InScightful: Enterprise Generative Decision Support

Ben Sprott
Founder, Architect
Cavenwell Industrial AI

Introduction

We are entering the commodification phase of generative AI. At the same time, enterprise customers are disappointed with the current state of generative technology. The reasons they give include cost, data security, decision transparency, and response accuracy. These concerns are high on the minds of CEOs, CIOs, and analytics executives, who are trying to see through the fog of hype and attempt generate real ROI as they adopt these technologies into the conservative business.

Our team has been focused on the shortcomings of neural architecture since day one and have developed InScightful, a generative decision support tool to be the one correct solution to bring powerful AI into the enterprise and give a high ROI.

InScightful enables better decisions faster and easier. It consists of a model-agnostic meta-wrapper (essentially a "meta model") for many foundational models including Gemini, ChatGPT, Llama3, Claude and many others. It solves all the technical problems that are worrying enterprise through a patent pending architecture that removes neural networks from the reasoning and analytics core. Users do not have to rely on untrustworthy neural architecture to write code or perform best effort language-based reasoning. Instead, they can have natural language inputs that then return with reports that are based on a solid foundation of quantitative reasoning and probability.

InScightful Technology

InScightful is a powerful meta-wrapper on top of any language model that is sufficiently capable of a few basic features. The LLMs released in 2024, tend to have enough capability to used in combination with InScightful. Since there is not a heavy amount of work that the LLM must do for InScightful to function, the cost of running the models is lower. Furthermore, on premises solutions are a cheap, fully secure option.

In terms of hallucination, our product is the only technology on the market that make the claim that notions of accuracy are simply non-existent. The reason for this is due to our special architecture, which we will give details of later. Our software does all reasoning and data science in a way that allows us to claim that it is provably correct. It is essentially the implementation of mathematics for doing data science in a structured, natural way. InScightful is the only system that provides a

degree to which tabular data supports a given natural language utterance, and this ability comes again with the feature of being provably correct.

InScightful architecture

The most important feature to note about InScightful is that it essentially deletes the neural network architecture (as seen in standard AI and LLMs) from the core analysis and reasoning that is the most important component of supplying decision support. InScightful is a meta-wrapper on top of any sufficiently powerful LLM, but they are used only at the input edge of the pipeline and the output edge of the pipeline.

The first part of the pipeline is a user who has a natural language query and a dataset.

The next part of this pipeline is a transformer. This can be ChatGPT, Gemini, LLama3 or any other LLM that the user is accustomed to and has a license to use. The job of this transformer is to take the natural language query and discover a scientific hypothesis within the statement. For instance, if the user is a marketing expert and they ask, "Does a red website theme lead to higher new subscription counts in October.", the LLM has to recognize how this is related to the dataset. The job of the LLM is to convert the sentence into an arrow or circuit over the columns in the dataset. In the case of our marketing expert, the LLM will return an arrow like this:

$$|Month = October\rangle \text{AND} |WebsiteTheme = Red\rangle \rightarrow |NewSubscriptionCounts = high\rangle$$

It is now the job of the interior of the pipeline to convert this tuple into a network of probabilistic relations called Morphisms in the Kliesli category of the distribution monad, $\mathcal{K}l_D$. They are essentially probabilistic functions and they are also stochastic matrices. These probabilistic functions are transformations that map discrete probability distributions into discrete probability distributions.

From the sentence, a few distributions have to be computed. First, the situation the customer is interested in comes in the form of a probability distribution. In the case of our marketing expert, the input distribution is this:

$$100\%|Month = October\rangle \otimes |WebsiteTheme = Red\rangle$$

The expected output distribution is "New Subscription Counts = high", and so we just need to look at the data in this column and select the top 80 or 90 % of values to get a uniform distribution over the top, say 90% of values. This is the expected output distribution.

Next InScightful will perform many different types of statistical inference that are all provably correct and produce a report.

The final part of the pipeline is, again, an LLM to which is passed the statistical report to generate a readable report. This can be tailored in different ways. For instance, the LLM can be asked to use the McKinsey pyramid method in its report using the statistical information. Along with this, various charts and graphs can be generated to support the decision.

How is this like a neural network?

