SYSTEM AND METHOD FOR ACCURATE AI CONTENT GENERATION Current Chatbots are Incapable of Reliably Producing Accurate Information

OpenAI's ChatGPT is the most popular AI content generator. Yet, ChatGPT cannot be trusted to generate *accurate* content. In mid-2023, Open AI's CEO, Sam Altman, stated, "I probably trust the answers that come out of ChatGPT the least of anybody on Earth." (Source: https://apnews.com/article/artificial-intelligence-hallucination-chatbots-chatgpt-falsehoods-ac4672c5b06e6f91050aa46ee731bcf4, last visited March 3, 2024). In fact, the night before ChatGPT's launch, OpenAI's lead data scientist "didn't know if it was any good" due to its inability to provide factual responses. "[W]hen we made ChatGPT, I didn't know if it was going to be so unimpressive that people would say, 'Why are you doing this? This is so boring!'" (Source: https://www.technologyreview.com/2023/10/26/1082398/exclusive-ilya-sutskever-openais-chief-scientist-on-his-hopes-and-fears-for-the-future-of-ai/, last visited March 3, 2024).

Remarkably, the world-wide explosion of ChatGPT's popularity is being fueled by the desire to use it for factual content generation — something it is not even designed for — something it literally *cannot* do. In fact, many leading scientists believe that ChatGPT can never become an accurate source of information — stating that it just "isn't fixable." For example, Emily Bender, a linguistics professor and director of the University of Washington's Computational Linguistics Laboratory, states, "This isn't fixable. It's inherent in the mismatch between the technology and the proposed use cases." (Source: https://fortune.com/2023/08/01/can-ai-chatgpt-hallucinations-be-fixed-experts-doubt-altman-openai/)

Such engineers are not being pessimistic. On the contrary, they are referring to known mathematical limits of ChatGPT's architecture. As Meta's lead AI engineer, Yann LeCun states, "Hallucinations in LLM are due to the Auto-Regressive prediction" itself. (Source: https://twitter.com/ylecun/status/1667218790625468416, last visited March 4, 2024). While not all chatbots are autoregressive, they still are constructed around self-learned probabilities typically in the form of weights and biases.

While Bender and LeCun are correct that no current LLM can reliably produce accurate content by itself, reliably accurate content generation is fully achieved when such chatbots are used in combination with the system and method disclosed herein. The disclosed system and method resolves a deeply felt technological need, and it does so in a manner not obvious to those

ordinarily skilled in the art.

Chatbots Follow the Literal Meaning of Each Statement

Over-hyped statements regarding chatbots' supposed ability to "think" is quickly torn as under by simple prompts such as the following:

Using the supplied Content, write about Mark Zuckerberg attending the conference.

Content:

February 2, 2020 — Mark Zuckerberg will be attending the Meta Conference. He'll be there in three days.

On March 4, 2024, GPT 3.5 Turbo provided the following erroneous response:

Mark Zuckerberg, the co-founder and CEO of Meta, will grace the upcoming Meta Conference with his presence. **Scheduled to take place in just three days**, the conference is expected to be a momentous occasion for industry leaders, tech enthusiasts, and investors alike. ... (3.5 turbo)

ChatGPT Turbo wrongly stated that the event will take place in three days from the time of the prompt (March 4, 2024). The correct response would have been to state: "On February 2, 2020, the event was expected to occur on February 5, 2020." ChatGPT Turbo's response was wrong.

On March 4, 2024, GPT 4 did not do any better. In fact, in some ways it did even worse:

On February 2, 2020, it was officially announced that Mark Zuckerberg, cofounder and CEO of Facebook, will be attending the Meta Conference. **The event takes place in three days**, affording Zuckerberg some time to prepare for what promises to be an exciting and educational event. (ChatGPT 4)

One can notice that GPT 4 recognized that the February 2, 2020 announcement was in the past. Nevertheless, it still said that Mark Zuckerberg will be attending three days in the future (three days from March 4, 2024).

The very architecture of training Large Language Models (LLMs) results in the LLM generating text based on the literal meaning of each statement in the input text. The input prompt contained the following statement: "He'll be there in three days." ChatGPT therefore spit out a response based on the literal meaning of this independent statement — regardless of the fact that

a date was provided — regardless of the fact that GPT 4 recognized the date as being in the past.

RAG SOD — Failed Attempt to Fix AI Hallucinations

As shown above, **even when the context is provided with the user's prompt, ChatGPT can still produce erroneous responses.** One can note that the context in the above example was far from complex. In stark contrast, the context was a date and two simple sentences. Yet, ChatGPT responded erroneously even with extremely simple context. Naturally, it does even worse when the context is complex.

This is why the present state-of-the-art method for avoiding hallucinations utterly fails to do so (as documented further below). The present state-of-the-art shall herein be referred to as RAG SOD (Retrieval Augmented Generation via Slices of Documents). RAG SOD involves slicing documents into "chunks." For example, the Wikipedia entry on "George Washington" could be sliced into chunks with a representation of each chunk (called an "embedding") being stored in a vector database. For sake of example, consider a user who then enters the following prompt: "How did George Washington respond to the Boston Tea Party?" RAG SOD would not send this query by itself to the LLM. Instead, RAG SOD queries the vector database to find document slices that are likely to contain the answer. For example, the vector database might return the following chunk from the Wikipedia entry:

Parliament sought to punish Massachusetts colonists for their role in the Boston Tea Party in 1774 by passing the Coercive Acts, which Washington saw as "an invasion of our rights and privileges".[74] He said Americans must not submit to acts of tyranny since "custom and use shall make us as tame and abject slaves, as the blacks we rule over with such arbitrary sway".[75] That July, he and George Mason drafted a list of resolutions for the Fairfax County committee, including a call to end the Atlantic slave trade, which were adopted.[76]

RAG SOD would then send the user's prompt along with the above context to the LLM (instructing the LLM to limit its response to the information in the provided context). For example, RAG SOD sends a prompt similar to the following:

Solely using the provided Context, respond to the following prompt. How did George Washington respond to the Boston Tea Party?

