Symbolic Logic is Also Needed

Greg Coppola

December 24, 2023

Abstract

In this paper, we present the **Quantified Bayesian Network** (**QBN**). This is a data structure that, we believe, can end the problem of *LLM hallucination*. The **QBN** provides a **generative model** of the **logical structure** behind the sentences in natural language. If the world knowledge of the LLM can be transferred into the QBN, we believe that that will constitute AGI. The QBN is able to generative model data. It is a generalization of both first-order predicate logic, as well as the Bayesian Network. A QBN can represent logical relationships between logical propositions.

1 Introduction

1.1 Importance of LLM's

The **Large Language Model** (**LLM**) is the hottest data structure in artificial intelligence and computer science right now. Pioneered by the product *ChatGPT* by *OpenAI*, leading to a robust competition in artificially intelligent agents, often called "chat bots," or simply *LLM*'s.

There is widespread agreement that the *LLM* is a powerful technology. For example, neutral but well-known investor Jason Calanis recently said of the *LLM*-based technology boom:

> AI is the opportunity of our lifetimes... bigger than the internet, PCs, mobile, and cloud – combined. [4]

But, there are some major, well-known limitations with *LLM*'s that limit their applicability.

The main problem is **hallucinations**: the LLM does not know what it does not know, and this lack of reliability limits the ability of the system to be used fully automatically, i.e., without a human "in the loop."

1.2 Short History of LLM's

Bengio et al. [1] introduced a novel neural network-based approach to language modeling, significantly impacting subsequent developments in natural language processing. Sutskever et al. [8] presented a breakthrough in sequence-to-sequence learning, revolutionizing machine translation and other sequence-based tasks in AI. Vaswani et al. [9] introduced the Transformer model, fundamentally changing the landscape of neural network design for language processing. Devlin et al. [3] proposed BERT, setting new standards in natural language understanding tasks. Radford et al. [6] advanced language understanding through generative pre-training, paving the way for sophisticated language models. Radford et al. [7] demonstrated the versatility of language models as unsupervised multitask learners. Brown et al. [2] showed that language models can be effective few-shot learners, expanding possibilities with minimal training data. Kaplan et al. [5] explored scaling laws for neural language models, providing insights into model size and performance.

2 Limitations of LLM's

2.1 Hallucinations

The model is "hallucinating" when it is answering a question with an answer completely unsuported by the training data. This is the most commonly listed problem with LLM's. Sutskever has said this limits the usefulness of LLM's because there must always be a human "in the loop", and so things cannot be fully automated.

2.2 Cannot Do Reasoning

It is known that LLM's cannot do reasoning. Hinton has underlined the problem that LLM's 1) do not have a "logically consistent worldview" and

2) do not understand how different "worldviews" (what we call *theories*) lead to different conclusions.

2.3 Our Analysis

These two problems are related. In order to not hallucinate, a model must be able to explain *why* it believes what it does. But the ability to answer *why* implies an understanding of *causality*. And, *causality* is just another way of saying *logical inference*, because you cannot have one without the other.

3 Existing Approaches to Hallucinations

3.1 Fine-Tuning

Fine-tuning is the strategy that *ChatGPT* originally used to constrain the generative model, and is also used in the competing products. Fine-tuning is a process by which access to the activation outputs of the underlying LLM generative model are used to train a *discriminative* model to achieve certain tasks. This gives an option to reduce many kinds of LLM erros, but not all, because hallucination remains an unsolved problem. Also, we think we should aesthetically object to the *necessity* of a discriminative model in order to not hallucinate. If the spirit of the LLM is to be *generative*, we should push for a generative model that can avoid hallucinations on its own.

