Hi, I am a chatbot, how can I help you?
Enhancing reliability and traceability of Large Language Models

Anja Vrečer, Novo Nordisk A/S, Søborg, Denmark
Vicky Poulsen, Novo Nordisk A/S, Søborg, Denmark

ABSTRACT

The capability of Large Language Models (LLMs) to process large volume of text, retrieve valuable information and generate human-readable text and their versatility have offered the pharmaceutical industry numerous new opportunities, from optimising existing processes to accelerating drug discovery and development.
In Novo Nordisk (NN), we are exploring the possibilities of harnessing the power of LLMs in various areas. In this presentation, we will demonstrate our attempt in developing a chatbot that provides end-user support to the day-to-day operation of SDTM automation. Other than providing reliable SDTM related information based on internal and trusted external sources, our ambition is to build a chatbot that can act as first-level support in triaging SDTM generation issues. We will also share how we intend to overcome the known limitations of LLMs and incorporate transparency in the solution to boost confidence in its use.

INTRODUCTION

In 2017, Novo Nordisk implemented an SDTM Generation Framework (the Framework) that automates SDTM and define.xml generation using a metadata-driven approach. The scope of the Framework was expanded to cover SDRG generation in 2023. In addition to the ever-growing number of studies supported by the Framework, the increased complexity of study design and diversified data sources as well as implementation of multiple SDTM IG versions, have created growing demand for end-user support. Inspired by the potentials of Large Language Models, a cross-department initiative has been launched to explore the possibilities of using a chatbot to meet the need.

LARGE LANGUAGE MODELS

Large Language Models (LLMs) are advanced artificial intelligence (AI) systems which can process and generate human-like text. They are trained on large amounts of text which enables them to “learn” to understand it and generate an appropriate text response. This can be text continuation, question answering, text summarization, translation or other tasks based on the instructions given to them during training and the task the end-user is aiming to solve.

If we break down the term, Language Model (LM) means that the basis of the system’s capabilities are mathematical or numerical representations of the language patterns that were learned during the training. In its essence, LM is a machine learning model, based on neural networks, which means that every word the model responds with is a result of a game of probabilities. In simple terms, the way language model generates an answer is that it calculates the next most probable words, based on the previously given (and generated) words. Therefore, the model’s effectiveness corresponds to the amount and quality of the text it was trained on. The term Language Model was given an additional adjective “Large” to emphasize that these LMs require a large amount of training text and parameters.

Due to their immense size, training and maintaining these models is a highly extensive and computationally intensive task. It requires substantial computational resources, including powerful GPUs or TPUs, large-scale data storage, and includes significant energy consumption making this option currently beyond the reach of an average computer user. Fortunately, large companies and organizations provide a large collection of pre-trained LLMs, available both for direct download and use, for further fine-tuning or as Application Programming Interfaces (APIs). The differences between the LLMs from different providers are mainly architecture distinctions, parameter sizes or different text sources. To distinguish between them, they are given names such as GPT-3.5, GPT-4, Llama, Bert, etc.
The easiest, computationally least expensive and fastest way to use LLMs is through APIs. This means that providers host their LLMs on their own server and expose an endpoint where the developers can connect to their LLMs. Developers can then simply interact with the LLM on the server through API, without needing to host or manage the LLM themselves. A popular and one of the most widely recognised providers of the LLM APIs is OpenAI[1]. They enable developers to connect to one of their models, such as GPT-3.5 or GPT-4 and easily interact with them using their API endpoint.

Through the advancement of LLMs and Generative AI (GenAI), which is a term encompassing all AI systems that generate content, new ways and opportunities to harness their abilities have introduced themselves. This can include text summarization, image, audio or video generation from prompts, or even generation of code from descriptions. However, the most prevalent way to use these models still remains the initially introduced text generating system with chat-like interface. These systems utilize the text-generating ability of LLMs to answer user's questions, making them a convenient tool for quick information retrieval. In this paper, we focus on this type of GenAI, namely, an end-user support chatbot.

