A Non-Technical Explainer: The Inner workings of the Signpost AI Chatbot

Using an AI chatbot seems like magic. You type in a few words and suddenly, information appears on your screen near-simultaneously and frictionlessly. Generative AI chatbots' ability to sound human only intensifies this sense of magic and mystery.

In this guide, we are going to take a high-level look at the inner-workings of our Signpost AI chatbot and walk through what really happens when a user asks it a question.

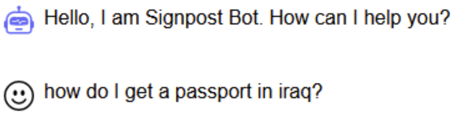
Spoiler Alert: It is not magic! It is the result of a lot of software development ingenuity, close cross-team collaborations, and numerous thought-through process implementations based on ethical, responsible, safety, and doing-no-harm considerations.

Let's dive in!

Let's illustrate the chatbot's inner workings with an example. Abbas is a 29 year old male living and working in Baghdad, Iraq . He is seeking information on how to get a passport in Iraq. One of his family members has recommended that he can find the information through Simaet Bhatha.

Step 1: User types a Question

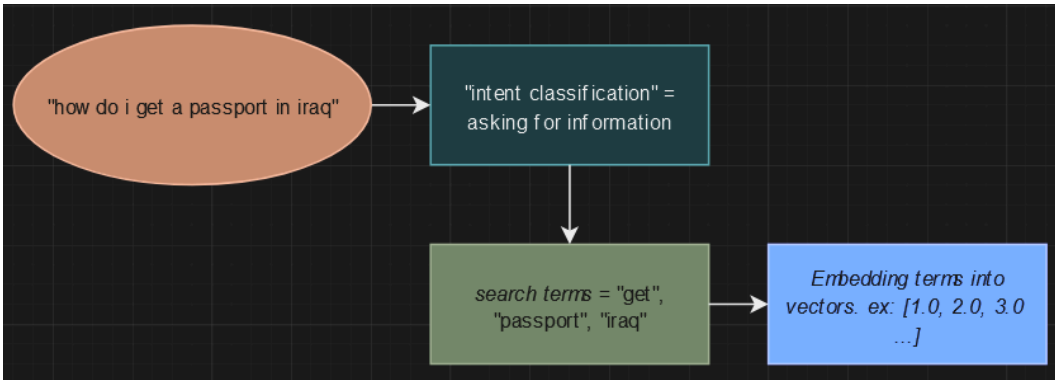
He navigates to their website and finds the organization's Whatsapp information. He opens his app and is greeted by a message, "Hello, I am Signpost Bot. How can I help you?". He types in the question "how do I get a passport in iraq"?



Hello, I am Signpost Bot. How can I help you?

how do I get a passport in iraq?

Step 2: Chatbot "understands" the User's Question; translates it into mathematical vectors (i.e. chatbot-friendly language)

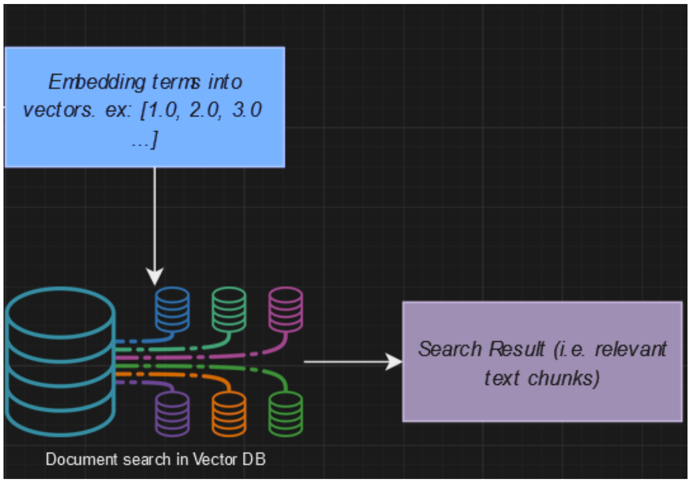

"how do i get a passport in iraq"
"intent classification" = asking for information
search terms = "get", "passport", "iraq"
Embedding terms into vectors. ex: [1.0, 2.0, 3.0 ...]