3.2 Retrieval Augmented Generation

Retrieval Augmented Generation (RAF) is a process by which access to an extrinsic traditional *discrete* database-based information product, especially a *search engine*. The idea is that a discriminative or even programmatic process can be created that 1) uses the LLM activations as features, and 2) has access to the discrete outputs of the traditional search engine. This way, the content of the answer can be checked against the web page. One limitation of this is that *it does not let the LLM make its own theory*. We are limited to agreeing with the web page. Also, as with fine-tuning, we believe it is an aesthetic problem that the *generative* part of the model cannot stop hallucinating by itself.

3.3 Trusted Sources

We list **trusted sources** as its own element, for clarity. However, the *trusted sources* strategy is ultimately a RAG strategy, so the same arguments would apply.

3.4 Vector Databases

Vector databases have been proposed as a solution to hallucinations. However, vectors do not have any relation to logical calculus that is established. Thus, there is no concept of *true* or *false* with vectors in a vector database any more than there is truth or falsity in the LLM. Also, empirically, this approach has not been gotten to work, despite being suggested as early as *ChatGPT*'s hallucination issue was realized.

4 Logic

4.1 Propositional Logic

Logic comes in a a few forms, varying in their levels of complexity, and scope. The most basic logic is *propositional logic*. Propositional logic deals with statements like A, $\neg A$, $A \wedge B$, $A \rightarrow B$. This is limited because it does not allow different propositions to "share" any of their "meanings". That is, in the propositional calculus, if A represents the sentence *John runs* and B represents the sentence *Mary runs*, there is no way to represents that both sentences share something in common, because A and B are different letters, with different indices, sharing no reusable parts.

4.2 First-Order Logic

First-order logic solves this problem of sharing structure between similar sentences by giving us *predicates*, *entities* and *quantifiers*. Thus, we can say that *John runs* as $run(john)$ and *Mary runs* as $run(mary)$, and we see that both expression share some part of the structure, run, corresponding to our idea that the surface forms *John runs* and *Mary runs* are related.

First-Order Logic also makes use of *quantifiers*. The sentence $\forall x, late(x) \rightarrow runs(x)$ says that *for any person, if they are late, they are running*. This kind of sentence allows us to identify *patterns* between facts, e.g., being late

and running. The problem with First-Order Logic is that it is too *deterministic*. In real-life scenarios, we exploit patterns that are true *most* but not *all* of the time. Thus, it might be that being late *tends to lead* to running, but it doesn't always.

4.3 Beyond First-Order Logic

First-order logic is not enough to model all human natural language. This is not to say that it should be thrown out, but that there are certain "extensions" to first-order logic that are needed to model the most sophisticated examples of natural language. First of all, *intensional logic* is needed in order to allow the "concept" of a sentence to be the argument to a predicate, rather than simply its truth value (one of *true* or *false*). Intensional logics are less well-agreed upon that first-order logic. But, in any case, they use the same mechanism of universal quantification, only over a specially constructed entity set, based on the logical language. Second, first-order logic by itself does not tell us how to build up the meanings of complicated expressions, e.g. *the Mayor of New York*, *the Mayor of New Delhi*, from their simpler parts. This is again simply an area where there is less clarity than first-order logic, only because there are more options, but there are many options to choose from.

4.4 Conclusion

In the following work, we will focus on the first-order logic. First, because all other logics are extensions of the first-order logic, so we can reuse the same techniques for training a first-order system. Second, for simplicity. Third, because there is not a consensus on the details of the more complicated extended logics.

5 Bayesian Networks

We want to look at a certain kind of data structure in computer science: the Bayesian Network.

5.1 Basic Concepts

A Bayesian Network, **B**, consists of:

- A set of variables $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$.
- A set of directed edges $\mathbf{E}$ between these variables, which represent causal relationships.

Each variable X_i is associated with a probability distribution that quantifies the effects of the parents of X_i in the graph.

5.2 Advantages of Bayesian Networks

Bayesian Networks offer several advantages:

- They provide a clear and visual representation of causal relationships.
- They allow for efficient computation of conditional probabilities.
- They enable the integration of both prior knowledge and observed data.