LLMS IN NOVO NORDISK

Since LLMs and GenAI systems have demonstrated their effectiveness across a diverse range of tasks, we at Novo Nordisk (NN), have started exploring how to leverage these technologies to enhance our operation and drive innovation. An example of such a tool is the Novo Nordisk ChatGPT, shown in Figure 1. It is an application available to all the Novo Nordisk employees and it comprises three options for chatting: General Chat, for most common tasks, Detailed Chat, for more complex tasks and Custom Chat, where the user chooses the model, system prompt and temperature. All the chat options also give the user the ability to attach documents or figures to their prompt. Additionally, the interface provides the option to generate images based on the description.

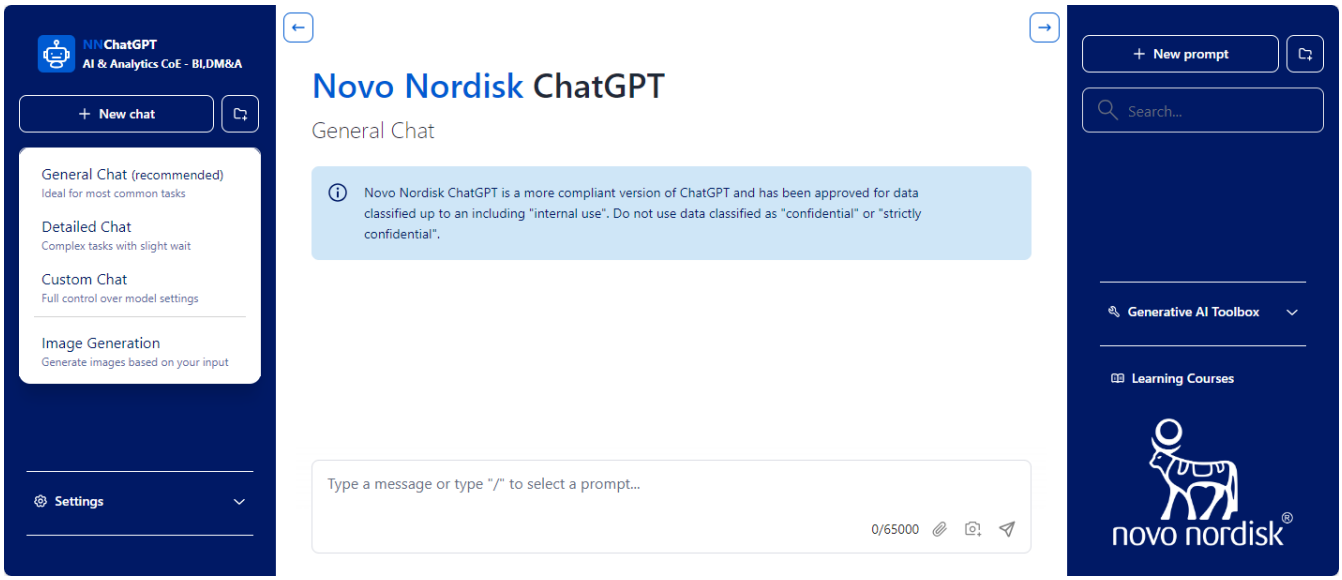

Figure 1: Novo Nordisk ChatGPT

The main motivation for developing this solution was to provide NN employees with an LLM-driven chat interface that is more compliant than the original ChatGPT and has been approved for internal data and information. Additionally, the solution can be better controlled to be trustworthy and ethical. This is the aspiration for all the NN's AI systems since a set of data and AI ethics principles were established to guide the AI related business processes and practices which all the NN AI systems must follow.

WHY A DEDICATED CHATBOT?

Although NN provides a compliant version of ChatGPT which can be used for general purposes, it comes with certain limitations. The primary obstacle is that it generates generalised answers, making it ineffective for a specific use-case such as ours. Additionally, it is configured in a way that it prefers to guess or give suggestions of possible answers than to ask for clarification or admit it does not have enough information to respond. The configuration also allows the model to generate different answers for the same question, making it unreliable and inconsistent. Furthermore, it does not provide the sources of answers, which would further enhance the answers' reliability and increase the users' confidence in them. This is especially important since if the users cannot trust that the system is reliable and that it generates true information, they will not use it. Also, as mistakes are an unavoidable part of such AI systems, we must always make sure to provide users the opportunity to provide feedback about false answers and continuously improve the chatbot, so it does not repeatedly spread misinformation. This feature is unfortunately not available in most chatbot interfaces.