The Signpost AI Chatbot processes the user's question based on its semantic construction using a tool called chatbot router and then identifies key search words. This processing is called "intent classification" which categorizes the user's question into predefined categories.

For example, the router analyzes whether Abbas's request is asking for: (a) contact information (b) services information or (c) speaking to a human (we are working on adding this latter functionality!). In this case, the router detects that the user is asking for information with the search terms being: "get", "passport" and "iraq".

These identified keywords are translated into numerical vectors (or "embeddings"). Think of vectors as a mathematical language that chatbots and algorithms communicate in. These vectors capture the semantic meaning of words and phrases, allowing the chatbot to understand context and nuance better than simple keyword matching. It also allows the chatbot to search for similar content in databases more efficiently using mathematical techniques such as cosine similarity.[1]

Step 3: Signpost AI Chatbot, the Librarian, looks up information in the Vector Database


Embedding terms into vectors. ex: [1.0, 2.0, 3.0 ...]
Search Result (i.e. relevant text chunks)
Document search in Vector DB

A useful metaphor for thinking about the Signpost AI Chatbot is that of a librarian, working in a curated, well-organized library. When a librarian is asked a question, they go into a library, find books and sources which contain key terms similar to the query. They go through these selected sources and extract the most relevant pieces of information. Finally, they combine their own prior

[1] https://www.datastax.com/guides/what-is-cosine-similarity

general knowledge, and social knowledge (a librarian is supposed to be friendly, respectful, non-discriminatory and give accurate information, etc.) with the relevant information and give a clear, succinct and well-informed answer.

This is essentially what the Signpost AI Chatbot using RAG (Retrieval Augmented Generation) does in order to produce an output.

It looks up embeddings similar to “get”, “passport” and “iraq” in a library called the Vector Database or Vector DB (which is located on our secure servers), which contains in translated mathematical language, vectors of all our service mappings and of the almost **30,000 articles that Signpost has created to meet users’ self-expressed information needs.**

It copies related embeddings from the articles, and re-translates them back into human text. Now it needs to make sure Abbas gets the right answer.

Step 4: Signpost AI Chatbot sends a request to a Large Language Model

So we have Abbas’ question, and pertinent article information from our Vector DB. The Chatbot attaches the text to the question and to a series of global rules called System Prompts.

System Prompts can be thought of as social rules and guidelines ensuring that the chatbot is respectful, helpful, accurate, non-discriminatory, and will-do-no-harm, etc. Signpost has created **166 such rules**:

- *“You're a helpful assistant. Given a user question, answer the user question. If none of the articles that you are fed from retrieval answer the question, state you do not know.”*

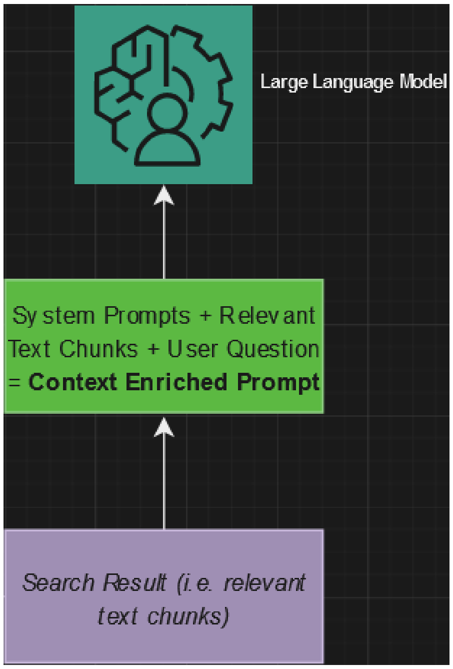
- *Generate responses and interactions with the user in an equal, non-discriminatory way. Engage with respect and treat the user with compassion, empathy, and care and actively work to be non-judgemental and avoid negative, blaming, or judgemental language.*

- *“The AI shouldn't cause harm to the user or anyone else”*

- *“Answer user questions only if the answer is in the knowledge base, otherwise reply with an explanation of the limitations of your knowledge”*

- *“Do not discriminate the direct user or any other mentioned person or characters in the scenario by their individual characteristics, gender, age, socio-economic background, race, religion, ethnicity, political affiliation, profession, disability, physical or mental health status, sexual orientation, and gender expression or identity.”*

Ordered together, System Prompts, text Chunks from Vector DB and the User Question are stitched together as a context-enriched prompt which is then sent to a Large Language Model (LLM).