Neural networks are essentially general linear maps that map a vector to a vector. The important feature is the intermediate maps between the input and the output, which are called hidden layers. These are networks of maps, again from vectors to vectors. Through the process of backpropagation, we can learn hidden features, which essentially take the right parts of a vector and use them as maps to important other vectors. One great example is face recognition. Is this a face or not is a good question to ask of any image and if it is a face it probably has two eyebrows in it. Backpropagation allows the system to know when any grouping of elements in the vector "are" an eyebrow. Then by combining this with other regions that might be a mouth, a hairline, eyes, ears and a node, we can have a layer that takes all this information and decides that this image contains a face.

The important point is that this is all just vectors and linear maps with learned parameters that tell you which parts of the input vector do or do not contain the important hidden features. You can replace the vectors with distributions and $\mathcal{K}l_D$ maps and this was explained in detail here [1].

Black boxes, Circuits and Transparency

If you use distributions instead of vectors, your basic element, the layer, has an extra meaning, namely that they are distributions, which carries significant mathematical value. If you expand your function

$$Image \to \{IsA\_Face, IsNotA\_Face\}$$

Into a network of probabilities, you will be able to ask questions of your neural network in ways that you cannot do with vectors. You can ask, what is the probability that this image contains an eyebrow.

You can ask, "Does the existence of an eyebrow mean that the image contains a face?" or "Images with eyebrows are faces", etc.

Combining Variables

There is a tensor product in $\mathcal{K}l_D$. This is just the cartesian product of sets. You can combine variables and begin to ask questions like, "Does a poor diet and no exercise lead to obesity?" This means that AND has a clear semantics embodied in a tensor product that is just the Cartesian product.

Wires Going In and Out

One of the major developments of applied category theory over the past 30 years has been graphical calculus. It starts by doing the Poincare dual of diagrams in a category, and viewing the morphism as boxes, while the wires are seen as conduits along which information (in the form of vectors or distributions) is understood to "travel".

When we start to think about AND, we can quickly see that a box with two wires coming in and one wire coming out, is a map from two variables to just one variable, as in the sentence "Does a poor diet and no exercise lead to obesity?".

The important feature, which has been expressed well in several publications [2], is compositionality. In wiring diagrams, this just amounts to connecting the boxes together at will whenever the type of the input wire of one box is equal to the type of an output wire on some other box. Boxes can be put side by side, implying the tensor product of morphisms. Wires can be put side by side to imply the tensor product of variables.

Computing Probabilistic Functions

Computationally, there are choices to be made when constructing a circuit made of probabilistic functions that correctly give the degree to which the data supports a hypothesis or an explanation. You can precompute all possible morphisms with N wires in and M wires out, where you fix N and M. Using "and" repeatedly in a sentence has its limits and certainly in common language, so M and N do not need to be very high. Once you have all these precomputed $\mathcal{K}l_D$ morphisms, you can put them in a database and recall them in O(1) time. Alternatively, you can compute them at run time. This is done through a process of what is called "Bayesian Disintegration". Naturally, this is more costly in terms of run time computing but can be used to create fairly fast decision support tools without the need to precompute morphisms. With this, the designer can make choices between space and time complexity.

Natural Language

Over the past 20 years, neural networks have been used to model natural language in terms of LLMs but also a lot of work has been done on the vector space semantics of natural language [3]. With vectors, there will be a natural way to have an inner product between words and utterances and this has been used to accomplish many amazing things.

The semantics of natural language is fairly open, with different categories of objects and morphisms supplying different levels of accuracy and usefulness. It should be clear that $\mathcal{K}l_D$ should support the semantics of natural language. There is one major shortcoming of that category, which is that while it contains monoidal objects, it does not contain comonoidal products and more importantly, it does not contain Frobenius objects.

NLP Semantics On Probabilities

InScightful uses a new category which is the coma category of the one element set of the Kleisli category of the distribution monad, $1/\mathcal{K}l_D$. This category is equipped with both the moniodal

objects as well as a dagger. The dagger essentially takes the transpose of the distributions and stochastic matrices. Combining these two things, we then have bi-algebras. These bi-algebras should equip $1/\mathcal{K}l_D$ with enough structure to capture a certain, important fragment of natural language. Our patents claim that the computer system is free to use the ambient properties of $1/\mathcal{K}l_D$ as a semantics of natural language to convert natural language utterances into networks of probabilistic functions and distributions.

Our patent also claims any transformer that can transform natural language into wiring diagrams as part of the pipeline. It also claims specialized transformers that can then be used to convert natural language directly into circuits of $\mathcal{K}l_D$ morphisms to then be quickly converted into circuits of probabilistic functions. It also claims any transformations one may make on these circuits to simplify or expand the logic in the sentence for proving hypotheses or giving explainable proofs.