6 The QBN Model

6.1 Entities

An entity is an atomic concept in logic. One can get very philosophical and mystical about the question "what is an entity." However, from an information retrieval system, it is very simple: each entity is associated with a unique identifier (say a string). Associated with this unique identifier we can associate *properties*. For example, in Google's *Knowledge Graph*, starting circa 2012, is a system which will display *cards* about a notable person, e.g., *Taylor Swift* or *Beyonce* is searched for. Each one of these tracked notable figures would correspond to an *entity* in this information retrieval database.

6.2 Propositions

Propositions are statements in our logical language that can either be **true** or **false**. It is possible to get very philosophical about the question "what is *truth*." In this case, we can side-step philosophico-mystical debates by appealing to a simple notion of "truth" that the LLM is not currently offering us, which is that: we want the LGM to be able to accurately represent what was in its training set. In other words, if a LGM tells us that a certain piece

of information is in its data set, we should be able to find it there. In fact, an LGM should be able to tell us why it believes a certain statement, and provide links to all of the data that was in its training set.

6.3 Semantic Roles

For us, a *predicate*, which is also a node in the probability graph, is a *key-value* tuple. The keys come from a fixed set, called *roles*. That is, while the number of different *languages* is not bounded, and, for any given language, the number of words or morphemes is not bounded, the set of roles *is* finite, bounded, and fixed, for all languages.

6.4 Arguments

Each semantic role contains as its argument either:

1. a *constant*, which identifies a concrete element in some universe (or database)

2. a *variable*, which ranges over some *domain* of entities in the database

A variable specifies a *domain*, each domain being a subset of the entities.

It is important to note that the entities are very high-level type categories, constant, finite and fixed in number ahead of any data. We choose the domains *entity*, *relation* and *predicate*. Entities refer to ontological *objects*, like *jack1* and *jill1*. Relations refer to what we would call *verbs* in linguistic analysis, such as *likes* or *dates*. Predicates, in this formulation, are unary predicate words like *lonely* or *exciting*.

It is important to note that any domain quantified over requires an assumption of membership. For example, the only thing we can say about a (hypothetical) entity, like $jack1$, is that it is an entity, rather than a relation or a predicate. To say that *jack* is a *man*, *woman*, *human*, etc., is to say something empirical that could, *technically* be true. However, it may be that it is more efficient or otherwise beneficial for engineering reasons to strictly assume category membership. This is what we do in our experiments, where we explicitly denote the set of $Jacks$ versus the set of $Jills$ to model a binary dating scenario.

In other words, consider the statement: * $\forall x \in Jack, y \in Jill, likes(x, y) \rightarrow dates(x, y)$ This is equivalent to: * $\forall x \in y, Jack(x) \wedge Jill(y) \wedge likes(x, y) \rightarrow dates(x, y)$

Thus, the probabilities $P(Jack(x))$ and $P(Jill(y))$ are implicitly part of the model. The only way we can state that quantification ranges over a subset of entities in the basic type is if we are willing to model the probability of membership as definitely 1.

6.5 Predicates

In traditional first-order logic, a predicate would be a statement like $likes(x, y)$ in the overall sentence $\forall x \in Jack, \forall y \in Jill, likes(x, y) \rightarrow dates(x, y)$.

We will refer to any statement in which the arguments contain at least one variable.

In other words, in our logical language, predicates include functions like:

- $\{relation : likes, subject : x \in Jack, object : y \in Jill\}$
- $\{predicate : lonely, subject : x \in Jack\}$
- $\{predicate : exciting, subject : x \in Jill\}$

In other words, a predicate is intuitively a function, where the input types are specified in the predicate defintion, and the output is a binary value.

6.6 Propositions

Propositions are statements in the logical language without quantification.