For these reasons, we developed a dedicated chatbot, VEDAM (Virtual Expert Data Manager), to solve these problems with the ambition to provide reliable end-user support to the day-to-day operation of the Framework. The next sections provide more insight into each of the problems of the general chatbots mentioned above, describe why it is crucial we address them to design a reliable chatbot and illustrate how our solution aims to solve each problem.

SPECIFICITY

In general, LLMs are pre-trained on large amounts of text. These can be, but are not limited to, books, websites, scientific papers and databases with examples of interactions, spanning a diverse range of topics. The more information LLMs are exposed to during training, the more they "know". Since most of the current LLMs are pre-trained on huge text corpora, typically at least 1000 GB in size, they have general knowledge about a great variety of different fields. After pre-training, they are usually fine-tuned on a specific task, such as question answering and are aligned with the human preferences of helpfulness, harmlessness and honesty. LLMs can also be fine-tuned on specific questions and answers, however as this is a computationally expensive task that requires a lot of quality examples, developers usually use generic LLMs prepared for chat use-case for designing chatbots. Even though these chatbots "know" a lot about most topics, they mostly lack specific domain-knowledge. This means that they either might not have been exposed to domain-specific texts or they do not adjust their answers to a particular end-user.

When we want to use a chatbot in a domain-specific setting, such as end-user support with SDTM automation, generic answers are not sufficient. There are a few reasons for this. One is that without additional explanation, the chatbot cannot know what domain the users are interested in. Thus, the users must include this detail in their prompt every time they have a question. However, sufficiently describing the domain and use-case introduces an additional burden for the users which can result in a reluctancy of using this system. Furthermore, even when such description is provided and the generic chatbot has some knowledge about the desired domain, it cannot have any information about the particular tools, implementations or other specifics the users might have questions about. Thus, the general chatbot proves ineffective for domain-specific questions.

One way of "teaching" the chatbot, or more precisely the underlying LLM, domain-specific information is fine-tuning. However, as discussed, this is a computationally and spatially extremely expensive task that is unattainable for average chatbot developers. Therefore, developers usually employ another technique for refining chatbot's answers, namely, using prompting. This is a technique that involves providing the chatbot additional guidance or instructions that are added in the background and are invisible to the end-users. They help chatbot decide what aspect of the question to focus on when answering and what domain the answers should come from. We utilise this technique in the development of VEDAM. At initialization, VEDAM is given a short but comprehensive description of the users' domain. Additional contextual prompting, meaning that the chatbot has access to additional documents or sources which it can use as a source of information when answering questions, is also used to enrich VEDAM with understanding of the Framework and SDTM in general. The documents or sources include internal documents, such as training materials/transcripts and guidelines, documents from external trusted websites such as CDISC and FDA, as well as questions and answers in connection with support requests to the Framework. In the future, we will additionally explore the ability to find answers directly in the standard programs of the Framework.

To illustrate the difference between a general and dedicated chatbot, we asked both the general NN ChatGPT and VEDAM a domain-specific question "What is CRF?" and compare their answers. Figure 2 shows the answer given by the general NN ChatGPT and Figure 3 shows the answer generated by VEDAM. Since the abbreviation CRF can have multiple meanings, the NN chatbot decided to give an explanation that is probably the most common for general users. Although the answer is not incorrect, it is not the most relevant in the context of SDTM. On the contrary, VEDAM responded with a relevant answer.

Figure 2: Example of a domain-specific question and an answer given by the general NN ChatGPT.