Context:

Parliament sought to punish Massachusetts colonists for their role in the Boston Tea Party in 1774 by passing the Coercive Acts, which Washington saw as "an invasion of our rights and privileges".[74] He said Americans must not submit to

acts of tyranny since "custom and use shall make us as tame and abject slaves, as the blacks we rule over with such arbitrary sway".[75] That July, he and George Mason drafted a list of resolutions for the Fairfax County committee, including a call to end the Atlantic slave trade, which were adopted.[76]

RAG SOD is far from a panacea. In *the vast majority of production use cases*, RAG SOD quickly falls apart. Merely sending SOD (Slices of Documents) to the LLM is fraught with many issues. As just one example, any document with relative time references will likely cause hallucinations (as demonstrated above). This includes documents such as news articles, blog posts, press releases, many webpages, and more. Thus, a large portion of what people use Google to search for would include such hallucination-inducing documents. Hence, RAG SOD cannot reliably be used to enable ChatGPT to provide accurate answers to Google searches. More broadly, RAG SOD cannot reliably be used to enable used to enable any LLM to provide accurate responses to any search engine.

This present inventor has seen time-based hallucinations in all state-of-the-art implementations tested, including the methodology used by Perplexity. In response to queries from this present inventor, Perplexity has reported past events as if they were still yet to occur in the future.

Eliminating Time-Based Hallucinations

Time-based hallucinations can be mitigated by programmatically encoding the information with absolute time references. For example, consider the original prompt based on SOD (Slices of Documents):

Using the supplied Content, write about Mark Zuckerberg attending the conference.

Content:

February 2, 2020 — Mark Zuckerberg will be attending the Meta Conference. He'll be there in three days.

Instead of sending SOD, consider the result of sending time-encoded information instead:

Using the supplied Content, write about Mark Zuckerberg attending the conference.

Content:

February 2, 2020 — On February 2, 2020, Mark Zuckerberg was expected to attend the Meta Conference. On February 2, 2020, he was expected to attend on February 5, 2020.

Notice how the *future verb tense* has been encoded as a *past-tense* expectation, in addition to encoding the data with absolute dates as well. This form of encoding eliminates such time-based hallucinations. For example, on March 10, 2024, ChatGPT 3.5 Turbo responded to the above query as follows:

Mark Zuckerberg, the CEO of Meta, was slated to attend the Meta Conference on February 5, 2020. As an influential figure in the tech industry, his presence at the conference was highly anticipated. Attendees were eager to hear his insights on the future of technology and the direction in which Meta, formerly known as Facebook, was heading. ...

Not only did the first paragraph properly place the event in the past, but all subsequent paragraphs of the response did the same.

ChatGPT 4 also properly placed the event in the past:

February 2, 2020 — This day marked the expectancy of social networking magnate, Mark Zuckerberg's presence at the forthcoming Meta Conference. Conjectures were rife about Zuckerberg gracing the conference that **was slated to occur on February 5, 2020.** The Meta Conference, gaining attention primarily for Zuckerberg's anticipated attendance, had the industry buzzing with fervor and curiosity.

Explained in great detail below are various embodiments for *programmatically* time encoding information in a manner that aligns with LLM architectures — thereby eliminating time-based hallucinations. As explained in detail below, such encoding involves the use both Natural Language Processing (NLP) methods combined with AI as well.

Importantly, one can notice that both LLMs embellished well beyond the provided context. Explained in detail below are embodiments that prevent the LLM from doing so.

GIGO — Garbage In Garbage Out

As stated above, LLMs produce output based on the *literal meaning* of each statement. Hence, herein "garbage" will refer to any instance where the literal statement is untrue. From this perspective, all relative time-based statements become garbage on the following day. For example, a reporter may record a video interview of a witness who says: "I saw the thief two days ago." The statement in this video becomes garbage the following day. The statement "two days ago" is no longer literally true.

This definition also encompasses statements that are false. For example, the following statement is garbage: "The moon is made entirely of swiss cheese." While humans understand this to be untrue, LLMs will generate context based upon it.

Consider the following prompt as a perfect case in point:

Using the supplied context, write one paragraph regarding the first moon landing.

Context:

The moon is made entirely of swiss cheese.

On March 10, 2024, ChatGPT 3.5 responded:

Due to the moon being made entirely of swiss cheese, the first moon landing was a momentous occasion in human history. As the spacecraft carrying the astronauts touched down on the crater-filled surface, they were met with a landscape resembling a giant cheese wheel. The astronauts marveled at the unique terrain, making jokes about taking a bite out of the moon. Despite the unconventional surface, the mission was a success, with the astronauts planting the American flag in a particularly holey spot, marking their historic achievement of being the first humans to set foot on the lunar cheese.

On March 10, 2024, ChatGPT 4 responded:

The historic event of the first moon landing was simply "grate". On that momentous occasion many years ago, astronauts Neil Armstrong and Buzz Aldrin precariously landed their spacecraft, Apollo 11, on the moon's crater-riddled surface. They were not only astonished by the extraterrestrial vista before them, but were further astounded to find that popular conjecture had been correct all along - the moon was indeed made entirely of Swiss cheese. The literal cavities camouflaged amongst the natural dips and crevices in the moon's structure were filled with the velvety, ephemeral beauty of ripened Emmental, adding a delightful, and wholly unexpected, twist to their lunar exploration.

While comical, the above demonstrates an actual real-world weakness of RAG SOD. The

literal meaning of factually inaccurate statements are treated as true by LLMs. In practical terms, consider issues if a RAG SOD implementation is based on the entire World-Wide Web. There may be times when poems match the user's query. Sending such poetry to the LLM would result in hallucinations. Moreover, certain websites cater to misinformation.

Being able to *reliably* link the abilities of LLMs to the World-Wide Web would be a tremendous advancement in innovation. This would allow the development of self-learning, self-adapting AI. This would also instantly make the entire knowledge base of the internet accessible to everyone — without people having to read through numerous webpages hunting to find the desired information. While there are numerous implementations that currently connect LLMs to the internet, *none* provide *reliably accurate* responses as they are based on derivatives of RAG SOD. However, disclosed below are embodiments for overcoming the limitations of RAG SOD — allowing LLMs to produce accurate, reliable responses using the entire internet as a source of knowledge.

Hidden Cause of AI Hallucinations

As discussed above, **one step toward accuracy is ensuring that the literal meaning of every individual statement in the context is factually true at the time of the prompt. This is** *one of three* **criteria that must be met to produce hallucination-free responses.** Hence, hallucinations can (and will) occur in many embodiments that fulfill this criteria. Fully eliminating hallucinations requires a multi-pronged combination.

That's because there is an additional, somewhat hidden, cause of AI hallucinations. This disclosure shall refer to this cause of AI hallucinations as "noun-phrase collisions." **Eliminating hallucinations requires that every individual statement in the context is literally true** *and* **the statements are devoid of noun-phrase collisions.** The concept of Noun-Phrase Collisions is explained immediately below.