Using this semantic-role notation we represent the traditional first order proposition $likes(jack1, jill1)$ as:

- $\{relation : likes, subject : jack1, object : jill1\}$

We can conjoin sentences like the following, expressing that $jack1$ and $jill1$ both like each other:

- $\{relation : likes, subject : jack1, object : jill1\} \wedge \{relation : likes, subject : jill1, object : jack$

6.7 Predicate Matcher

From one perspective, a predicate is a mapping from inputs to truth values, as we saw. From another perspective, we can view a predicate as being mapped one-to-one with a *proposition matcher*. The proposition matcher

associated with a predicate q matches any proposition p such that p is an instantiation of q, with values of the specified types.

For example, $p = \{relation : likes, subject : jack1, object : jill1\}$ is an instantiation of $q = \{relation : likes, subject : x \in Jack, object : y \in Jill\}$ using the subsitution $x = jack1$ and $y = jill1$.

Thus, $match(q, p)$ is true. However, p does *not* match $q2$, because its values for relation is $dates$, which does not equal $likes$. $q2 = \{relation : dates, subject : x \in Jack, ob$ using the subsitution $x = jack1$ and $y = jill1$.

Similarly, p does not match $q3$, because $q3$ specifies that the subject must be a $Jill$, not a $Jack$. $q2 = \{relation : dates, subject : x \in Jill, object : y \in Jack\}$

7 Substitution

8 Implication Links

The overall concept of the Quantified Bayesian Network is to allow a trainable network encoding and generalizing the concept of first-order quantification. That is, we want to generalize the ability to make statements like $\forall x, yexciting(y) \rightarrow likes(x, y)$. However, we must emphasize that the *entire point* of training a machine learned system is that statements of the form "*all* X are Y" rarely hold in practice. But, advantage is gained merely by noticing a correlation.

So, what then is first-order implication if we relax the literal logical implication? From our perspective, the generalization is simply to say that the premise and conclusion of first-order logic are *related* in some way (or, at least possibly related), and it is up to the model training to decide which.

9 Matcher

A matcher takes a predicate q and a proposition p and returns match(p, q):

1. None – if p does not match q
2. Some(r) – where r is a role mapping $\{r : a\}$ such that substitute$(q, r) = p$

Substitution Rule

A substitution rule takes in a predicate q and a role mapping r and gives back a proposition p, by:

1. type-checking that the role map r corresponds to the arguments in p
2. substituting the roles in role map r as the variables to premise predicate q

For example, the value of:

$$\text{substitute}(\{\text{relation} : \text{predicate}, \text{subject} : \{x \in \text{Jack}\}\}, \{\text{subject} : \text{jack1}\})$$

is $\{\text{relation} : \text{predicate}, \text{subject} : \text{jack1}\}$.

10 Forwards Links and Backwards Links

The generative story proceeds "forwards". That is, where the nodes of the Quantified Bayesian Network form a Directed Acyclic Graph, we recursively can compute the value of a node n by doing a depth-first traversal of its ancestors in the DAG. We refer to the link from a premise to a conclusion as a *forward* link.

Thus, in order to find the possible causes of a given node, we must traverse the edge in the opposite direction. We call the link from a conclusion to a premise a backwards link.

We will focus on forwards inference in this paper, and leave backwards inference for future work. Thus, we are primarily concerned with *backwards* links, and this is what is currently implemented in the BAYES STAR package.

11 Generalization Results

We propose that the **QBN** is a generalization of both:

1. *fast thinking* – this refers to "forward" thinking, with a fixed, trained QBN
2. *slow thinking* – this refers to the ability to reason with *or* conclusions

3. *creative thinking* – this refers to the creation and training of *new links*

We believe it is correct in a "strong" sense to say that:

- the Quantified Bayesian Network *generalize* some *important faction* of first-order logic

Now, here we require some care because we do *not* believe that a practical computational system *should* generalize full first-order logic in an overly specific way. This is because there are different kind sof thinking.

11.1 Thinking Fast And Slow

11.1.1 The Literature

Thinking Fast and Slow is a book by TODO that popularized the idea that there are "fast" and "slow" kinds of thinking. Let us put it this way. Everyone can speak and locomocate around the world. An act as simple as crossing the street requires some kind of logical calculation, based on the goal of *not getting hit by a car*.