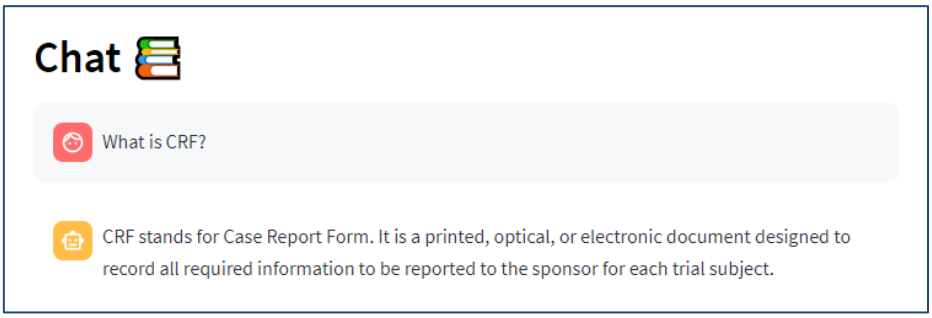

Figure 3: Example of a domain-specific question and an answer given by VEDAM.

CONSISTENCY & RELIABILITY

Simply put, LLM generates text by calculating probabilities of the next words, based on the given and previously generated text. Therefore, the general LLM-based chatbot without further superintendence always tries to answer the question, conditioning on whatever it was trained on. This means that even if it does not have adequate information to answer or if the question is ambiguous, it returns the most likely answer or gives suggestions for answers. Furthermore, most general LLMs are configurated in a way that allows them to be creative which enhances the likelihood of more innovative and interesting answers. Although this might result in more engaging answers it also means they are ultimately less deterministic. Consequently, the same question can yield multiple possible answers.

It can sometimes be useful if a chatbot gives suggestions of possible answers when posed with an unclear or ambiguous question. But the problem is that it can, in fact, introduce additional obscurity for the users. What is even more concerning is that if the chatbot tries to guess the answer, it can produce false, fictitious, or partly true information contributing the distribution of misinformation and jeopardising its reliability. This is a common problem of the LLMs, also known as hallucination. Furthermore, if the chatbot is allowed too much creativity, the answers to the same question can be too diverse, which can be confusing for the users. Consistent answers, on the other hand, make the chatbot more reliable.

We solve the problem of inconsistent answers by limiting the creativity and prioritizing the learned patterns of

VEDAM. This is determined with a LLM parameter called temperature, which sets the predictability of the generated text. By setting the temperature to 0, VEDAM does not look for alternative forms of answer, but always answers consistently. Since it uses additional files as a source of knowledge, the users can also trust that the generated answers are reliable. This is further enhanced by the fact that whenever the user asks something that none of the source files contains, VEDAM admits its ignorance and requests additional clarification. To add an additional layer of reliability, we add categories to the source documents, since files from different categories can contain distinct information about a certain topic. In case that multiple files from different categories provide different answer to the same question, VEDAM asks for clarification by asking the user which aspect of the question they are interested in. In the future, we will also investigate the possibility to extend this functionality by automatically selecting relevant categories based on the user's profile.

To demonstrate the importance of reliability and answering only when the model has enough knowledge to answer, we provide an example of how the general NN ChatGPT and VEDAM answer a rather irrelevant question, "What is the abbreviation for a foot size domain in SDTM?". Since this domain does not exist, we expect the chatbot to not answer anything. However, as shown in Figure 4, the general NN ChatGPT just makes up an answer that the abbreviation of a foot size domain is "FS". On the contrary, VEDAM admits that none of the provided files contains information about foot size domain in SDTM, which is the expected and desired behaviour of a reliable chatbot (Figure 5).