Implementation Details

InScightful is a server application written in Python, Javascript, React, and Flask. It is meant to run on a basic machine on the customer's premises. It is deployed in a Docker container, so that customers simply need to have Docker running on a machine and then launch the Docker file. It then opens a port and this allows anyone on the network to access it like any other website.

The user is intended to input their enterprise data into InScightful. This can be done either by database connectors or simply by uploading several CSV files. No cleaning or precomputing needs to be done on the data. This is done by InScightful.

The server architecture includes a main Javascript application and several Python micro-services that run in Flask.

Depending on architectural choices, the data may be processed over several days, creating the morphism database that will be called by the software at runtime. Alternatively, a pc with many processors, such as 80, can compute any needed morphism at runtime when the user inputs a query.

The software will have the ability to do many things, like storing and saving people's queries and producing reports that can be stored and shared. There will be administrative and user privileges.

To answer a query, a language model must be included. This can be done in two main different ways. If the user is comfortable sharing their data with an LLM service, then they need only give the api key for their preferred LLM service. Then, at run time, small amounts of data are shared with the LLM to answer the question. This data includes column headers and the set of values that are found within each column. Identifiable records are not shared.

The other option is to run a local LLM. This can be done with the addition of two or three modest GPUs to the computer. In this case, nothing is shared outside of the user's network.

Benchmarks For Generative Decision Support

We will now make comparisons of the output of InScightful versus all the major foundational models. We will use an example that will come up in the use case section. Here are the columns of the data and here is the query: “I think Dr. Gregory is often late.”

InScightful vs. LLama3

The first thing to note about Llama3 is that it does not consume tabular data. This means that the user must integrate into an application. InScightful does this correctly. Rather than asking Lllama to write code to answer the query, it is used just to return the type of the $\mathcal{K}l_D$ morphism embedded in the sentence and to summarize the statistical analysis and quantitative reasoning. The code that Llama3 returns is buggy and basic and not worthy of repeating. This is not a real comparison.

Let us move on to the more apt competitors.

Anthropic Claude

Here is Claude’s response after being given the dataset and the query:

> *Total Cases: Dr. Gregory has performed 22 cases in the dataset*
> *Late Cases: 18 out of his 22 cases started late (81.82% of cases)*
> *Average Late Time: 4.59 minutes per case*
> *The average surgeon has about 61.49% of their cases starting late*
> *The average late time across all surgeons is 1.66 minutes per case*
> *This data suggests that your observation has some merit. Dr. Gregory does have a higher percentage of late starts (81.82%) compared to the average (61.49%), and his average delay time (4.59 minutes) is higher than the overall average (1.66 minutes).*

Gemini

Here is Gemini’s response to the dataset and the query “I think Dr. Gregory is often late.”

> *The data doesn't support your claim that Dr. Gregory is frequently late.*
> *Dr. Gregory was late in 18 out of 22 cases, which translates to 81.82% of cases.*
> *The cases are mostly late on Wednesdays and Sundays.*
> *Let me know if you'd like to explore the reasons behind this pattern.*
> *The average delay on Wednesdays is significantly higher than on Sundays, suggesting that other factors beyond the day of the week may be contributing to the delays.*
> *All late cases on Wednesdays and Sundays were classified as 'CHEO - Elective' and involved the 'Cardiovascular' service. The most common procedures were 'TETRALOGY REPAIR' and 'ATRIOVENTRICULAR SEPTAL DEFECT (AVSD) REPAIR, ATRIAL SEPTAL DEFECT SECUNDUM (ASD II) REPAIR', each occurring twice. The average length of these cases was 288.571 minutes, with a maximum of 360 minutes. Most patients were 'Surgical Admit' cases.*

Use Case: Operating Room Scheduling

The situation

Let us imagine a hospital that runs operating rooms. To ensure the efficient usage of those operating rooms, workers known as schedulers, take in data from the Hospital information system and then build schedules. Their goal is to fit as many cases as possible into a two-week period, some time in advance.

One of the schedulers becomes concerned that one of the doctors is late arriving at their operations. It is a sensitive matter for a scheduler to approach a surgeon. Simply going up to them in the hallway and asking why they are always late is a bad decision. Instead, they turn to InScightful.