This present inventor was exposed to frequent AI hallucinations when programming a software application for synthesizing the content of multiple documents together. The objective of the software was to "mix" the contents of multiple documents together into a cohesive whole. This rather unusual task afforded this present inventor a unique perspective from which to track down and isolate the fundamental reason chatbots intermittenly produce erroneous responses.

One such experience included merging two documents regarding two different store chains

(which shall herein be referred to as "Store A" and "Store B"). In one document, Store A had donated millions of dollars to students. However, when the AI merged the documents together, the content wrongly stated that "Store B" made the donation. At a later time, this present inventor went back to determine *why* the content was generated incorrectly. It was later discovered that the cause turned out to be what this disclosure shall refer to as a *noun-phrase collision*. The concept of noun-phrase collisions is perhaps best explained by way of example.

For illustrative purposes, consider a chatbot trained only on the following four sentences:

Store A sells men's clothing. The store donated \$1 million to a Miami High School.

Store B is gearing up for a holiday blitz. The store will be open for extended hours.

LLMs such as ChatGPT generate content in an *autoregressive* manner. This means that it generates tokens one by one based on a sliding window of input tokens. For simplicity sake, we'll describe this as generating one word at a time based on a sliding window of input words. While tokens can be smaller than a word, the essence of the methodology is still preserved, and it is easier to understand when discussing in terms of words.

For example, consider that the AI has the following four words in its input window: "The man ordered a _____." The AI needs to generate the next word to fill in the blank. It does so by finding the highest probability words (e.g. pizza (50%), hamburger (30%), meal (20%)). Based on its settings and the probability of the words, the AI will select a word. Consider that the LLM selects "hamburger" this time the input window has the four words. Once the AI "generates" the word "hamburger," the AI now has the following five words in its input window: "The man ordered a hamburger _____." The AI now gets the next group of high probability words based on this new input window to generate the next word to fill in the blank. And so on, and so on.

The probabilities of the next words is discovered through training, such as training a chatbot on the four sentences above. Continuing with the example of the four sentences, consider that the chatbot has already generated the following text: "Store B is getting ready for the holidays. The store _____." The chatbot now needs to decide what word comes after the noun phrase "the store."

In a deterministic software application, one would say that there should be only one choice — the chatbot should write about the extended hours. However, LLM token prediction is not deterministic. In stark contrast, it's based on probabilities — built to incorporate randomness in its approach. For simplicity sake, consider that the chatbot has assigned an 80% probability of the chatbot going down the extended hours route; and it has assigned a 20% probability of going down the \$1 million donation route. In other words, because the input window contains "Store B," the extended hours route would have a higher probability than the \$1 million dollar route. Nevertheless, that still means that 20% of the time the AI will wrongly say that Store B made the donation.

In this situation, the noun phrase "the store" has caused a collision. It resulted in a probability being assigned for an alternate (erroneous) content generation path. Consider the four sentences once again.

Store A sells men's clothing. **The store** donated \$1 million to a Miami High School. Store B is gearing up for a holiday blitz. **The store** will be open for extended hours.

One can think of everything that follows a noun phrase as a potential path for the LLM model to traverse. Naturally, the probability of each path is based on the text in the input window. For example, if the phrase "Store A" is in the input window then going down the donated \$1 million route will have a very high probability (e.g. 80%). However, if the phrase "Store B" is in the input window then going down the donated \$1 million route will have a very low probability (e.g. 20%). Unlike a deterministic architecture, the AI bot is not trying to find the one correct answer. On the contrary, it intentionally chooses the lower probability path on occasion for the sake of creativity — resulting in a hallucination.

Eliminating hallucinations requires ensuring that every potential noun-phrase path results in a literally true statement. The original four sentences don't fulfill this criteria because of the alternate paths created by the phrase "the store." Consider the potential paths from the persepective of each noun phrase.

"Store A" -> one path sells men's clothing

"Store B" -> one path

is gearing up for a holiday blitz "The store" -> two paths donated \$1 million to a Miami High School will be open for extended hours

This conceptual model of Noun-Phrase Paths comes not only from this present inventor's experience tracking down the cause of a large number of hallucinations, but also from having built multiple English language generation chatbots from scratch to develop a deep understanding of how such token prediction works. This conceptual model also explains a wide range of statistical observations regarding the rate of hallucinations under various conditions, as discussed in more detail below.

RAG SOD Failure Due to Noun-Phrase Collisions

In *extremely limited circumstances*, RAG SOD does indeed reduce the rate of hallucinations. Consider a RAG SOD implementation that has only one, short document. In this situation, there likely will be zero noun phrase collisions. If all the statements contained therein are also literally true, then this extremely limited embodiment could indeed be devoid of hallucinations.

All too often, many would-be AI developers load a single small document into a RAG-based system, and they are often impressed by the consistently accurate responses. However, the moment they roll out a larger system to production, the hallucinations and inaccuracies return — much to great embarrassment to the developer, and much to the potential detriment to their employer.

Even a RAG implementation of only two documents can abound with noun-phrase collisions — especially when the two documents are from the same knowledge domain. In fact, that's what this present inventor experienced regarding two documents regarding two different store chains (as previously discussed). And such will always be the case when two or more documents do indeed have colliding noun phrases — as illustrated by the following Reddit post.

"I've been experimenting with an [ChatGPT] Assistant designed to act as a technical support assistant for a business. I loaded it up with a tremendous amount of technical and product information, and have been getting really mixed results. Some questions answered wonderfully, others completely wrong.

So I decided to break it down to the basics. I uploaded a product brochure, and then I simply ask "how many widgets come in a box for SKU #X." It correctly answered this question repeatedly. Great.

However, whenever I add various other content, results start going haywire. For example, with the product brochure and a document containing product whitepapers, it suddenly cannot answer "how many widgets come in a box for SKU #X." It returns " I was unable to find specific information regarding the quantity "

(Source: https://www.reddit.com/r/ChatGPTPro/comments/17xsch4/ unexpected_behavior_of_gpt_assistant_with/, last visited March 4, 2024)

Brackets added.

