# SYSTEM AND METHOD FOR RELEVANT AI RESPONSES

## Need for Relevant AI Content Generation

At present, Large Language Models (LLMs) and the chatbots built on top of them fail to produce sufficiently accurate and relevant responses. The industry at large is currently pursuing a wrong strategy for resolving the interrelated issues of accuracy and relevancy. In fact, AI leaders such as Google are pursuing a *backwards* strategy that is literally the *mirror opposite* of what is needed to produce perfectly accurate, fully relevant AI responses. Consider Google's Gemini 1.5 Pro as a perfect case in point.

Gemini 1.5 Pro offers a *context window* of 1 million tokens. The context window is the maximum input + output that an LLM can handle in a single request. Google's researchers present Gemini Pro as a "near-perfect" model: "Gemini 1.5 Pro achieves **near-perfect 'needle' recall (>99.7%) up to 1M tokens** of 'haystack' in all modalities, i.e., text, video and audio. It even **maintains this recall performance when extending to 10M tokens in the text modality (approximately 7M words)**." (Source: "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context" by Google's Gemi Team, https://storage.googleapis.com/deepmind-media/gemini/gemini_v1_5_report.pdf, last accessed March 16, 2024.)

Yet, even pro-Google, pro-Gemini, reviewers report real-world experiences that stand in stark contrast to the notion of a "near-perfect" model. For example, Dipanjan Sarkar, Data Science Lead at Towards AI and published Natural Language Processing (NLP) author, is one such example. This Google/Gemini enthusiast was granted early access to Gemini 1.5 Pro. In one test, he loaded the contents of three videos into Gemini's context window. Each video described how to build a separate character for a popular Role Playing Game called Honkai Star Rail.

- The first video was a guide for building the character *Blade*.
- The second video was a guide for building the character *Ruan Mei*.
- The third video was a guide for building the character *Dan Heng*.

Sarkar asked Gemini to give "a full character build for Blade." Sarkar reports: "However it does **a fundamental mistake** here by **suggesting the 4-piece Thief set for relics when that is actually the best set for another character in the 2nd video I put (Ruan Mei)**." Gemini mixed up the "relics" used for "Blade" and "Ruan Mei" — a hallucination predicted by this present inventor's Noun-Phrase Collision Model — the model that both explains the root-cause of such

inaccuracies and also provides the solution to finally fixing them. This model that solves the issue of *accuracy* has already been detailed in a separate disclosure. This present filing discloses additional methods needed to fully resolve the issue of *relevancy*, as well as detailing processes for combining relevancy and accuracy into a cohesive whole. However, it is first important to establish the current unfulfilled, deep-felt need for accurate and relevant AI responses.

Sarkar sought to fix the situation by giving Gemini another prompt that was "a bit more specific." Sarkar then reports: "It does get it correct this time ... Of course **it's still not a 100% correct because** ..." As a pro-Google, pro-Gemini enthusiast, Sarkar presents the response as "correct" despite the acknowledgement that "it's still not a 100% correct." (Source: "How Good is Google Gemini 1.5 With a Massive 1 Million Context Window?" by Dipanjan Sarkar, https://pub.towardsai.net/how-good-is-google-gemini-1-5-with-a-massive-1-million-context-window-b386d285845d, last visited March 16, 2024.)

Sarkar's real-world example is examined in more detail below in demonstration of this present inventor's novel system and method that fixes the root cause of these inaccuracies — a method that would allow Gemini Pro to produce hallucination-free answers to the very same prompts.

Importantly, the hallucination issue Sarkar encountered with Gemini 1.5 Pro is not a cherry-picked anecdote. On the contrary, LLMs can often excel in laboratory benchmarks and then fall apart the moment they are deployed for real-world tasks. In real life there remains an unfulfilled, deeply-felt need for accurate and relevant AI responses — as the following section demonstrates.

**Actual Performance**

Many in the industry have written blog posts focusing exclusively on the "near-perfect" 99.7% statistic prominently featured in Google's report ("prominently featured" as it is the only statistic cited in the report's abstract). For example, YesChat AI which has more than 1 million users, wrote a blog post entitled "Google Stuns AI World with Revolutionary Gemini 1.5 Pro Large Language Model." The one statistic repeated throughout this post is the "near perfect 99.7% recall accuracy." (Source: https://www.yeschat.ai/blog-Google-Stuns-AI-World-with-Revolutionary-Gemini-15-Pro-Large-Language-Model-3811, last visited March 17, 2024.) Data scientist Venkata sai santosh cites this singular statistic as "proof" of Gemini 1.5 Pro's accuracy. ("Google Gemini 1.5 Pro Is Insane" by Venkata sai santosh, https://medium.com/@venkata_sai/

google-gemini-1-5-pro-is-insane-213cdea0e0fa, last visited March 17, 2024.)