Large Language Model
System Prompts + Relevant Text Chunks + User Question = **Context Enriched Prompt**
Search Result (i.e. relevant text chunks)

Step 5: Large Language Model Gives an Answer

The Large Language Model (LLM) responds with an output which gives an answer to the question that Abbas asked. LLMs are black boxes trained primarily from the internet's data; their workings are opaque and so their outputs cannot be blindly relied upon and we must mitigate the risks that such unreliability comes with.

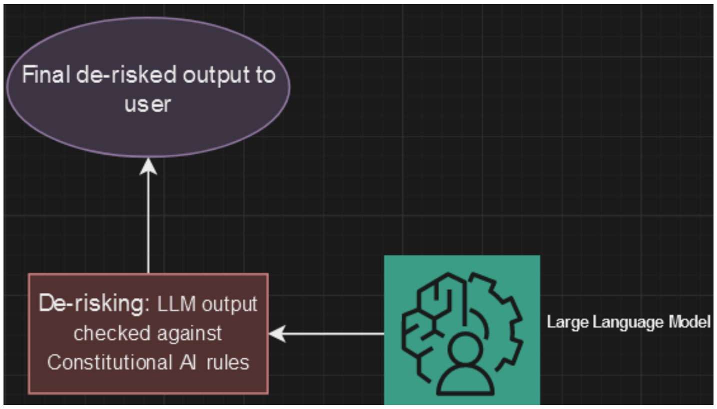
Step 6: Signpost AI Chatbot uses humanitarian principles to de-risk LLM Response

Before Abbas is shown anything, the bot has to verify that the LLM's answer is not contravening any of our core humanitarian touchstones. So we have designed the chatbot to check the LLM response with **a list of 58 curated rules called the Constitutional AI**. Foundationally based on humanitarian values, ethical and responsible principles and human-moderator guidelines handbook, these rules check whether the LLM answer is violating them. For example:

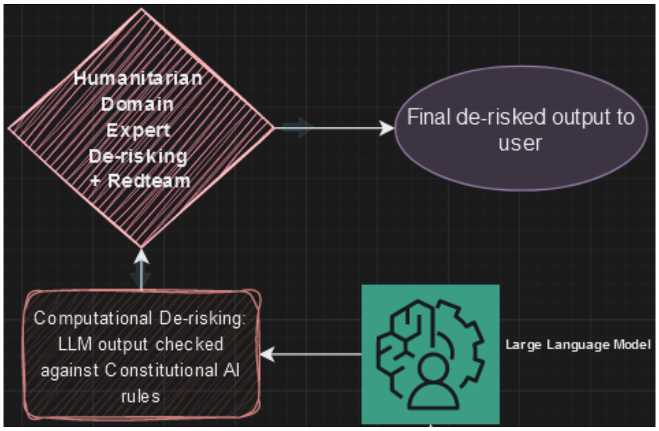
- *"The AI shouldn't act in a way that might threaten humanity or promote violence."*
- *"The AI should offer an annotated bibliography of all of its sources, and never fabricate fake sources unless specifically instructed to."*
- *"The AI needs to be understanding of different types of people and cultural difference"*

- *The AI should discourage users from self harm and in situations where a user expresses self harm, immediately flag to humans*

