1

1. In regards to an Open LLM, can I legally use it for my commercial business?
   Apache 2.0.

2. Using a list of Open LLM's that I can legally use, was it pre-trained on similar content?
   paper -> variety of publicly available conversation datasets, on instruction datasets publicly available on the Hugging Face repository. No proprietary data or training tricks were utilized.

3. Using a list of Open LLM's that are highly relevant to my business content and use case
   instruction based
Starting point
https://huggingface.co/tiiuae/falcon-180B

1. In regards to an Open LLM, can I legally use it for my commercial business?
   https://huggingface.co/spaces/tiiuae/falcon-180b-license/blob/main/LICENSE.txt

2. Using a list of Open LLM's that I can legally use, was it pre-trained on similar content?
   Falcon-180B was trained on 3,500B tokens of RefinedWeb, a high-quality filtered and deduplicated web dataset which we enhanced with curated corpora. Significant components from our curated corpora were inspired by The Pile (Gao et al., 2020).

3. Using a list of Open LLM's that are highly relevant to my business content and use case
   instruction based fine tuned available
Starting point
https://huggingface.co/tiiuae/falcon-7b

1. In regards to an Open LLM, can I legally use it for my commercial business?
   Apache 2.0

2. Using a list of Open LLM's that I can legally use, was it pre-trained on similar content?
   Falcon-7B was trained on 1,500B tokens of RefinedWeb, a high-quality filtered and deduplicated web dataset which we enhanced with curated corpora. Significant components from our curated corpora were inspired by The Pile (Gao et al., 2020).

3. Using a list of Open LLM's that are highly relevant to my business content and use case
   instruction based fine tuned available
Compare Large Language Models
https://chat.lmsys.org/?arena

- Select Model A/ Model B
- Run Prompt in parallel
- give your vote
AGENDA 1/2

- Introduction to Large Language Models (LLM)
- The Open Source landscape (and repositories) for LLMs
- Selecting the best LLM for your use case
- Open Source frameworks and projects to interact with LLMs
- Ways of adding LLMs to Search
Interact with a Large Language Model
To encode From text to vectors (or in general to enrich your data):

- On premise
  - own both the services and servers
- In the cloud
  - just use your favourite cloud provider and host your own services
- As a service
  - https://huggingface.co/inference-endpoints
  - https://aws.amazon.com/sagemaker/
  - ...
**Interact with a Large Language Model**
LangChain is a framework for developing applications powered by language models

[https://www.langchain.com/](https://www.langchain.com/)

This framework consists of several parts.

- **[https://github.com/langchain-ai/langchain](https://github.com/langchain-ai/langchain) - MIT license**
- **LangChain Libraries**: The Python and JavaScript libraries. Contains interfaces and integrations for a myriad of components, a basic run time for combining these components into chains and agents, and off-the-shelf implementations of chains and agents.
- **LangChain Templates**: A collection of easily deployable reference architectures for a wide variety of tasks.
- **LangServe**: A library for deploying LangChain chains as a REST API.
- **LangSmith**: A developer platform that lets you debug, test, evaluate, and monitor chains built on any LLM framework and seamlessly integrates with LangChain.
**Build event-based applications**
Use the power of streaming to connect to LLMs

[https://langstream.ai/](https://langstream.ai/)

- [https://github.com/LangStream/langstream](https://github.com/LangStream/langstream) - Apache 2.0 license
- **Easy Development to Production** - Develop and test robust Gen AI apps right on your laptop. When you're ready, deploy to a production environment powered by Kubernetes and Kafka—all through a single CLI command.
- **Event-Driven Architecture** - LangStream is engineered event-driven computing by a team with decades of experience. Build reactive, scalable, and fault-tolerant Gen AI applications with ease.
- **Configurable Agents** - Out of the box, LangStream offers configuration-driven agents designed for various tasks—be it AI chat completions, text processing, or working with vector databases. If they don't meet your unique needs, craft your own agents in Python.
- **Up-to-Date Libraries** - The LangStream runtime is always stocked with the latest Gen AI libraries like LangChain and LlamaIndex, so you're never left behind.
Interact with a Large Language Model