Thus, we see that all people can think.

But, not everyone can do quantum physics and not everyone can do complex mathematics. Major mathematical results often take years of collective work by scientists. *Fermat's Last Theorem* took TODO years to solve. Obviously some kinds of thinking are slower than others!

11.1.2 Our Interpretation

We propose that there are three kinds of *speeds* of *thinking*:

1. *fast thinking* – this refers to "forward" thinking, with a fixed, trained QBN
2. *slow thinking* – this refers to the ability to reason with *or* conclusions
3. *creative thinking* – this refers to the creation and training of *new links*

11.2 Fast Thinking in a QBN

We will use the analogy from psychology to describe the QBN. This is not surprising, because the QBN is modeled on the human brain.

We believe that the *fast thinking* subset of first-order logic is that part which:

1. only uses conclusions that do not contain the symbol $\cup$ TODO

That is, let us look at some *complex* and *simple* statements:

1. if A and B, then C – simple
2. if A or B, then C – simple
3. if A, then B and C – simple
4. if A, then B or C – complex

Intuitively, the complex situation is when we learn that either one thing is true *or another*, and we don't know which.

For example, consider the planning of an outting to the beach. If we get reliable information that it *will* rain, then we can pick an indoor activity instead. If we get reliable information that it *will not* rain, then we can plan the trip to the beach. If we don't know if there will be rain, we are in a complicated position. Now, we must come up with a strategy that works for both cases (rain and not rain). A full decision analysis would require us to:

1. determine the probability of rain
2. determine possible courses of action
3. define the utility of each possible outcome
4. compute the expected utility

12 Knowledge Transfer from LLM to QBN

12.1 Review of Assumptions

We have advanced the hypothesis that:

LLM's Have World Knowledge We are assuming that LLM's *do* have world knowledge. This could be because the query-key-value function of the attention mechanism is actually discovering the semantic role nature of natural language.

And, at this point we are assuming that the QBN is an "appropriate" knowledge engine:

QBN is "Appropriate" By "appropriate", I mean that whatever requirements we have of an *inference engine*, an "appropriate" engine meets all of those.

So, with these two assumptions, then since 1) knowledge is in the LLM and 2) if we could get it into the QBN we would be good, the question now becomes: how do we transfer knoweldge from the LLM to the QBN?

12.2 Latent Syntactic Structure

12.2.1 CFG Parsing

The original annotation standard was CFG phrase-structure labeling. This labeling scheme recursively divids a sentence into recursively contained, non-overlapping *sub-spans* of the sentence. This is like the CFG phrase-structure described by Chomsky. However, the phrase-sturcture formalism was perhaps always just a compromise: instead of picking some complex, theory-laded formalism on offer from Linguistics, the phase-structure representation is the simplest phrase-structure syntactic analysis possible. Thus, while phrase-structure parses deeply offend no one, they also endear themselves to no one: working with a CFG parse is not convenient or powerful.

12.2.2 Unlabled Dependency Parsing

One way to look at a phrase-structure parse is not in terms of *spans*, but in terms of lexical head-modifier *dependencies*. This is attractive because the word-word dependencies seem more semantically "meaningful" than phrase structures. In other words, in *John loves Mary*, intuitively the relationship between *John* and *loves*, or between *Mary* and *John* is, intuitively, more relevant than the indicies of the span that either phrase takes up. A flurry of work looked at learning dependency parisng directly.

12.2.3 Labled Dependency Parsing

However, in practice, it is virtually useless to have a sentence that is annotated with only an unlabeled parse. In order to use a parse, one must not only know *which words modify which heads*, but also to know the *label* sub-type of that modification. For example, in *John loves Mary*, both *John* and *Mary* modify the verb *love*, however knowing who loves who requires knowing which is the *subject* and which is the *object*. Reversing the roles played by the arguments reverses the meaning of the sentence. Fortunately, algorithms that work well for unlabeled dependency parsing also work well for labeled dependency parsing.