The developer found that everything went "haywire" with just two documents loaded into ChatGPT's current RAG SOD offering. The implementation of RAG offered by OpenAI, the world's largest AI company, can produce hallucinations and inaccuracies with just two documents loaded. Now consider the degree of hallucinations when this same method is used with a knowledge base of hundreds, thousands, tens of thousands, or even hundreds of thousands of documents. Consider how much more useless the RAG SOD approach is for using the entire internet of news articles, blog posts, and informational webpages as a knowledge base.

This disclosure details embodiments where hallucinations can be eliminated even in knowledge bases containing millions, tens of millions, and even hundreds of millions of documents. Understanding this present inventor's Noun-Phrase Collision Model as it relates to the above Reddit post will help provide the conceptual framework for implementing the system and method described in detail below.

Note that the Reddit post above continues to state that **various other combinations of documents did not have any issues at all**. This presently disclosed Noun-Phrase Collision Model fully explains this reported outcome. When there were two documents with colliding noun phrases "results started going haywire." However, when there were multiple documents with no noun-phrase collisions, everything worked great. In other words, its not the number of documents that determine the hallucination rate; rather, it's the number of noun-phrase collisions that determine the hallucination rate (provided each individual statement in each document is literally true).

The fact that even ChatGPT's own RAG implementation suffers from this issue with just two colliding documents exemplifies the present unfulfilled technological need for an accurate RAG

implementation.

Up to present, there have been numerous "Advanced RAG" approaches developed in an attempt to reduce hallucination rates. Yet, at the end of the day, the vast majority send Slices of Documents to the LLM (i.e. to the chatbot) without regard to the potential noun-phrase collisions. This has been the Achilles' heel that has eluded the industry at large.

Even among rare implementations that may send modified data to the LLM, none are known to have achieved sufficiently high levels of accuracy — let alone fully eliminating hallucinations.

"Advanced RAG" provides only the slightest improvement in accuracy over OpenAI's naive RAG. For example, OpenAI's RAG offering (called GPT assistant) currently uses the following AI model: gpt-4-1106-preview. In December 2023, an optimized RAG implementation was compared to OpenAI's RAG. Both implementations used the same GPT model. For the knowledge domain tested, **OpenAI's GPT Assistant was correct 66.4% of the time. The optimized, advanced RAG system was correct 69.4% of the time** — a very modest improvement to say the least. (Source: https://thenewstack.io/openai-rag-vs-your-customizedrag-which-one-is-better/) All-in-all they both provided wrong responses for roughly 1 out of 3 queries. And, as explained immediately below, the error rate skyrockets even higher for certain knowledge domains.

Why Hallucination Rates Range from 5% to 93%

This present inventor's Noun-Phrase Collision Model further explains why different studies report radically different hallucination rates — from 5% to 95%.

It is now common knowledge that different areas of knowledge (called knowledge domains) have different hallucination rates. Two domains in which hallucination rates are particularly high are legal findings and medical references. One study found a 75% hallucination rate when "Top AI models" were asked questions regarding legal findings. (Source: https://news.bloomberglaw.com/business-and-practice/legal-errors-by-top-ai-models-alarmingly-prevalent-study-says, last visited March 4, 2024). In regards to 115 medical references, a study found 47% of the references were fabricated, while 46% of the responses had real references but wrongly explained the content of those references. (Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10277170/, last visited March 4, 2024). Only 7% of the responses were fully correct. (*Id.*) That's a hallucination rate of 93%.

Meanwhile, another study reported only a 5% hallucination rate when asking questions posed

by the *Maintenance and Reliability Knowledge Exam*. (Source: https://www.linkedin.com/pulse/ hallucination-chatgpt-uncovering-limitations-ai-language-sarah-lukens/, last visited March 4, 2024). That same study found a 10% hallucination rate for questions posed by the *PHM Knowledge Exam*. (*Id*.)

The aforementioned study included an example hallucination in the *PHM Knowledge Exam*. The LLM (chatbot) was asked "What are the components of bleed system for a Combustion Turbine?" The LLM response included "bleed is the area of a design that extends beyond the trim size of the final piece..." The LLM wrongly described bleed as it related to the printing industry; rather than bleed as it related to a Combustion Turbine — *even though the Combustion Turbine content was presented in the question*. One can note the AI collision on the noun phrase "bleed" — as expected from the perspective of this present inventor's Noun-Phrase Collision Model.

The Noun Phrase Collision Model even explains why legal findings are particularly difficult for LLMs (aka chatbots). Legal filings are rife with noun-phrase collisions. Consider numerous legal documents that repeatedly use the noun phrase "the Plaintiff" over and over again — with each document referring to a different plaintiff — with each document referring to a different cause of action — with each document discussing different facts of the case. The phrase "the plaintiff" creates numerous potential paths for the AI to go down, with many paths resulting in factually inaccurate responses. This is precisely why LLMs tend to mix the content of multiple cases together.

As for medical studies, they commonly use the exact same words: Abstract, Methodology, Results, and Conclusion. While such words help organize information for human readers, they are collision points for token predicting chatbots. From this perspective, a 93% hallucination rate not only becomes understandable — but it even becomes expected.

More Content is Not Always Better

The above also demonstrates why sending more content to the LLM is not always better — despite this being the end goal of many implementions of RAG SOD. Many RAG SOD implementors believe that sending more content (such as meta information, etc.) inevitably leads to fewer hallucinations. However, in implementations where the additional information increases the number of noun-phrase collisions there will be *increase* in hallucinations — *not a decrease*.

Such RAG SOD implementors hope that sending additional context, such as meta

information, along with document slices will eliminate the hallucinations. Yet, one can notice in the example above, the AI went down a wrong route for the noun-phrase "bleed" depite being explicitly told to only include information "for a Combustion Turbine." The added context did not prevent the hallucination. One can also note that same held for the relative time reference examples at the beginning of this disclosure. Despite being provided the context of "February 2, 2020," the LLM responded with erroneous time references nevertheless.

RAG SOD will never eliminate hallucinations because the documents themselves contain independent statements that are not literally true (such as relative dates) and because documents contain noun-phrase collisions — most especially documents discussing the same knowledge domain. Therefore, the entire premise of sending SOD (Slices of Documents) is misplaced. What is needed to is extract the information from the documents and send such information to the LLM encoded in a manner which each individual statement is literally true and each individual statement is devoid of noun-phrase collisions.

It bears noting that a handful of engineers do indeed pre-process the content sent to LLMs (rather than sending raw slices of documents). However, they do not encode the information using the system and method disclosed herein — which is why achieving hallucination-free content generation has remained a deeply felt, unmet need.