However, Google's report itself explicitly acknowledges "observed limitations" regarding the 99.7% statistic. ("Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context" by Google Team, p. 12) Moreover, Google further acknowledges that the 99.7% result was produced using "the simplest possible setup." (*Id.*)

> Despite the excellent performance of Gemini 1.5 Pro model on the needle-in-a-haystack tasks for all three modalities, significantly surpassing previously reported results (>99.7% for text, 100% for video and 100% for audio), we also present **early findings of observed limitations**. By design, the needle-in-a-haystack task is a retrieval task measuring recall and so far we have considered **the simplest possible setup**. A natural extension to the task is to increase the number of unique “needles” in each haystack, and require the model to retrieve them all. For a context length of up to 1M tokens, we inserted 100 dierent needles and measured the total number of correct needles retrieved. (*Id.*)

Documenting the current unfulfilled need for accurate and relevant AI responses requires placing the oft-cited "near perfect" statistic in its proper context. Remarkably, an astute reading of the graphical plots included in Google's report reveals situations in which the model produced **near-zero accuracy**. The following places both the "near-perfect" results and the "near-zero" results into proper context — revealing the actual performance of Gemini 1.5 Pro — demonstrating the still unfulfilled need for accurate and relevant AI responses.

The previously reported 99.7% result was in regards to the Single Needle Benchmark — where the AI attempts to find *one* single statement that is relevant to the given prompt. As discussed in detail below, vector databases can do single needle searches independent of LLMs. The more interesting question is how well the LLM does in finding *multiple* relevant statements related to the user's prompt. In other words, how well does Gemini 1.5 Pro do on the Multi-Needle Benchmark?

As reported by Matthias Bastian, co-founder and publisher of THE DECODER:

> If you dig a little deeper into Google's technical report for Gemini 1.5, **you will find a graphic** that shows that Google has not yet solved the lost-in-the-middle problem.
>
> The more complex retrieval task "multiple needles in a haystack" test, in which up to 100 specific pieces of information are extracted from the text, shows **an average accuracy of between 60 and 70 percent, with numerous outliers below the 60 percent mark**.
>
> (Source: https://the-decoder.com/dont-get-too-excited-about-google-gemini-

pro-1-5s-giant-context-window/, last visited March 16, 2024.)

In fact, six of the outliers in the aforementioned graphic were **close to zero percent**. (See original Google report for confirmation: "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context" by Google Team, p. 12.)

Moreover, the overall statistic does not mean that Gemini will even achieve a 60 - 70 percent performance rate in the real world. As Bastian notes:

> Moreover, even **the "multiple needles in a haystack" test is simplistic compared to most real-world application scenarios**, where you are not looking for specific information in a set of data, but rather solving complex tasks such as summaries and analyzes.
>
> (Source: https://the-decoder.com/dont-get-too-excited-about-google-gemini-pro-1-5s-giant-context-window/, last visited March 16, 2024.)

It's important to note that the "needles in a haystack" benchmark is a measurement of *relevance* (not *accuracy)* at least in terms of the manner in which these two words shall be used herein. Accuracy is applied *after* the relevant information has been located. Thus, as Bastian notes, Gemini fails to identify 30-40% of the statements most of the time (while missing close to 100% on occasion). Next, the AI must accurately use the identified statements to fulfill the chosen task (something challenging to do with so much of the relevant information already missing).

The actual, real-world performance of an AI system depends on the combination of both relevancy and accuracy. The following is a simple, yet useful, formula for computing the performance of an AI chatbot: (% relevancy) x (% accuracy).

In other words, if the AI system only locates 60% of the relevant statements and achieves a 60% accuracy when utilizing them then the overall performance is 36%. Despite the simplicity of this metric, it's very useful for one important reason: it treats relevancy and accuracy as equally important.