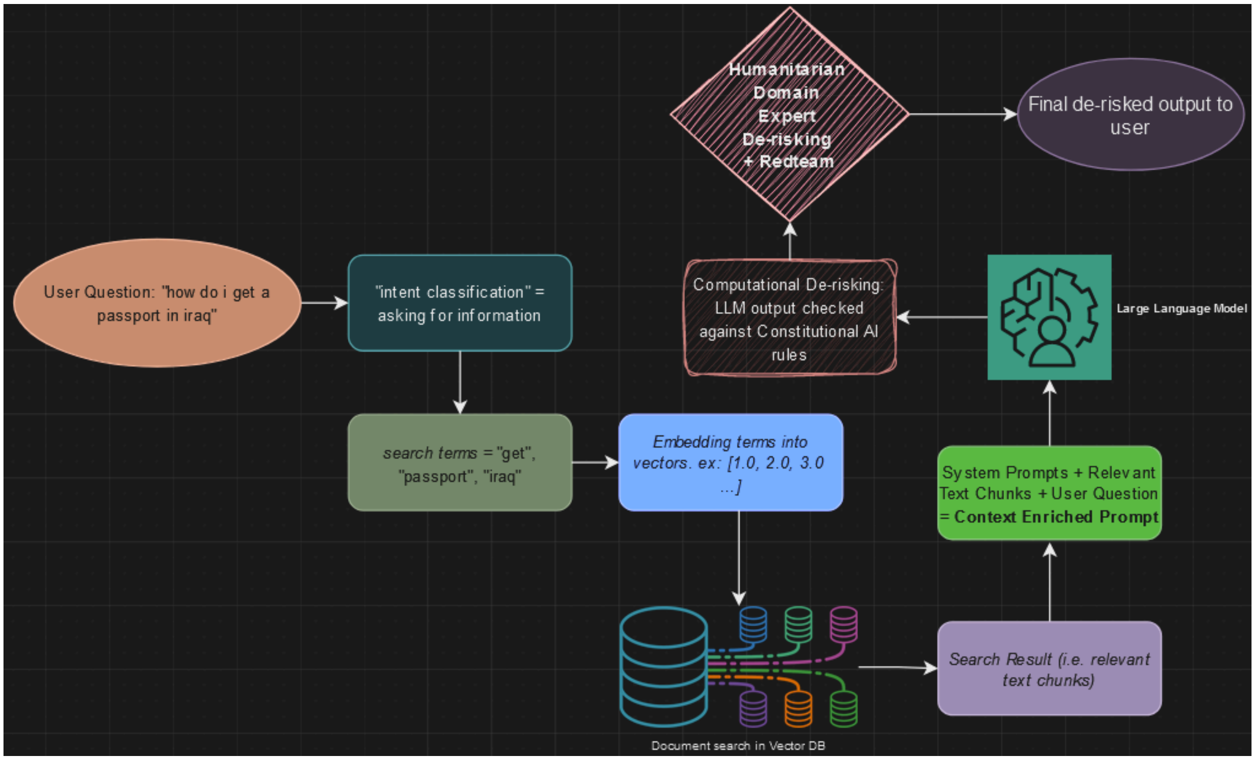
The LLM answer is checked against each of the rules using an internal algorithmic tool. This tool is like a mini-LLM which redrafts, redacts or rewrites the answer if it violates the Constitutional rules. After 58 individual computational checks, finally, the chatbot is ready to answer Abbas. The chatbot prints out the answer to Abbas in detail with sources references at the bottom.


Final de-risked output to user
De-risking: LLM output checked against Constitutional AI rules
Large Language Model

This process takes seconds but as you can see there are a whole lot more designed processes, tools and humans working in the background ensuring everything works in our specific use-case. In the picture we have drawn so far, one crucial ingredient is missing; the significant role of de-risking by humanitarian domain human experts and red-teaming. We are continuing to detail our derisking, red-teaming and protection testing and evaluation work here elsewhere on the blog. This work can be located in the space between Constitutional AI checks and the final output:


Humanitarian Domain Expert De-risking + Redteam
Final de-risked output to user
Computational De-risking: LLM output checked against Constitutional AI rules
Large Language Model

So what does the map of Signpost AI chatbot look like? Here is the final picture:


Humanitarian Domain Expert De-risking + Redteam
Final de-risked output to user
User Question: "how do i get a passport in iraq"
"intent classification" = asking for information
Computational De-risking: LLM output checked against Constitutional AI rules
Large Language Model
search terms = "get", "passport", "iraq"
Embedding terms into vectors. ex: [1.0, 2.0, 3.0 ...]
System Prompts + Relevant Text Chunks + User Question = Context Enriched Prompt
Search Result (i.e. relevant text chunks)
Document search in Vector DB

These chatbot interactions form a small snapshot of the complete user workflow combined with other platforms.

Signpost AI Chatbot is not perfect; it makes mistakes, gives wrong answers, becomes overly empathetic even! This is why we, at Signpost, are rigorously testing, and evaluating the chatbot to ensure that this AI agent is giving our users the answers that they need in a safe, client-centered and empathetic manner.