Haystack is the open source Python framework by deepset for building custom apps with large language models

https://haystack.deepset.ai/

- [https://github.com/deepset-ai/haystack](https://github.com/deepset-ai/haystack) - Apache 2.0 license
- Use the latest LLMs: hosted models by OpenAI or Cohere, open-source LLMs, or other pre-trained models
- All tooling in one place: preprocessing, pipelines, agents & tools, prompts, evaluation and finetuning
- Choose your favorite database: Elasticsearch, OpenSearch, Weaviate, Pinecone, Qdrant, Milvus and more
- Scale to millions of documents: use Haystack's proven retrieval architecture
HAYSTACK

- Effortless deployment of models from Hugging Face or other providers into your NLP pipeline
- Create dynamic templates for LLM prompting
- Cleaning and preprocessing functions for various formats and sources
- Seamless integrations with your preferred document store (including many popular vector databases like Faiss, Pinecone, Qdrant, or Weaviate): keep your NLP-driven apps up-to-date with Haystack’s indexing pipelines that help you prepare and maintain your data
- The free annotation tool for a faster and more structured annotation process
- Tooling for fine-tuning a pre-trained language model
- Specialized evaluation pipelines that use different metrics to evaluate the entire system or its individual components
- Haystack’s REST API to deploy your final system so that you can query it with a user-facing interface
We preferred **Haystack** to develop our neural highlighter plugin because it proved to be quite intuitive for supporting bert-like models. It was a quick investigation but its documentation was straight away useful to build a quick prototype for our use case. The perception on **LangChain** was it was more oriented to remote inference services (such as OpenAI ones).
**LangChain**, while feature-rich, presents a **steeper learning curve** compared to the more straightforward **Haystack**. While LangChain is being harnessed for comprehensive enterprise chat applications, Haystack is often the choice for lighter tasks or swift prototypes.

https://www.stork.ai/blog/langchain-vs-haystack-a-comparative-insight
Both **LangChain** and **Haystack** are LLM orchestration frameworks, so the mission is pretty much the same.

**LangChain** offers many integrations with LLM providers and vector DBs, it has a huge community and popularity, it’s good for quick prototypes but difficult to customise, because of the many layers of abstraction, releases happen frequently (with many breaking changes).

**Haystack** has less integration but an active community.

The focus is on modularity and easily customisable pipelines.

Very stable, production ready (used by Nvidia, Netflix…).