Consider 100% acurracy and 0% relevancy as a perfect case in point. If the LLM responds with "I don't know" that's 100% accurate and 0% relevant. At the end of the day, this is 0% performance because the user still has 0% of the information that (s)he was looking for. The same goes for 0% accuracy and 100% relevancy. If the AI embodiment finds all the relevant statements, but it hallucinates when presenting all of them, then the user is still left with 0% information (as 100% is *hallucination* not *information*).

Hence, the combination of both relevancy and accuracy is a very useful metric to use when assessing an LLMs performance *in the domain of Textual Question/Answer Chatbots* — a very important distinction discussed immediately below.

**Achieving Maximum Peformance for Textual Question/Answer Chatbots**

Today's AI explosion is now being used for a myriad of things, including coding, computing, and querying. For the purposes of this present disclosure, *querying* shall refer to the situation in which the user asks a *Question/Answer Chatbot* to generate a factual response based on *textual* information contained in a *knowledge base* (where the knowledge base could be the entirety of text accessible via the internet). Such chatbots can be requested to provide an answer to a question, or to write a factual news article, or to generate a factual blog post, or to create a factual product entry for a catalogue, or even to write an entire factual essay.

While this present disclosure solves the issues of accuracy and relevancy for a very common use case, this present disclosure does not inherently resolve all issues of accuracy and relevancy for other AI use cases such as coding and computing. However, also disclosed below are methods that can be used to maximize both accuracy and relevancy for certain commonplace analyses that users often request. Contrasting the findings of two expert reports in a court case is one such example. The application of this present system and method within the domains of basic analyses (comparing and contrasting) is additionally disclosed herein.

While this present disclosure may not resolve all the issues for other use cases, it does present methods can be used as the building blocks for doing so. For example, eliminating noun-phrase collisions (as explained below) may not fully resolve the issue of accuracy for coding requests such as "write a Python script that ...". However, such embodiments will still need to eliminate noun-phrase collisions using the methods disclosed herein as building blocks for achieving the desired accuracy — since failure to do so will still create a degree of hallucinations (as also explained below).

Thus, any embodiment that uses any of the methods disclosed herein — either in whole, or in part, or any obvious derivative thereof — to improve accuracy and/or relevance for any AI/ML/ NLP use case falls within the Spirit and Scope of this disclosure.

The rest of this disclosure proceeds with the understanding that the reader shall consider all explanations and conclusions to be within the context of a Textual Question / Answer Chatbot

(unless otherwise specified). For example, the following section discusses achieving maximum performance. This section, and the sections that follow, are to be read in the context of Textual Question / Answer Chatbots. In other words, the following section discusses achieving maximum performance within this specific, commonplace use case.

## Marrying Accuracy and Relevance for Maximum Performance

As previously stated, relevancy and accuracy to two separate, yet interrelated issues. ***Relevancy* shall herein refer to finding facts within the knowledge domain that are useful for responding to the user's query. *Accuracy* is a measurement of the degree to which the generated concurs with the provided facts.**

This present inventor considers there to be two large issues that must be overcome in regards to relevancy. Consider the common practice of creating a knowledge base by storing slices of documents in a vector database. Let's say that one of document slice says the following "Mary likes to browse the internet using her iPhone 15." Thus, a corresponding embedding is stored in the vector database — an embedding that is a numerical representation of "Mary likes to browse the internet using her iPhone 15."

Consider a user who submits the following prompt: "Tell me about the device that Mary uses to visit websites." In common practice, the user's prompt is converted into embedding — a numerical representation of "Tell me about the device that Mary uses to visit websites." Such numerical representations, called *vector embeddings*, allow AI implementations to perform *similarity searches*. For example, the phrase in the prompt "uses to visit websites" is conceptually *similar to* the phrase "browse the internet" that is contained within the document slice. Vector embeddings allow AI systems to recognize that such phrases are similar, and to even compute a percentage representing the degree of linguistic similarity.

Hence, the prompt "Tell me about the device that Mary uses to visit websites" is converted into a numerical representation — converted into a vector embedding. The degree of linguistic similarity between the prompt and each document slice is found by performing a mathetmatical operation on the prompt's vector embedding and the vector embedding corresponding to each document slice. In the above example, the following document slice would likely be computed as the most similar document slice: "Mary likes to browse the internet using her iPhone 15."

