On prior knowledge

Research statement, Peter Bloem, 2021

What can be learned?

The simplest approach to machine learning is to train a model that is as large as possible and to feed it as much data as possible, with only minimal engineering effort in the design of the architecture or the source of the data (but great engineering effort in scaling up as far as possible). In recent years, this simple approach has actually led to great breakthroughs, with OpenAI's GPT-3 a prime example. The question arises whether this is all we will ever need. Not just is machine learning, but in AI in general.

As part of my PhD research I answered part of this question using the tools of Kolmogorov complexity. I showed that it is very likely that in the limit of unbounded computational power and an arbitrary amount of data, there is no definite way to separate model from code. There will be several competing models that all represent the data equally well, and some questions will remain unanswerable.

I also proved that, given the right assumptions about our data, we can increase what we can theoretically learn from it. The Kolmogorov complexity, which is famously incomputable, and functions as a proxy for many learnable properties of the data, can be safely approximated if we make assumptions about the way the data is structured.

In statistics, the most common assumption is that our data consists of independent, identically distributed (i.i.d.) entities that were sampled one after the other. This assumption is so common, in fact, that many statisticians ignore that this is an assumption not just about the source of the data, but also about its structure. The idea that we can break our data up into equal pieces is a great luxury.

In other settings, such as graph learning (with the dataset forming one large graph), this luxury is not given, and the picture is far more complicated. I developed a novel method for motif analysis, breaking a graph up into component subgraphs, which replaces the iid assumption by a more general assumption of a modular structure. Recently, I also adapted this approach to knowledge graphs.

In other work, I show how to exploit the assumption that the data is *self-similar*. This leads to a natural algorithm for learning fractal models of data, marrying complexity theory and machine learning. While complexity theory does not take center stage in my current research focus, it remains something I am fascinated by, and for which I see many opportunities in modern machine learning.

Practical relational learning

In recent years, my focus has been the combination of relational knowledge and machine learning. I contributed, for instance, to the R-GCN, one of the main machine learning models in the field. I have also set out a research vision for how R-GCNs and other message passing models can be used to learn end-to-end from integrated relational and multimodal

models. To further this vision I have contributed to proof-of-concept models in this area, and new benchmark datasets to help the field to move forward.

These models form a simple kind of neuro-symbolic integration: they are fed with symbolic data, which potentially integrates various modalities of raw data, and a model is learned on top of this data.

This is a practical, and very powerful approach. However, I believe this approach will never lead to truly intelligent behavior in a learning system. This is because the parts of the system that map to the symbols in the relational data are *hard-wired*. One vector always represents the same symbol: the content of the vector may change, but the number of symbols doesn't.

This means that these system cannot understand, or invent new symbols. If I tell a child that there is such a thing as a *spork*, they may not know what I'm talking about. If I tell them it's a combination of a spoon and a fork, they may come up with two or three different ways in which the two may be combined. If I then show them a picture of a spork, they know exactly what a spork is and how one might use it.

This shows that humans have no problem introducing new symbols into their vocabulary. Even before we know exactly what a spork is and what it looks like, we can give the concept a space in our internal registration, and reason about its properties.

Architecture engineering or data engineering?

The spork problem, I believe, is fundamental to the future of AI. The reason that people can so fluently accept and develop new symbols is that we develop our own, personal registration of the world. We are helped, of course, in our development by parents and teachers, but the symbols they use to teach us are never forced on our internal registration. We are free to place them in our mind as we see fit: making room for new ones when they emerge, mapping them to our existing symbolic and sensory memories and resolving inconsistencies whenever they occur.

This ability is missing from most, if not al,l neuro-symbolic integration available today. Some approaches exist for integrating new data without retraining too much, but these are features bolted on to a rigid skeleton. The ability to adapt both to sensory and symbolic inputs by adapting a non-discrete registration of the world is a challenge that is rarely even voiced explicitly.

So what approaches are available? If we sketch the history of machine learning in broad strokes, we can call the pre-deep learning era the time of *feature engineering*: extracting vectors of features from raw data, and feeding these to simple one-step models. Deep learning replaced this by *architecture engineering*: building a pipeline from raw data to output in which each part is learned, preferably end-to-end. Here, the challenge is to build the right inductive biases into the model that allow it to learn effectively.

In recent years, a third alternative has emerged: *data engineering*. With the advent of transformer models especially, we can create increasingly homogeneous models and simply feed them with large amounts of data. GPT-3 is a prime example. When we find places

where such models struggle to learn concepts, we can instead *shape the data*. If we want the model to learn arithmetic, we can arrange for the data to contain more examples of arithmetic. If we want the model to learn to translate between French and English, we can provide it with samples from a parallel corpus.

So how does this affect the problem of neuro-symbolic integration? Most architecture engineering approaches suffer from the spork problem. They take the symbolic and relational data as a given, and hardwire it into the model.

The data engineering approach is more akin to the way a human mind develops: a large amount of unstructured data, together with a small proportion of tailored input to allow quick and efficient learning of concepts that are difficult or impossible to learn in an i.i.d. setting.

Of course, data engineering comes with its own problems. We lose safety guarantees. A self-driving car that is allowed to learn its own registration of what a green traffic light is, may well optimize its time to destination by broadening that particular registration to unsafe levels. Social impact too, becomes more difficult to quantify and control, when we give up our ability to precisely control how the data affects the registrations of our agents.

I believe both approaches have their place. It is likely that a self-driving car, if it should even exist, should never be a fully autonomous agent. In many use cases, architecture engineering is the only way to ensure the required levels of control. However, in those areas where we want to move towards true intelligence, it will be crucial to relinquish some of our control. Just like raising a child, we must learn to trust the agent to develop its own registrations, and its own mapping of the rules we provide it to that internal registrations.

From this view emerge the research questions that I believe are important in the long term. While some measure of trust will always be required, we can build this trust by careful investigation. We can measure how well the behaviors and registrations of a learned agent correspond to our values and rules. We can measure the impact of data shaping methods on these outcomes. And we can develop formal representations of the rules and guidelines that we want to shape the agent in its development.

Finally, I should emphasize again that both data engineering and architecture engineering have their place in the future of AI. Ultimately, I plan to pursue both approaches actively, both in my own research and the research I supervise.