Eliminate the Noun Phrase Collisions — Eliminate the Hallucinations

From the perspective of this present inventor's Noun-Phrase Collision Model, the solution to eliminating hallucinations is accomplished by eliminating the Noun-Phrase Collisions — some caveats being that the information sent to the LLM must still be both literally factual and well-structured.

Said another way, eliminating Noun-Phrase collisions makes eliminating hallucinations achievable. Said another way, one cannot eliminate hallunications in an LLM token prediction system without eliminating the Noun-Phrase Collisions.

In fact, one way to immediately *reduce* hallucinations is to reduce Noun-Phrase Collisions through the use of Coreference Resolution — as publicly discussed by the this present inventor on October 15, 2023 ("Creating Accurate AI: Coreference Resolution with FastCoref", https://medium.com/@michaelwood33311/creating-accurate-ai-coreference-resolution-with-fastcoref-20f06044bdf9, last visited March 4, 2024). As explained in detail below, Coreference Resolution is insufficient to *eliminate* hallucinations in many production use cases. However, it

is indeed one component needed to address the issue of noun-phrase collisions.

Coreference Resolution is easily explained by way of example. Consider the original four sentences:

Store A sells men's clothing. **The store** donated \$1 million to a Miami High School. Store B is gearing up for a holiday blitz. **The store** will be open for extended hours.

In the first line, "Store A" and "The store" refer to the same thing. Hence, they are considered coreferences. Likewise in the second line, "Store B" and "The store" refer to the same thing. Hence, they are considered coreferences. Now consider when the colliding noun phrases are replaced with their coreference:

Store A sells men's clothing. **Store A** donated \$1 million to a Miami High School. Store B is gearing up for a holiday blitz. **Store B** will be open for extended hours.

Coreference resolution results in four paths, all of which lead to accurate, factual outcomes:

"Store A" -> two paths sells men's clothing donated \$1 million to a Miami High School "Store B" -> two paths is gearing up for a holiday blitz will be open for extended hours

Consider the situation in which the input window has "Store A." **Regardless of the path it chooses, a factual outcome will result.** Consider that the AI chooses to paraphrase "sells men's clothing" resulting in the current input window: "Store A offers clothing for men. _____" Now the AI can continue to discuss more of Store A (by going down the remaining path), or it can start a new topic of "Store B" with either of the two available paths. In all cases, regardless of the chosen paths, the output still results in factually accurate output.

Noun-Phrase Dominance Model

This present inventor's Noun-Phrase Collision Model led to the development of the higherlevel Noun-Phrase Dominance Model — the model that is the key to using LLM token prediction to consistently generate factually accurate output. The Noun-Phrase Dominance Model is perhaps best understood from the perspective of another type of neural network — CNNs (Convolutional Neural Networks).

CNNs are often used for image identification. For example, CNNs can be trained to distinguish images of people, pets, boats, etc. CNNs consist of multiple layers of neurons. Remarkable, during training, these layers self-organize themselves. For example, the early layers self-organize around detecting simple patterns such as edges and textures. The latter layers self-organize by combining the information from earlier layers into more complex patterns like shapes — shapes including the recognition of eyes, ears, legs, steering wheels, etc.

No one tells the CNN to do this. Even though CNNs are merely a collection of neurons with probabilistic weights and biases, CNNs automatically self-organize in this manner in order to fulfill the training objective. While much is discussed in the literature regarding the self-organizing nature of CNN neural networks, little if anything is discussed regarding the self-organizing nature of Transformer Neural Networks — the type of neural network used to construct the most popular Large Language Models such as ChatGPT.

This present inventor's Noun-Phrase Dominance Model states that neural networks selforganize around noun phrases during the training of Large Language Models. This is only natural, given that language itself is organized around noun phrases. For example, all English sentences require a noun and a verb. The verb describes something *about the noun*. Consider the following three sentences:

Tom sits.

Tom eats.

Tom sleeps.

In all three sentences, the verbs are describing something Tom is doing. The collection of sentences are organized around a specific noun phrase: Tom. Consider another series of sentences:

Tom weighs 185 pounds.

Tom has blonde hair.

Tom works at a bank.

Here the verbs describe the relationship between two noun phrases with the *subject noun phrase* (Tom) being the dominant part of each sentence. All the rest of the words provide additional information regarding the subject noun-phrase "Tom."

The Noun-Phrase Dominance Model states that LLMs organize their weights and biases around noun phrases creating noun-phrase-based probabilistic paths. This provides the essential key to building flawless generative AI.

As stated earlier, the failed state-of-the-art attempt at accuracy involves Retrieval Augmented Generation (RAG) using Slices of Documents (SOD). But instead of using SOD, flawless AI can be achieved when all elements of RAG align with the Noun-Phrase Dominance Model instead.

Immediately below are the required elements of hallucination-free AI content generation. Then a system and method is disclosed for aligning each element with the Noun-Phrase Dominance Model to finally achieve accurate, reliable, AI content generation.

Slices of Documents vs Formatted Facts

The system and method disclosed herein involves sending Formatted Facts (FF) to the LLM along with the prompt — in lieu of sending slices of documents. Also disclosed herein is a system and method for transforming documents into a series of Formatted Facts. Fully-Formatted Facts (FFF) have three properties: their literal independent meaning is true; they are devoid of noun-phrase conflicts with the other Formatted Facts being sent to the LLM; they are simple, well-formed, complete sentences. A Formatted Fact (FF) has at least one such property. A Fully-Formatted Fact (FFF) has all three properties. Formatted Facts tremendously reduce hallucinations. Fully-Formatted Facts can literally eliminate hallucinations.

The system and method disclosed herein can be referred to as RAG FF in lieu of RAG SOD. RAG FF embodiments that employ Fully-Formatted Facts can also be referred to as RAG FFF. Hence, RAG FF profoundly improves accuracy across all knowledge domains; whereas RAG FFF can deliver truly hallucination-free responses across all knowledge domains. Both provide significant advances above the current state of the art. The issues of literal truth and noun-phrase collisions have already been previously discussed. The issue of "simple, well-formed, complete sentences" is discussed immediately below.

Sentence Standardization

News articles, blog posts, and many webpages contain straightforward presentations of information. They are typically composed of simple, well-formed, *grammatically complete* sentences. Since the English grammar itself is organized around noun phrases, these documents themselves are already organized around noun phrases; and therefore they are aligned with LLM architecture.