In the present state-of-the-art, the AI system would send the LLM both the prompt ("Tell me

about the device that Mary uses to visit websites") and the "context" as well (i.e. the slices of documents that are most similar to the prompt itself). In other words, the AI system would send something similar to the following:

---

Fulfill the provided Prompt solely using the information in the provided Context below.

Prompt:

Tell me about the device that Mary uses to visit websites.

Context:

Mary likes to browse the internet using her iPhone 15.

---

Consider an AI embodiment that uses GPT 3.5 Turbo Model 1106 as the LLM. In this present example, the LLM has been provided the name of the device that Mary uses: "iPhone 15." However, that's all the LLM knows because the iPhone 15 was released in September 2023 (long after the GPT 3.5 Turbo Model 1106 had finished training). In other words, this LLM has no internal information regarding the iPhone 15. Therefore, the LLM has only two choices: solely state the name of the device, or start making things up. Given that the user told the chatbot that it must "tell about the device," the latter option may indeed be the chatbot's chosen course of action — start making things up.

When chatbots do not have sufficient relevant information, they often make things up in an attempt to fulfill their instruction. This can often be mitigated by using prompt engineering to constrain the chatbot to solely use the provided context when constructing its answer. But even where mitigated, this results in *extremely poor performance* from the user's perspective. The user wanted information about the device (e.g. what size screen it has, when it was released, what ports does it have, etc.). Hence, this is an instance of extremely poor performance due to a very low percentage of relevance (performance = % relevancy x % accuracy).

Continuing with this example, let's say that another document contains the statement: "The iPhone 15 contains a USB-C port." This document slice is relevant to fulfilling the user's prompt. However, this slice has virtually zero wording similarity to the original prompt: "Tell me about the device that Mary uses to visit websites." Hence, this document slice would not be classified as relevant when the similarity score for this slice is computed in relation to the user's prompt. This is an example of what this present inventor considers the first large issue that needs to be

addressed in to solve relevancy.

The first large issue is that vector embeddings provide *linguistic* similarity searching. The phrases in the document slices will only be considered relevant if they contain *similar words and phrases*. There are no linguistically similar words and phrases in "The iPhone 15 contains a USB-C port" and "Tell me about the device that Mary uses to visit websites."

Some advanced embodiments in the state-of-the-art attempt to address the limitations of vector embeddings through *hybrid* searching. Consider Microsoft's Azure AI Search:

> In Azure AI Search, vector fields containing embeddings can live alongside **textual and numerical fields**, allowing you to formulate **hybrid queries** that execute in parallel. Hybrid queries can take advantage of existing functionality like filtering, faceting, sorting, scoring profiles, and semantic ranking in a single search request.

In other words, they allow for a combination of similarity searching based on vector embeddings along with the more traditional keyword searching (search for matching text and numbers). But such hybrid searching does nothing to resolve the problem at hand, as there are no keywords in the prompt that would match any of the keywords in the statement regarding iPhone 15's USB-C ports.

While the provided example was intentionally written for ease of explanation, it is conceptually based on a real-world issue repeatedly encountered during this present inventor's experimentations with developing methods to find the maximum number of relevant facts for any given user prompt.

The second large issue comes into play when *none* of the chosen slices of the documents chosen retrieved from the vector database are relevant to the user's query. For example, consider a user prompt that specifies the name of a person, yet the slices of documents sent to the LLM all refer to a different person who has the very same name. This can also occur in a support chatbot that may send information regarding a different product model than the one the user wants, because both models have the same product name. There are numerous instances in which computing similarity based on vector embeddings results in many irrelevant statements being sent to the LLM. The LLM then constructs a response based on irrelevant information, causing a very bad user experience — a potentially existential crisis for companies depending on the severity of the mismatch and the type of chatbot being used.

Hence, relevancy requires minimizing the number of irrelevant statements while

simultaneously maximizing the number of relevant statements as well. Stated another way, maximum relevancy is achieved when sufficient relevant facts are located and irrelevant passages are sufficiently filtered out. (A metric for measuring filtering sufficency is described later below.)

Once all the relevant statements have been located, relevancy must still be married with accuracy for maximum performance. For example, consider the hallucinations that Sarkar encountered when prompting Google's Gemini 1.5 Pro regarding the Role Playing Game. The LLM was provided *all* the relevant information needed to correctly fulfill the prompts. Nevertheless, the LLM hallucinated incorrect responses.