The project was born for RAG, so there’s good support for search engines and rankers.
<table>
<thead>
<tr>
<th>Aspect</th>
<th>LangChain</th>
<th>Haystack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Website</td>
<td>LangChain</td>
<td>Haystack</td>
</tr>
<tr>
<td>Cost &amp; Model</td>
<td>Open source with value-added services for enterprises</td>
<td>Open source supporting deepset's products</td>
</tr>
<tr>
<td>Funding</td>
<td>$10 MM</td>
<td>$45.2 MM (deepset)</td>
</tr>
<tr>
<td>Integrations &amp; Tools</td>
<td>Abundant, e.g., AWS Lambda, APIFY</td>
<td>Limited but customizable</td>
</tr>
<tr>
<td>Community Support</td>
<td>Very Good</td>
<td>Adequate</td>
</tr>
<tr>
<td>Complexity</td>
<td>High, with extensive object-oriented concepts</td>
<td>More intuitive and user-friendly</td>
</tr>
<tr>
<td>Workflow</td>
<td>Uses Chains and Agents for routing</td>
<td>Uses Nodes and Agents for task management</td>
</tr>
<tr>
<td>Data Tools</td>
<td>Comprehensive set of tools</td>
<td>Slightly fewer tools with basic features</td>
</tr>
<tr>
<td>Memory Retention</td>
<td>Multiple options for conversation history</td>
<td>Limited options with REDIS integration</td>
</tr>
<tr>
<td>Output Parsers</td>
<td>Highly flexible response structuring</td>
<td>Basic parsing with regex patterns</td>
</tr>
<tr>
<td>Debugging</td>
<td>Proprietary framework, LangSmith (beta)</td>
<td>Standard IDE debugging</td>
</tr>
<tr>
<td>Additional Features</td>
<td>Asynchronous support, autonomous agents</td>
<td>OCR support, Rasa integration</td>
</tr>
<tr>
<td>Observations</td>
<td>Some tools may crash on invalid queries</td>
<td>Handles invalid queries more gracefully</td>
</tr>
<tr>
<td>Features</td>
<td>LangChain Support</td>
<td>Haystack Support</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>LLM Support</td>
<td>OpenAI, Cohere, AI21, HuggingFace, etc</td>
<td>OpenAI, Cohere, AI21, HuggingFace, etc</td>
</tr>
<tr>
<td>Prompt Templates and Engineering</td>
<td>PromptTemplates, Custom templates, Prompt Serialization, Selectors, Partial Prompts</td>
<td>PromptNode, PromptTemplates</td>
</tr>
<tr>
<td>Process orchestration</td>
<td>Chains</td>
<td>Pipelines &amp; Ready-made pipelines</td>
</tr>
<tr>
<td>Document Stores</td>
<td>Chroma, FAISS, Elastic Search, Milvus, Pinecone, Qdrant, and Weaviate</td>
<td>Elasticsearch, FAISS, In Memory, Milvus, OpenSearch, Pinecone, SQL and Weaviate</td>
</tr>
<tr>
<td>Information Retrieval</td>
<td>Fetching Data &amp; Augmenting</td>
<td>Reader, Retriever, Ranker, and QuestionGenerator</td>
</tr>
<tr>
<td>(Semantic Search &amp; Question Answering)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployment</td>
<td>No REST API</td>
<td>REST API</td>
</tr>
<tr>
<td>Agents &amp; Memory</td>
<td>Agents to perform actions</td>
<td>Working on adding Agents to components.</td>
</tr>
<tr>
<td></td>
<td>Memory classes</td>
<td></td>
</tr>
<tr>
<td>GPU</td>
<td>DeepInfra Integration</td>
<td>Enables GPU Acceleration</td>
</tr>
<tr>
<td>Other features</td>
<td>Generic utilities e.g Python REPL, Web search API, SearxNG Search API, etc. Evaluation</td>
<td>Generative Pseudo Labelling, Evaluation</td>
</tr>
<tr>
<td>Use Cases</td>
<td>Completion, Summarization, Question-Answering, Conversational AI, Data Augmented Generation</td>
<td>Completion, Summarization, Semantic Search, Question-Answering, Conversational AI, Data Augmented Generation</td>
</tr>
</tbody>
</table>

*Sease*
• Better support for event-based ingestion and vectorization
  ○ compatible with Kafka connect as data source
  ○ compatible with Apache Camel as data source
• Micro-batching and write on vector DB out of the box
• Support for many libraries (LlamaIndex, NTLK)
• No code - already available agents
• Code - internally using LangChain
Exercise - LANGCHAIN

Exercise
https://github.com/SeaseLtd/LLM-in-search

From Natural Language to an element in a taxonomy
AGENDA 1/2

Introduction to Large Language Models (LLM)

The Open Source landscape (and repositories) for LLMs

Selecting the best LLM for your use case

Open Source frameworks and projects to interact with LLMs

Ways of adding LLMs to Search
Ways of Adding Large Language Models to Search

Why?
Why add LLM’s (Large Language Models) to your Search Platform?