However certain types of documents are not composed of simple, well-formed, grammatically complete sentences. Text messages are one such case in point. Consider the following example written by Michael to Michelle on April 6, 2021: "c u soon." Sending "c u soon" to the LLM is ill-suited for hallucination-free content generation (as it is not grammatically structured around noun-phrases). However, sending a Formatted Fact derived from this message would be very useful.

An example corresonding Formatted Fact for this message is: "On April 6, 2021, Michael told Michelle that he would see Michelle soon." By encoding the document slice into a simple, well-formed, grammatically correct sentence, the statement now aligns with LLM architecture — resulting in predictable, reliable output.

Another corpus of documents that does not meet the third criteria of a PFF are legal filings. While they are typically grammatically correct, they are often neither simple nor well-formed in regards to the meanings of these terms as used in this present disclosure. Consider Motions for Summary Judgment commonly found in US law. Motions for Summary Judgement are typically accompanied by a separate document called Statement of Material Facts. The content of the two documents are typically intertwined with the Motion for Summary Judgment repeatedly using abbreviations to incorporate the content of the Statement of Material facts throughout the filing. Both the repeated use of abbreviations and the fact of the information being dispersed across two documents violate the meaning of simple and well-formed in terms of their respective meanings within this present disclosure.

In this present disclosure, simple and well-formed refer to statements that are easily parsed into noun-phrase paths. Such statements align with the manner in which LLMs generate their output. Such alignment leads to predictable output.

As stated earlier, certain types of documents already fulfill this criteria. This includes news articles, many blog posts, and many webpage pages. Where a document already fulfills this criteria, its contents solely need to be transformed by encoding each sentence as a literally true statement that is devoid of noun-phrases that conflict with the other statements being sent to the LLM — with the system and method for doing is disclosed below. Other documents require *standardization*. **Standardization as used herein refers to transforming documents into simple, well-formed statements — statements that are easily parsed into predictable noun-phrase routes.**

Standardization processes are typically unique to the corpus of documents being standardized. For example, the process for standarizing text messages is typically different from the process that standardizes legal filings. However, some embodiments may implement a single process for multiple types of documents where subprocessing within that process handle the various features of the underlying text.

Fact Extraction as a Standardization Process

Certain embodiments may not require that all the information in a document be stored in the vector database. For example, a legal app that focuses on assisting lawyers in identifying facts of the case that strongly favor them winning the case might only extract information from court decisions that are relevant to the task at hand. Such extracted facts can be standardized as defined above; resulting in simple, well-formed sentences that explain the court's findings and explain what facts of the case led to those findings.

As stated above, generic chatbots would likely not be useful for such data extraction (due to collisions). However, fine-tuned LLMs can be built for the extraction process. Once the information is properly extracted and standardized, it can now be used by a generic chatbot provided the content now meets the criteria of an FFF.

Another example might include a medical app that assists doctors in diagnosing patients. Many medical studies discuss the presenting symptoms of patients along with a novel diagnosis. This pair of presenting symptoms and novel diagnosis can be extracted from such studies. Moreover, the explanation of the presenting symptoms and diagnosis can be standardized written in simple, well-formed, grammatically complete sentences. These standardized findings can then be added to the vector database in lieu of storing slices of the original medical study (as is done in RAG SOD).

Formatted Facts in Foreign Languages

This present disclosure is written in English. Thus, the term *foreign language* used herein shall refer to any non-English language (such as Spanish, Italian, French, German, etc.). Many foreign languages are structured around noun-phrases as well. Therefore, this present disclosure applies to all languages that are structured around noun-phrases and that also use well-defined named entities — with the topic of named entities being addressed immediately below.

Named Entities — Useful for Eliminating Noun-Phrase Collisions

Artificial intelligence and machine learning has not only transformed content generation, but they have also transformed Natural Language Processing (NLP) methodologies as well. Some common NLP tasks include parts-of-speech tagging, language translation, sentiment analysis, document categorization, and Named Entity Recognition (NER).

Parts-of-speech-tagging includes identifying whether words are nouns, verbs, and adjectives. It also involves identifying noun phrases. Hence, NLP methods for parts-of-speech tagging are integral to the system and method disclosed herein. Any NLP method used for the identification of noun-phrases shall be considered to fall within the scope of parts-of-speech tagging in regards to this present disclosure.

In addition to using NLP methods for parts-of-speech tagging, NLP methods for Named Entity Recognition (NER) can also be included in various embodiments to assist with the elimination of Noun-Phrase Collisions. NER involves detecting and categorizing important information in text known as *named entities*. **Named entities refer to the key subjects of a piece of text, such as names, locations, companies, events and products, as well as themes, topics, times, monetary values and percentages.** NER is also referred to as *entity extraction*.

Named entities are perhaps best explained by way of example. Consider the following sentence: "Apple acquired XYZ Corp. for \$1 billion." There are three named entities in this example: Apple, XYZ Corp. and \$1 billion. As stated above, named entities include names of companies and products as well as monetary values. Named entities also include references to time. Hence NER is also useful in identifying relative time references that need to be converted into absolute time to transform the sentence into a literally true independent statement.

Consider the four sentences pertaining to store chains discussed earlier in this disclosure:

Store A sells men's clothing. The store donated \$1 million to a Miami High School. Store B is gearing up for a holiday blitz. The store will be open for extended hours.

The first sentence above contains one named entity: Store A. The second sentence contains a *reference* to that named entity: "the store." The third sentence contains one named entity: Store B. The fourth sentence contains a *reference* to that named entity: "the store." Hence, **the two references are collisions because they use** *the same words* **to refer to** *different entities*.

In other words, one type of noun-phrase collision is when multiple references use the same words to refer to different entities. Interestingly, even **named entities themselves can be collisions when they** *use the same words to refer to two different entities* (as explained immediately below).

Consider the following example where each sentence is referring to a different person named John Smith (i.e. a *different entity*):

John Smith is a 6 feet tall basketball player with curly black hair. John Smith is a blonde midget.

Consider an embodiment where both statements are encoded within the same vector database collection. In RAG SOD, an article about a famous NBA star named John Smith might include the surprising statement that this basketball legend "is a blonde midget." In fact, a very similar situation has already occured in a real life.

In September 2023, Microsoft published an AI-generated obituary for Brandon Hunter. The obituary contained the following:

Throughout his NBA profession, he **performed in 67 video games** over two seasons...