In regards to Sarkar's experiment, this present inventor's Noun-Phrase Collision Model predicts:

- The vast majority of the response will be accurate.
- There will be occasional hallucinations due to the limited presence of noun-phrase collisions.
- The number of hallucinations Sarkar experiences would increase if he added information regarding building additional characters — even if Sarkar kept the overall token consumption the same (at 700k tokens).
- Sarkar could achieve 100% hallucination-free responses by eliminating noun-phrase collisions.

The first two predictions are confirmed by Sarkar's report itself. The majority of the response was accurate. The reported hallucinations were due to a collision over a colliding noun phrase. Sarkar's report confirms that the noun-phrase "relics" caused the LLM to traverse an erroneous "relics" noun-phrase path — in that the LLM traversed the "relics" noun-phrase path related to *Ruan Mei*. The final two predictions can be verified through empirical experimentation.

Remarkably, Sarkar's experience is easily understood in light of this present inventor's Noun-Phrase Collision Model. In fact, the cause of the hallucination is self-evident in light of this model. However, it is the model that makes the cause self-evident. In fact, Sarkar's report shows that he did not truly understand the cause of the hallucination, as demonstrated by the way he tried to fix it.

Sarkar sought to fix the hallucination by providing a more specific prompt. And herein lies the *wrong* pursuit of the industry as a whole. Prompts can be important (as discussed farther

below). **However, prompts alone can *never* eliminate hallucinations when the *context window contains noun-phrase collisions*. Moreover, the greater the frequency of noun-phrase collisions, the greater the rate of hallucinations — even in the presence of the most optimal prompt.**

Had Sarkar understood this, he would have worked on developing a different method to inform the LLM of how to build each character — a method that ensures there are no noun-phrase collisions when providing such information. Nevertheless, the root cause of the hallucination eluded Sarkar — just as it has eluded the industry at large. Just because the cause of the hallucination is obvious in light of the Noun-Phrase Collision Model, that does not mean it is obvious without it. It is the model itself that makes the cause obvious. It is the model itself that makes methods for fixing the issue obvious as well.

The Noun-Phrase Collision Model has been discussed in the previous disclosure on how to build *accurate* AI systems. This present disclosure teaches how to build AI systems that produce *relevant* responses. This present disclosure also teaches how to marry both relevancy and accuracy for maximum performance. Therefore, in view of the latter objective, a brief summary of the Noun-Phrase Collision Model is presented later below. Then detailed methods for finding the relevant facts for each user query is provided below. Then instructions are given on how to marry the relevant facts with the aformented Noun-Phrase Collision Model to achieve maximum performance. Then instructions are given to extend this performance beyond mere information retrieval and presentation. Such instructions related to basic analyses of contrasting and comparing. Finally, this disclosure discusses using the methods contained herein as building blocks for providing maximum performance for other use cases as well.

## Industry's Backwards Approach

From the perspective of the Noun-Phrase Collision Model, Google's pursuit of a 1 Million context window is a *backwards* approach. It is based on the wrong notion that as long as the input window contains the needed information then the LLM will be able provide an accurate, relevant response. Therefore, if one can just throw everything into the input window (including the kitchen sink) then the LLM can provide the correct response.

This "throw everything possible in there" approach is truly backwards and wrong. The Noun-Phrase Collision Model states that one wants to *limit* what is sent to the LLM — limited by the

avoidance of sending colliding noun phrases. The fact that this present inventor's disclosures are opposite of the current industry pursuit evidences that novelty of this present inventor's disclosures. The methods disclosed herein are both novel and useful. They are non-obvious to those ordinarily skilled in the art, and they fulfill a deep technological need that has remained unresolved up to this present time.

It bears noting that the methods disclosed herein provide utility also in regards to both processing time and cost. Processing 1 million tokens is both slow and expensive. The methods detailed herein are both fast and cheap.

Which leads to yet one additional utility of the disclosed system and method — less powerful LLMs can be used to provide fully relevant, perfectly accurate responses. In other words, the system and method disclosed here allows smaller LLMs to provide relevant, hallucination-free responses. The minimum criteria of such LLMs is discussed below. Also discussed is building LLMs that can fit on mobile devices, allowing anyone to have instant access to relevant, hallucination-free responses wherever they go.