12.2.4 Parsing to Semantics

Bar-Hillel showed how a semantic intepretation can be created for a surface form as a by-product of parsing using a *categorial grammar*. Steedman's *combinatory categorial grammar* showed how to extend this to cases with non-projective dependencies. Ultimately, we can create a semantic using semantic role labeling, any time we can get a *labeled dependency parse*. The problem with traditional "labeled dependency parsing" is that these labels are automatically extracted from a CFG-labeled treebank (e.g. Penn Treebank). Thus, the labels in these schemes do not really correspond to any actual logical inference language that can be used for thinking. So, the task in practice is to arrive at a *labeled* dependency parse. The question of which *labels* exist, and how we will determine them are open questions. Assuming that there *is* some labeling, we can begin discussing how to learn it.

12.3 Generative Syntactic Models

Now, there are generative models, especially the LLM. Before this trend, the most accurate syntactic analysis methods were *discriminative*, meaning that they could estimate the *conditional* probability of an analysis given a sequence of tokens, but could not generate the tokens based on a generative model.

12.4 A Generative Transformer-Based Model

In order to generate the surface form (words data), there will need to be a *recursive tree-structured* generative model in the style of Collins' thesis parser.

The parses are latent so must be estimated with expectation maximization. The generative probability of generating a parse is a function of 1) the logical probability according to the QBN that a given logical form would be expressed in a given linguistic context, 2) the syntactic probability that a given logical form would be expressed in a particular way.

12.5 Syntactic Parses are Latent States

The problem with using syntactic parses, relative to LLM's–and this is probably the reason n-gram LLM's worked before structured ones–is that one must *infer* a latent state. That is, the *correct* syntactic analysis is "hidden." This corresponds to the fact that we can *definitely* see what a person has written, or hear what they have said. But, we can *never* be sure that we know exactly what they *meant*. This is the basis for someone saying that they have been "taken out of context." In other words, the allegation that one is being taken out of context is the allegation that their *intended* latent meaning is not the one being ascribed. Thus, clearly, even humans cannot be sure about the *correct* analysis, if by "correct" we mean "intended."

13 Future Work

13.1 Transferring LLM Knowledge to QBN

In order to build full AGI, we need to combine the LLM, which has the knowledge now, to the QBN, which can store the data in a "logically perfect" way. Although this is an area where litle is known, it is an area where we believe it is easy to work along the obvious dimensions.

13.2 More Complex Logical Problems

We would probably actually be more interested in developing a way to encode complex logical problems.

13.3 Functions (Beyond Predicates)

Another question is the extension to functions. The ability for human people to do math and calculate sums and products is an interesting phenomenon.

To what extent is addition an example of *thinking fast*? Well, obviousy people must use paper for complicated operations, so it must be thinking slow. But, what about the most basic operations? Perhaps we are equipped to compute basic calculations like $2 * 2$ fast, but $22 * 22$ requires paper or a good memory to manage those two operations. In any event, it would seem like the most efficient option for building technology would *not* be to mimic the human mind exactly, since computers are better at computing. Instead, we would probably want to break with whatever humans do for complex calculations, and instead just use the computer. Karpathy TODO has said that in *OpenAI*'s strategy is to use hard-coded functions for things like math, and only use the LLM to detect when a math problem is relevant.

14 Conclusion

We have presented the Quanified Bayesian Network, a probabilistic graphical model that allows for 1) a generative model of the logical forms behind language, 2) that can explain its reasoning, and so not hallucinate We have described how the QBN must be paired with a tree-structured generative model that generates the surface form (words) data from the logical form chosen.

References

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

[4] Jason. Tweet content, 2023. Twitter post. Retrieved from https://twitter.com/Jason/status/1736986300652433748.

[5] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[6] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[7] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[8] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 27, 2014.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.