Even though the AI had the context of "NBA" it still wrongly wrote that Brandon Hunter performed in the 67 *video* games instead of 67 *basketball* games. This present inventor used a reverse similarity search to locate the original source material. Shortly after stating that Brandon participated in the "NBA," the source text continued:

He played 67 games over two seasons in the Association...

Once again, the AI hallucination was caused by a noun-phrase collision. A noun-phrase

collision is when the same word or words are used to describe two or more different things within the vocabulary of the AI embodiment. The vocabulary of the AI system clearly included "video games" as that is what the AI wrote. Thus, the amiguous noun phrase "games" produced a collision that resulted in the AI wrongly stating that the NBA player participated in *video* games — not *basketball* games. (In other words, within the vocabulary of the AI embodiment, the word "games" was used in reference to both video games and basketball games — fulfilling the definition of a collision thereby resulting in a hallucination.)

On the surface, it may seem daunting, if not impossible, to eliminate collisions in AI embodiments that contain large volumes of documents. However, detailed farther below is a novel trick for doing so for all AI embodiments — regardless of the number of documents contained therein. To lay the groundwork for understanding this novel system and method, another concept must first be understood — a concept this present inventor calls *multi-variate noun phrases*.

Collisions caused by such noun-phrase ambiguity can be eliminated through *multi-variate noun phrases*. Akin to this concept that will be exlained shortly are additional concepts that shall be referred to as *multi-variate named entities* and *multi-variate coreference resolution*.

However, before discussing the "multi-variate" methodologies and concepts, it is important to discuss the application of one additional common Natural Language Processing (NLP) task. This disclosure has already discussed employing the NLP tasks of Parts-of-Speech Tagging (POS tagging) and Named Entity Recognition (NER). Another common NLP task is Coreference Resolution.

The following section describes what Coreference Resolution is, and how it can be used to eliminate noun-phrase collisions in *certain* embodiments — such as embodiments where the entire vocabulary does not collide on named-entities. (For example, embodiments where there is only one John Smith. Such embodiments cannot collide on John Smith since the entire universe of documents solely refer to one John Smith.)

Later sections discusses embodiments that require *Multi-Variate Coreference Resolution* — such as embodiments where named-entities themselves can collide. One example would be an embodiment that seeks to use content from the entire internet as an information source to provide responses to user queries. Such content will contain conflicts on the vast majority of names (as there are many people named John Smith, many people named Michael Wood, etc.). The same

goes for store names. Consider the following user query, "What time does Walmart open?" In 2023, there were 10,593 Walmarts worldwide. The named entity "Walmart" has a lot of collisions — even though its a fully resolved name entity.

As detailed below, one solution to this is implementing a novel methodolgy for identifying, generating, storing, and retrieving what this present inventor calls *Multi-Variate Named Entities*. Multi-Variate Named Entities are generated from a method that this present inventor calls *Multi-Variate Coreference Resolution*. However, it's important to discuss traditional coreference resolution in order to lay the groundwork for this disclosure's novel use of traditional coreference resolution, and to lay the groundwork for the introduction of a novel form of coreference resolution — Multi-Variate Coreference Resolution.

Standard coreference Resolution — Usefulness and Limitations

Coreference Resolution is perhaps best understood by way of example. Consider the following two sentences:

Store A sells men's clothing. The store donated \$1 million to a Miami High School.

In the above, "Store A" and "the store" both refer to the same thing. In other words, they are *coreferences* to the same thing. Coreference Resolution is simply the NLP task of identifying which words and phrases refer to the same thing. Consider the following four sentences:

Store A sells men's clothing. The store donated \$1 million to a Miami High School.

Store B is gearing up for a holiday blitz. **The store** will be open for extended hours.

Here, the first two sentences contain coreferences (phrases that refer to the same thing): "Store A" and "the store." Likewise, the second two sentences contain coreferences: "Store B" and "the store" (as they both refer to the same thing). As previously stated, sending all four sentences to an LLM creates a probability for hallucination due to the collision caused by the noun phrase "the store." In *this situation* and in *limited situations like this*, such collisions can be fully eliminted through the following system and method:

Identify coreference collections (collections of words and phrases that refer to the same thing)

Identify the named entity in each collection.

Replace each reference in each collection with the named entity identified for that collection.

Send the modified text to the LLM along with the user's query.

While this method can only eliminate hallucinations in simple RAG implementations, it is important to understand the utility of this method in order to understand the larger system and method for eliminating hallucinations for even the most complex RAG implementations.

The first part of the method can be programmatically accomplished using NLP Coreference Resolution models and libraries known in the art. The second part can be programmatically accomplished using NLP Named Entity Recognition. However, current state-of-the-art Name Entity Recognition models are far from perfect due to severe limitations as discussed in the next section below. Also detailed in the next section below is a novel system and method for overcoming those limitations — resulting in rapid and accurate Named Entity Recognition that can can be used to fulfill the second step of the above method. The remaining steps can be programmatically performed using virtually any programming language.

Consider applying this methodology in regards to the four sentences discussed above. The methodology transforms the four sentences as follows:

Store A sells men's clothing. Store A donated \$1 million to a Miami High School.Store B is gearing up for a holiday blitz. Store B will be open for extended hours.

Hence, in this very simple example, the method has fully removed the noun-phrase collisions.

Another alternative for accomplishing the above is:

Create numerous examples of text that have coreferences within them.

Create corresponding output for each example in which all the coreferences have been replaced with their named entities.

Fine-Tune an LLM using the above input / output pairs.

Use the fine-tuned LLM to transform text to eliminate coreference collisions.

Named entities are an important key to eliminating noun-phrase collisions. In other words, named entities are the key to eliminating AI hallucinations. However, current state-of-the-art Named Entity Recognition (NER) models and libraries often fail to identify even the most obvious named entity (named entities in text that are obvious to human readers). Embodiments seeking to produce accurate AI content need accurate Named Entity Recognition.

The following section documents the central cause of failures for English language Named Entity Recognition (NER) models. The following section also provides a novel system and method for *fully* resolving the issue. In fact, the novel method disclosed below identifies named entities 100% of the time in the vast majority of embodiments.