1. Query Expansion (Generative/Extractive)
2. Retrieval Augmented Generation
3. Dense Retrieval Neural/Vector-based search
Dense Retrieval (neural/vector-based search)
Retrieval Augmented Generation

https://neo4j.com/developer-blog/fine-tuning-retrieval-augmented-generation/
What were the sulfur oxide emissions in Australia in 2013

GPT Generative answer is:
['Sulfur dioxide emissions', 'Air pollution', 'Environmental impact', 'Fossil fuel combustion', 'Acid rain']

GPT Extractive answer is:
{'srQMgw1_en_ss': ['1|Environment#ENV#|Air and climate#ENV_AC#'], 'dimensions_en_ss': ['Time period', 'Reference area', 'Pollutant', 'Country']}

Generate synonyms, reformulations…

Select expansion terms from taxonomies
Explainability - Highlighting

Neural highlighter MS Marco DEMO
What were the sulfur oxide emissions in Australia in 2013

```json
{
  "filters": {
    "Country": "0|Australia#AUS#",
    "Pollutant": "0|Sulphur Oxides#SOX#",
    "Variable": "0|Total man-made emissions#TOT#",
    "Year": "2013"
  }
}
```
Multi-modal search

black striped orange big cat
Stato di conservazione della tigre [it]
Deciding on how to measure success. (KPI’s)

Measure Success
How do you know you have improved your Business Search with LLM’s?

1. Search + LLMs KPI’s:
   - Operational
   - Search Session
   - Improve Search-driven Business Metrics.

2. What KPI’s specific to LLMs?
   - What Data metrics?
   - Combine Metric for Business?

3. Focus on limited KPI’s that impact business.
   - Track customers onsite Behaviors for positive or negative trends.
Deciding on how to measure success. (KPI's)

PURPOSE (Measure Success)
Search + LLMs KPIs, select those that improve Search-driven Business Metrics. What KPI's are specific to LLM's. What Data to collect or Combine Metrics? Focus on Impactful KPI's. Track onsite Searcher Behaviors for +/- Trends.

Prior to adding LLM's to your Business Search platform:
ensure a baseline of KPI's & Data for comparisons
Monitor for lift (positive) and sink (negative),
Monitor for Trends, short & long-term.
Track delta of Conversion Rate
Track delta of Life-Time Value
Track Engagement combined with renewals & conversion.
Search is the Product and Search is Data-driven for Searcher Satisfaction.
AGENDA 2/2

Popular Open Source search engines and LLMs

Rabbit Holes

Future Works
Apache Solr 9.4

- Features
  - Vector Based Search (KNN)
    - pre-filtering
    - as a Learning To Rank feature
  - Hybrid Search
  - Neural Highlighter (commercial)

JIRA ISSUES

Elasticsearch 8.11

- Features
  - Vector Based Search (KNN)
  - Pre-filtering
  - Hybrid Search (Reciprocal Rank Fusion)
  - Learned Sparse Encoder
  - Retrieval Augmented Generation (ChatGPT + simple python script)
  - End to End neural search (including ingestion+vectorization)
  - Question Answering
  - Text Classification
  - Text Similarity
  - Dedicated Machine Learning Nodes

Commercial (Platinum)
OpenSearch 2.11

- **Features**
  - Vector Based Search (KNN)
    - pre-filtering
  - End to End neural search (including ingestion+vectorization)
    - multi-modal (text/image)
  - Learned Sparse Encoder
  - Connect to Remote Models (ChatGPT, Cohere, etc) - Not possible to use custom LLM
  - Hybrid Search
  - Conversational Search (Conversation Memory + Retrieval Augmented Generation)
    - As of OpenSearch 2.11, the RAG technique has only been tested with OpenAI models and the Anthropic Claude model on Amazon Bedrock.
Vespa

- Features
  - Vector Based Search (KNN)
    - pre-filtering
  - End to End neural search (including ingestion+vectorization)
  - Hybrid Search (Reciprocal Rank Fusion)
OPEN SOURCE VECTOR DATA BASES

- Milvus
  - Vector Based Search
- Weaviate
  - Vector Based Search
  - Generative Search
  - Weaviate modules
- Qdrant
  - Vector Based Search
  - Quaterion (fine-tuning framework)
LANGSTREAM - RAG in Apache Solr

Exercise


- brew install LangStream/langstream/langstream
- docker run --rm -p 8983:8983 --rm solr:9.3.0 -c
- export OPEN_AI_ACCESS_KEY=<your-openai-api-key>
- export SOLR_HOST=host.docker.internal
- langstream -v docker run test -app
  /Users/sease/PythonProjects/SearchSolutions/2023/langstream/examples/applications/query-solr
  -s examples/secrets/secrets.yaml
AGENDA 2/2

Popular Open Source search engines and LLMs

Rabbit Holes

Future Works
RABBIT HOLES

PURPOSE
identify critical decisions and avoid wasting much time and resources
Do NOT go down that Rabbit Hole: Danger, Will Robinson, Danger!