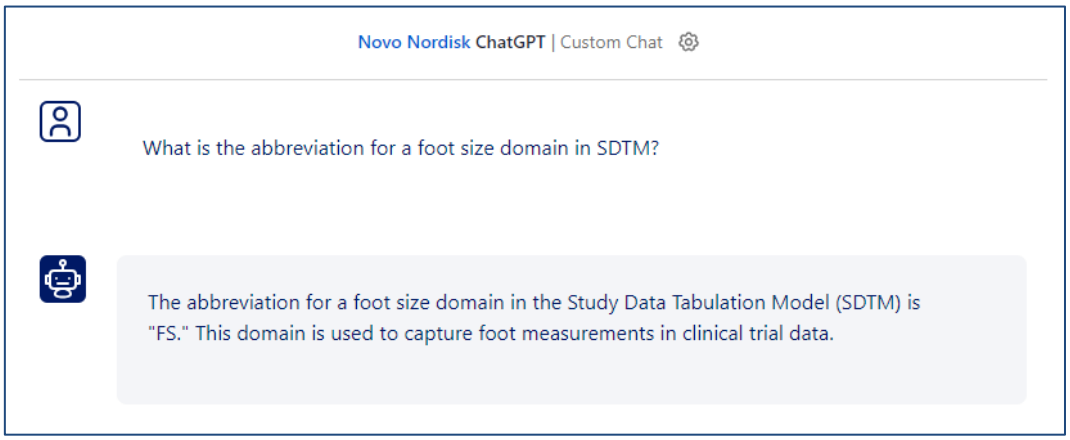

Figure 4: Example of an incorrect answer provided by general NN ChatGPT, caused by the system's over-confidence.

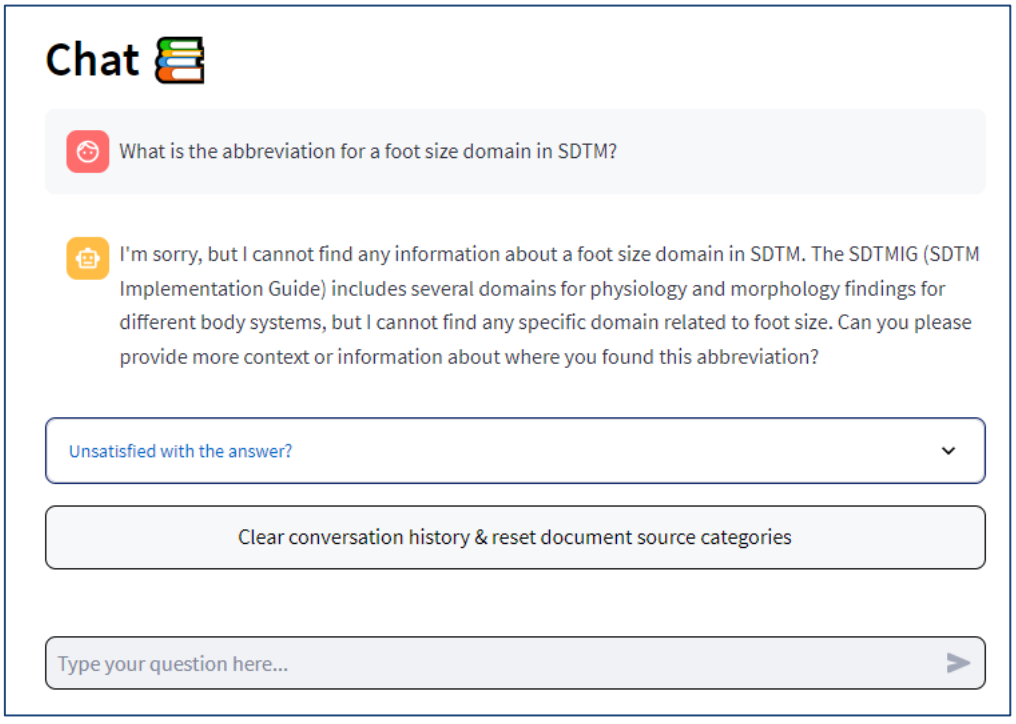

Figure 5: Example of a question that VEDAM does not have an answer to.

TRACEABILITY

Traceability is a chatbot's ability to provide sources or documents its answer is based on. Since the general NN ChatGPT does not use any additional documents as the source of knowledge, it cannot link its answers to any source. It does not provide any information about what other text sources influenced its decisions either. Thus, the users cannot get any insight into the reason why the chatbot answered the way it did. Traceability is, therefore, tightly related to reliability - the more relevant sources the chatbot can provide as reference, the more reliable the answer is.

The underlying mechanism of how LLMs generate text relies on thousands of internal parameters and representations of language, which are impossible to be easily understood or explained by a human. Consequently, it is impossible to directly trace the generated text to the sources that an LLM learned certain information from. One way to get this detail, is to directly instruct the model to cite the source next to the answer. While this option is the simplest to establish, it can cause the chatbot's answers to be incomprehensible. The user might not be able to see how many and which documents the answer is based on. The model would also be likely to leave out some of the less relevant texts and would not have a way of indicating how relevant each text is. An alternative that we use in VEDAM is to have a separate retrieval algorithm working together with the main LLM to find the relevant texts and provide them as source. Since the main LLM only answers the question if it can find the answer in any of the given source files, the answers can only be based on those internal sources. The retrieval algorithm can therefore independently find the files that the answer is based on and append them as reference to the final answer. This way, the users can easily see all the relevant files and confirm that the answer is reliable.