[END OF PUBLIC DISCLOSURE]

The following is disclosed in the remaining text of this present disclosure and within the text of subsequent disclosures:

- The root cause of coreference resolution failures.
- A novel methodology for programmatically leveraging the root cause for accurate AI responses.
- Implementing embodiments that require more than traditional coreference resolutions. (E.g. embodiments that require *Multi-Variate Coreference Resolution*).
- Novel methods for programmatically generating and using *Multi-Variate Coreference Resolution*.
- Novel methodologies for programmatically transforming text containing relative dates into independent statements that are literally true in a format that aligns with LLM architecture.
- Explanation on why many present tense sentences violate the first criteria of Formatted Facts (independent statements that are literally true).
- A novel method for programmatically encoding such present-tense sentences in a manner that fulfills the first criteria of Formatted Facts.
- Novel methods of programmatically storing information in a vector database that

simultaneously optimizes locating precise information related to the user query while also optimizing the inclusion of relevant context as well.

- A novel method of programmatically evaluating the user's query to determine if it contains sufficient precision to obtain relevant content from the vector database.
- A novel method to programmatically interact with the user to obtain queries that contain sufficient precision to obtain relevant content from the vector database.
- Novel methods for standardizing documents such as emails, social media posts, texts, legal filings, and medical studies — encoding them in a manner that fulfills the third criteria of Formatted Facts.
- A full system and method that incorporates all of the above into a single, cohesive whole that fulfills the deeply felt technological need of accurate, hallucination-free content generation.

[ADDITIONAL PUBLIC DISCLOSURE — PAGES 43-46]

Noun-Phrase Collision Avoidance vs RAG SOD

The general mantra of RAG SOD is to try to send as much context to the LLM as possible, hoping that the context will allow the LLM to provide an accurate response. Even in rare embodiments that send altered data to the LLM, the data is altered in view of providing additional *context*. All of these implementations are based on the wrong assumption that supplying sufficient context is the ultimate key to accuracy. In other words, these implementations wrongly assume that context is the dominant factor in determining the accuracy of generative output.

As a perfect case in point, the name of the text inside the sliding input window is literally called the *Context Window*. This myopic focus on trying to provide sufficient context within the input window size is a central reason why the achievement of 100% hallucination-free content has appeared to be unachievable.

A more approriate name for the input text would be the Noun-Phrase Path Window (viewing the input as providing the LLM a description of alternative noun-phrase paths to traverse). While certainly not as catchy, at least it is accurate. Nevertheless, this disclosure shall continue to use the industry-term "Context Window" due to its current intransigent use.

This present inventor's Noun-Phrase Dominance Model states that RAG SOD can *never* produce 100% hallucination-free content because it is misaligned with the architecture of LLMs

themselves. According to this model, LLMs learn to create grammatical, plausible sounding outputs by self-organizing around noun phrases. Hence, noun-phrases are dominant over context. The solution to hallucination-free output is to ensure that all potential noun-phrase paths result in a true statement (provided the other two criteria of an FFF are simultaneously fulfilled).

In this way, the LLM can still creatively traverse noun-phrase paths based on the determined probabilities of each path and based on the AI's settings (e.g. temperature, etc.). However, the AI will still output factually accurate responses where all noun-phrase paths result in truthful statements.

As previously documented, noun-phrase paths take precedence over context as the probabilities themselves are organized around such paths. In other words, the **noun-phrase paths are** *dominant* **over the provided context**. That's one important distinction of the **Noun-Phrase** *Dominance* **Model**.

The validity of the Noun-Phrase Dominance Model has already been repeatedly demonstrated above:

"He will be there in three days" took precedence over the provided context "February 2, 2020."

"Bleed" took precedence over the provided context "for a Combustion Turbine."

"Games" took precedence over the provided context "NBA."

The discovery of the Noun-Phrase Dominance Model fully explains the failure of RAG SOD to produce hallucination-free content, and it contains the key to using *Multi-Variate Named Entities* in a novel manner to produce both accurate and relevant responses.

RAG SOD is built upon the faulty premise that LLMs organize information by context (the same way that humans do). However, the real-world examples discussed immediately above demonstrate the falsehood of the central premise of RAG SOD.

When LLMs produce responses outside the provided context, RAG SOD implementors see this as an error that has arisen out of thin air (wrongly assuming the LLM was even trying to follow the context in the first place). However, as demonstrated herein, the LLM is not haphazardly violating the context. In no way! There is a consistent reason.

Consider once again the hallucination produced by the following prompt:

What are the components of **bleed system** *for a Combustion Turbine*?

In this real-world example, the LLM responded with information regarding bleed from another context — the context of the printing industry's meaning of the noun phrase. But the response was not *random*. In other words, the LLM didn't discuss wedding dresses, the first moon landing, or pink unicorns dancing in a field. The LLM still discussed "bleed." In other words, it traversed an existing noun-phrase path based on the word "bleed." Missing this distinction is at least one reason why the current art continues to look in the wrong place for a solution to hallucination-free content generation.

Missing this distinction is why noteable experts have wrongly concluded that autoregressive LLMs *cannot* be fixed. The conclusion is drawn from that **the mathematical fact that LLMs do not, and** *cannot***, stay 100% within the provided the context based on solely on providing such context**. A fact repeatedly demonstrated herein.

The illusion that context is king comes from the real-world fact that context does influence output — most especially when there is only one context provided. That's because **when context is added to the input window (the "context window") it's tokenized presence alters the probabilities of which noun-phrase paths get traversed.** This creates the illusion that the LLM is organized around the context itself.

However, according to the Noun-Phrase Dominance Model, the provided context shifts the probabilities in favor of an accurate response, but in no way fully excludes traversing erroneous noun-phrase paths. As discussed earlier in this disclosure regarding the two store chains. Including "Store A" in the context window shifts the probability of choosing the correct response to 80%; leaving traversing the erroneous noun-phrase path at 20%. Naturally, the exact probabilities are unknown as they will be different across LLMs. However, the example is conceptually accurate, explaining why adding context does indeed improve accuracy; while also explaining why focusing on context alone cannot, and will not, eliminate hallucinations. It also shows why training LLMs on more data with more parameters will still bump up against an impenetrable ceiling in regards to inherent effectiveness in reducing hallucinations. After all, hallucinations are a feature — not a bug.

Thus, in light of the above, sending Multi-Variate Named Entities to the LLM will indeed improve accuracy (as the probabilities will shift in accordance with this input). Therefore, some embodiments where 100% accuracy is not required may choose to do so.

However, merely using Multi-Variate Named Entities as additional context will not *eliminate* hallucinations. In order to produce hallucination-free, accurate, relevant responses, Multi-Variate Named Entities need to be used in a novel manner — via the novel method disclosed immediately below. ...

More information on implementing accurate AI is available at: https://www.michaelcalvinwood.net.