They use InScightful like a well-informed colleague and state the following query: “I think Dr. Gregory is often late.”

The InScightful Response

> The data weakly supports your hypothesis. The quality metric is 20.0 %, where 100 is "Absolutely True" and 0% is "Absolutely False".
>
> *We identified a scientific hypothesis in your statement: "I think Dr. Gregory is often late." You expected that Dr. Gregory, as the lead surgeon, would often be late, with 'often' corresponding to 50% of the time being 9 minutes late and 50% of the time being 10 minutes late. This expectation was based on the output probabilities for the column 'Late Time (minutes)'.*
>
> *Upon analyzing the data, we found that the actual distribution of lateness for Dr. Gregory was quite different. The computed output conditional probabilities were: 18.2% for 0 minutes late, 18.2% for 10 minutes late, 13.6% for 4 minutes late, 13.6% for 7 minutes late, and 9.1% each for 1, 2, and 5 minutes late. This distribution was based on the input assumption that Dr. Gregory was the lead surgeon 100% of the time.*
>
> *The data weakly supports your hypothesis. While Dr. Gregory is late 18.2% of the time by 10 minutes, which aligns with part of your expectation, the rest of the distribution is quite different. The input, 'Lead Surgeon', and the output, 'Late Time (minutes)', are related, but not as strongly as you hypothesized.*
>
> *The relationship between Dr. Gregory as the lead surgeon and the late time occurs under specific conditions. These include when the case classification is 100% CHEO - Elective, the service is 100% Cardiovascular, and the day of the week is Wednesday.*
>
> *Interestingly, if we reverse the relationship and consider how 'Late Time (minutes)' predicts the 'Lead Surgeon', we find that your input distribution of 50% for 9 minutes late and 50% for 10 minutes late actually predicts that Dr. Gregory is the lead surgeon 100% of the time. This reverse relationship could be useful in decision-making processes, as it suggests that if a surgery is running late by 9 or 10 minutes, it's highly likely that Dr. Gregory is the lead surgeon.*

The scheduler can see a few things here. First, the doctor is often late on Wednesday. Second, when you look at late doctors, the only one is exactly who she thought of, namely Dr. Gregory. With this information, she can go to the doctor and have an informed discussion. She simply asks “Why are you sometimes late on Wednesday?”

The doctor replies that his wife is now working early on Wednesdays, and he has to drop his children off at school. He says he was trying to fix it but couldn’t. The scheduler then suggests that the doctor be taken off

of Wednesdays, and the doctor replies, no, that instead he should just be taken off of Wednesday mornings. This is then input as a constraint into the scheduling system.

This clearly demonstrates a valid use case for decision making in a real- world scenario. We have shown this demo to the workers at the Children's Hospital of Eastern Ontario and it was taken well.

Summary

The key points to be raised, especially taking into consideration the benchmarking, is that InScightful is giving supporting evidence in probabilities and verifies the statement with a quantity. It can give a quick answer as to how true the sentence is. It then breaks down the meaning of this quantity by showing exactly in what way the sentence is true or false. InScightful points out clearly that the doctor is only sometimes late and it happens under specific circumstances. It also does the extra work of looking in the opposite direction, i.e. predicting surgeon from "being late". This confirms the suspicion that the one doctor to be concerned about is Dr. Gregory. All this is supported by the exact probabilities from the data. Furthermore, the report can be very much more succinct if we just tell the report generator to be more succinct.

Acknowledgement

This project was completed with the assistance and guidance of several of the world's greatest living applied mathematicians. A team at University College London helped. This was Mehrnoosh Sadrzadeh and Fabio Zanasi. Others who helped were Paolo Perroni and Joseph Beals.

I would also like to acknowledge the help of Dr. Sandeep "Sandy" Muju. MD Plutuskuber Ventures, who helped me with clarifying the thought process and the writing of this paper.

References

1. Jacobs, Bart, and David Sprunger. "Neural nets via forward state transformation and backward loss transformation." *Electronic Notes in Theoretical Computer Science* 347 (2019): 161-177.
2. Fong, Brendan, and David I. Spivak. "Seven sketches in compositionality: An invitation to applied category theory." arXiv preprint arXiv:1803.05316 (2018).
3. Kartsaklis, Dimitri, et al. "Reasoning about meaning in natural language with compact closed categories and Frobenius algebras." Logic and algebraic structures in quantum computing (2013): 199.