1. Not measuring search metrics offline/online
2. Choosing the wrong technology for your problem
3. Choosing the wrong Large Language Model for your problem
MEASURING SEARCH QUALITY

PROBLEM
How can I measure if I am bringing improvements to my search system?
Choosing the wrong Large Language Model for your problem

**RABBIT HOLE**

No offline/online search quality evaluation. The new system is evaluated based on vibe, feelings, subjective opinions etc… A lot of time and resources can be spent for un-promising solutions. Promising solutions may be discarded just because of superficial analysis.
BETTER TO...
Set up an offline and online evaluation system that drives research and development
PROBLEM
What technology should I use to implement Large Language Models in search?

- Should I use a traditional search engine that supports also vectors?
- Should I use a dedicated vector search engine?
- What kind of inference service should I use?
- …
Choosing the wrong technology for your problem

Choosing the wrong Large Language Model for your problem

RABBIT HOLE
A technology is chosen because:

- cool/popular
- already used
- imposed from above

You end up wasting a lot of time and resources trying to make such technology work in your use case.
You accumulate technical debt, team and business discontent.
You end up failing and having to go back and revisit the tech stack and architecture with immense costs.

CHOOSE THE RIGHT TECHNOLOGY
BETTER TO...

Carefully assess the technologic landscape, identify the most promising softwares that satisfy your needs and proceed with a comparison both functional and performance-wise.

The Large Language Model and vector-search domain is going so fast, especially in the open source realm, so this activity is crucial and deserve the right amount of time and attention.
PROBLEM
What model should I use?
Choosing the wrong technology for your problem

Choosing the wrong Large Language Model for your problem

RABBIT HOLE
A model is chosen:

- from the shelf
- out of context
- based on popularity

You end up wasting a lot of time and resources trying to make such model work in your use case.
You use it as it is accumulating team and business discontent.
You spend a lot of time fine-tuning it even if it’s not meant to be.
You end up failing and having to go back and choose a different model.
BETTER TO...
Carefully assess the models available, starting from the pre-training:
- you want a domain that is as close as possible to yours

The look for fine-tuned examples, potentially close to your use case.
Iterate and refine if additional fine tuning is necessary.
Investigate the models deeply, understanding the datasets used for both pre-training and fine tuning.
Look at those datasets!
Compare different candidate models on your data.
Popular Open Source search engines and LLMs

Rabbit Holes

Future Works
Apache Solr Roadmap

- END-TO-END NEURAL SEARCH
- BETTER HYBRID SEARCH
- LARGE LANGUAGE MODEL QUERY REWRITER
- RETRIEVAL AUGMENTED GENERATION
- LUCENE MULTI-VALUED VECTORS

- LANGCHAIN Document Store
- HAYSTACK Document Store

NEED HELP IMPLEMENTING LLMs FOR SEARCH?
Our collaboration with John aims to give our customers comprehensive guidance and implementation capabilities to design and develop LLMs integrations:

WHAT'S NEXT

- PROJECT/PRODUCT MANAGEMENT
- ENGINEERING DIRECTION AND ARCHITECTING
- DESIGN AND DEVELOPMENT
To Wrap it Up

Introduction to Large Language Models (LLM)

The Open Source landscape (and repositories) for LLMs

Selecting the best LLM for your use case

Open Source frameworks and projects to interact with LLMs

Ways of adding LLMs to Search
To Wrap it Up

Popular Open Source search engines and LLMs

Rabbit Holes

Future Works
THANK YOU!

SCAN ME

@seaseltd @sease-ltd @seaseltd @sease_ltd