Figure 6 displays an example of how VEDAM provides document sources next to the answer. We asked, "How to name an SDRG for clinical data?". As shown in the figure, VEDAM found one relevant document and attached a button with its name after the answer. If the user hovers over the document name, the details of the document are displayed as shown in Figure 7. The details include the full document name, its category, relevancy to the question which is displayed as percentage in the brackets, and the relevant sections from the document. Displaying document names additionally follows colour coding to indicate the relevance of each referenced file.

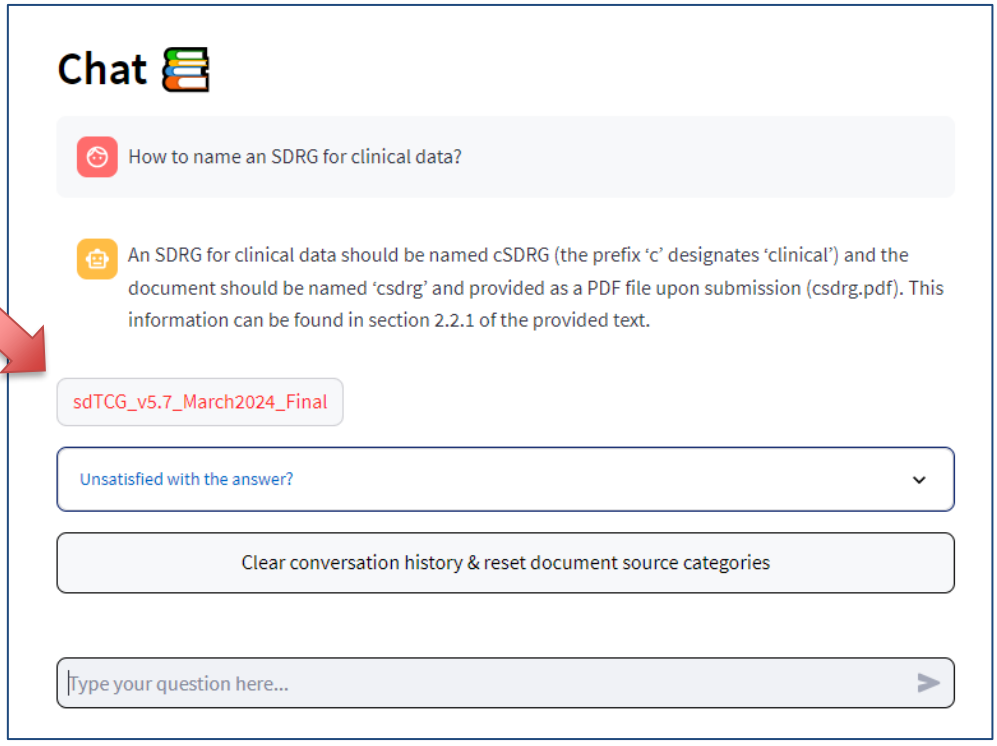

Figure 6: Example of a question and answer where VEDAM provides a source.