Hence, the methods disclosed herein not only resolve the issues of relevancy and accuracy, but they also resolve a number of other practical issues as well, including speed, cost, and general availability.

## Noun-Phrase Dominance Model

A prior disclosure discusses the Noun-Phrase Collision Model in depth, as that disclosure focused on teaching how to build *accurate* AI responses. Below is a sufficient explanation to lay the groundwork for marrying the relevancy methods disclosed below with accuracy to produce maximum performance.

The Noun-Phrase Collision Model is part of this present inventor's larger Noun-Phrase Dominance Model. The Noun-Phrase Dominance Model states:

1. LLMs self organize around noun-phrase paths during training. This is only natural given that the English language itself is organized around noun-phrase paths.
2. A conflicting noun-phrase path is where there are multiple statements using the same noun phrase, but the information describing each noun phrase is erroneous when it is applied to any other of the noun phrases.
    a) For example: "**The store** opens at nine. **The store** sells shoes." This creates two

noun-phrase paths off the noun-phrase "the store." **The two paths are: "opens at nine" and "sells shoes."** If both statements are about the same store, then there is no noun-phrase collision because either path can be applied to "the store." However, if they are talking about two different stores then this is a noun-phrase collision (as the two paths cannot be factually applied to either store).

3. The probability of which noun-phrase paths get traversed is dependent upon the current text in the sliding input window. Where noun-phrase collisions exist, there remains a degree of probability that the colliding path can be selected and traversed.
    a) For example, if there are multiple stores being discussed, and one store is mentioned in the context window, then the noun-phrase paths related to that store will have a higher probability of being traversed. However, there still remains a degree of probability for the colliding paths to be traversed — which results in a "hallucination" where the information regarding one store gets wrongly conflated as being about the other store.
4. The rate of hallucinations increases based on the frequency of noun-phrase collisions — whether in the provided prompt, the provided context, or both.
    a) The hallucination rate increases *in proportion to* the frequency of collisions in *the provided context*.
    b) The hallucination rate *exponentially increases* with the frequency of collisions contained in *the prompt*.
        (1) As a reminder, the content of the prompt adjusts the probabilities for the paths that are selected in the provided context. If the prompt itself contains multiple collisions then this affects the probability distribution for the paths in the context.
            (a) For example, if the prompt mentions two stores then there can no longer be one store that receives a very high probability with the other stores receiving significantly lower probabilities. In this situation, the noun-phrase paths for both of the mentioned stores are assigned high values — causing much greater hallucinations as the LLM now frequently traverses the colliding noun-phrase paths given the existence of multiple high probability stores.

5. Documents in the same knowledge domain often use the same words in reference to different things. Hence, documents in the same knowledge domain often cause a higher rate of hallucinations (as they contain a higher rate of noun-phrase collisions).
6. Hallucinations can be eliminated by eliminating noun-phrase collisions provided two straightforward caveats are met.
    a) First, every individual statement must be independently literally true. (Naturally untrue statements lead to untrue output.)
    b) Second, the provided context must itself be organized around clear-cut noun-phrase paths so that the LLM can easily select and traverse them. Simply stated, the input must be full sentences.
        (1) Input that does not consist solely of noun-phrase paths is misaligned with the LLM architecture. One common example are citations. These sometimes lengthy text chunks are not full sentences. (I.e. they are not comprised of noun-phrase paths.) Therefore, they throw the LLM off track. This is why LLMs are notorious for seeming to make up citations out of thin air. (Methods for encoding citations to align with LLM architecture are provided below — finally resolving a very common source of hallucinations — thereby expanding the application of LLMs for use in academic research and writing.)

In terms of Sarkar, each of the videos was regarding the same knowledge domain (building characters) using the same noun-phrases (e.g. "relics") in reference to two different things. After all, the "relics" noun-phrase paths of one character cannot be applied to the "relics" noun-phrase paths of another character. Hence the reported hallucination.

Notice that the reported hallucination did not come from thin air! Nor was it even a fluke. On the contrary, it's the predictable outcome based on the provided input (predictable based on the Noun-Phrase Dominance Model). Hence, in this situation the solution to fully accurate responses cannot be achieved through prompt engineering nor through providing additional context. In fact, the latter would only increase the hallucination rate if such context introduced additional noun-phrase collisions.

[End of Public Disclosure]

*More information on implementing accurate AI is available at:*

*https://www.michaelcalvinwood.net.*