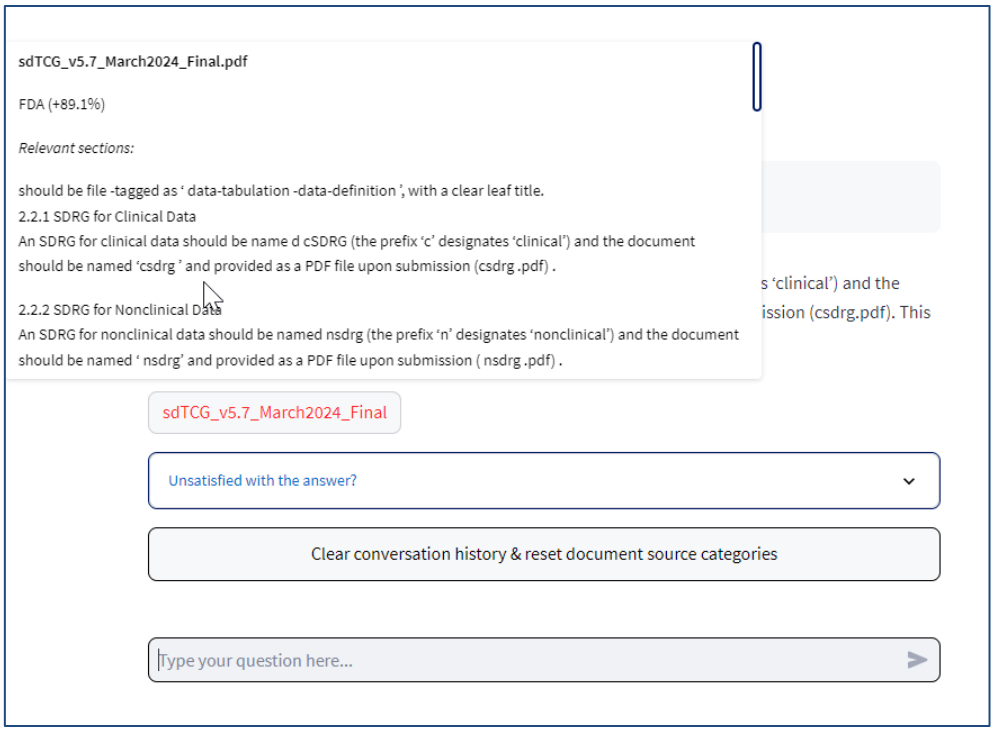

Figure 7: Example of the source details for the question and answer in Figure 6.

FEEDBACK-AWARENESS

Most of the currently available chatbot interfaces do not sufficiently acknowledge the users with the fact that the chatbot does not know everything and that it can sometimes still make mistakes. And even if attentive users recognise a mistake in the chatbot's answer, the interface does not allow them to report it. This means that if the same or different user asks the chatbot the same question again, the answer will remain incorrect. The only way users of such systems can remark these observations is to send their comments directly to the administrators or maintainers of the chatbot.

Incorporating user feedback into the evolvement of the chatbot is an important aspect of a reliable chatbot. The fact that every chatbot can make a mistake should not discourage the potential users from finding reliable answers. Instead, the chatbot should make the users aware of possible errors and encourage them to confirm the answers with the provided context. That is why we include an option to submit a comment or report an incorrect answer directly from VEDAM's interface, as shown in Figure 8. The button "Unsatisfied with the answer" is expanded upon pressing to allow the user to submit their feedback. Currently, the feedback, along with the user's initial question and the VEDAM's answer, are sent to the chatbot administrator for review. Afterwards, the administrator can investigate the source of the error and aim to resolve it. If the issue relates to incorrect information in any of the documents, they can revise the information in the document or remove it from VEDAM knowledge database. The documents that the VEDAM is using as knowledge source should be curated and continuously updated since our system currently does not have a mechanism to detect false or outdated information. One way we could ease the investigation and maintenance of the documents would be to add an option to directly report a document with false information in the interface. The document could then be quarantined and not used as source of information until it is reviewed. Additionally, we could create a separate machine learning (ML) system that can automatically detect outdated information and suggest review to the chatbot administrator.

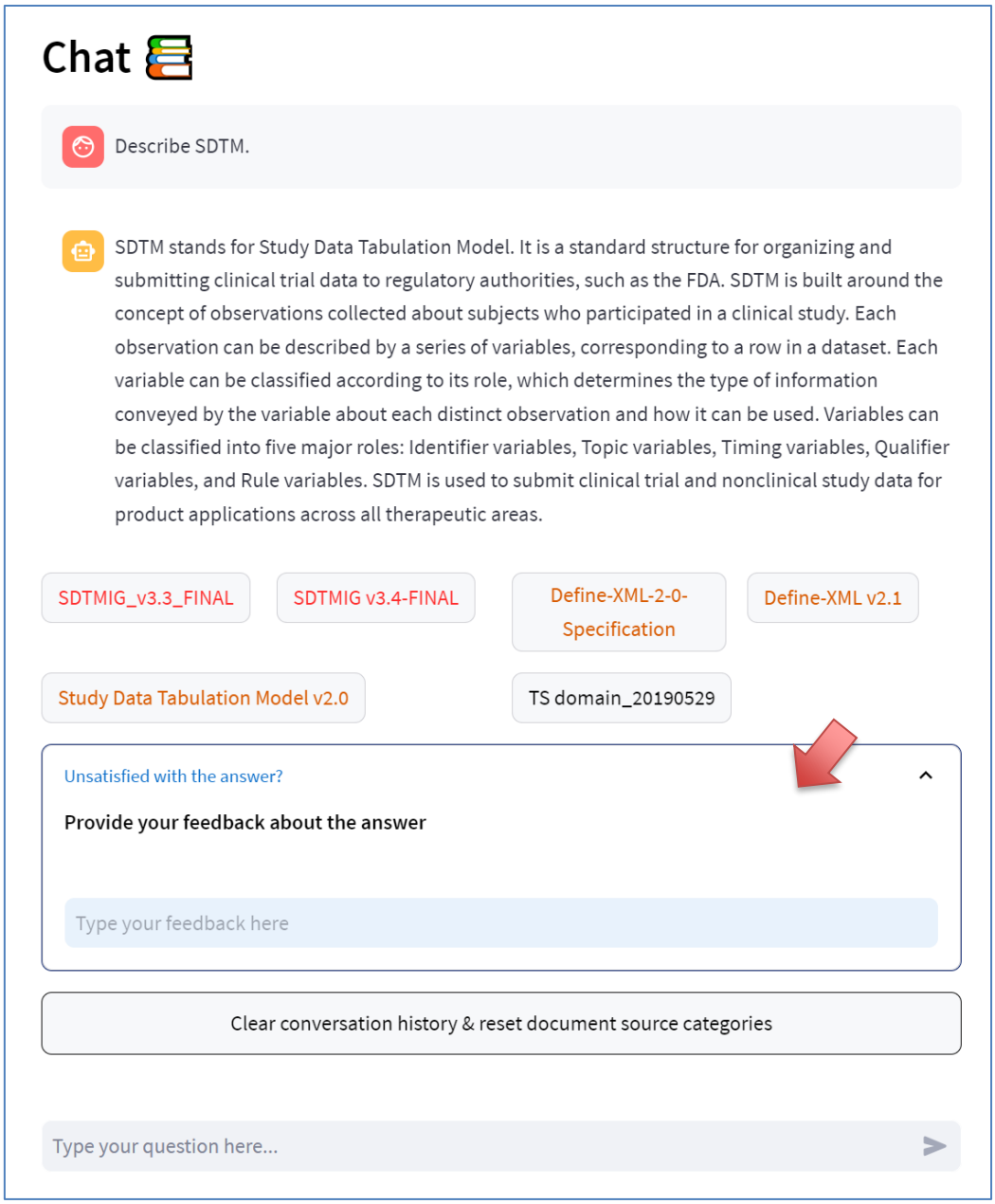

Figure 8: The option to provide feedback in VEDAM interface.

CONCLUSION

LLMs/Chatbots have become an increasingly popular tool for information retrieval and solving text generation tasks both for personal and professional use. However, most general chatbots do not meet the requirements crucial to be useful in a specific professional environment such as end-user support for SDTM Generation Framework. Therefore, we introduce a domain-specific, reliable, traceable and feedback-aware chatbot, VEDAM. VEDAM is especially designed for our use-case with features to enhance its reliability and traceability, but it heavily relies on the accuracy and timeliness of the information in the documents it uses as reference. Therefore, constant curation and maintenance of the knowledge database is required. Lastly, users' input can play a key role to ensure continuous improvement of VEDAM.

REFERENCES

1) https://platform.openai.com

ACKNOWLEDGMENTS

VEDAM is developed in collaboration with the Data Systems and Standards department of Novo Nordisk.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:
Anja Vrečer
ajvr@novonordisk.com
Novo Nordisk A/S

Vicky Poulsen
vicp@novonordisk.com
Novo Nordisk A/S