

Open in app ↗

Sign up

Sign in

Medium

Search

Write



Machine Learning +

Knowledge Graphs +

AI +

Procedural Programming +

Knowledge Graphs: The Third Era of Computing



Dan McCreary

Follow

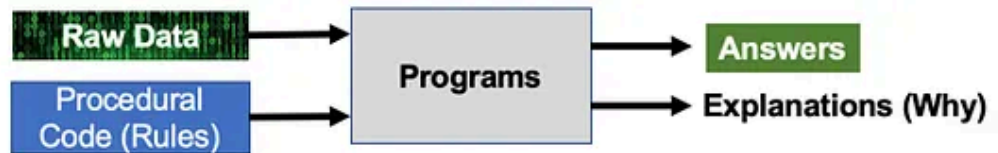
5 min read · Mar 23, 2019

885

3



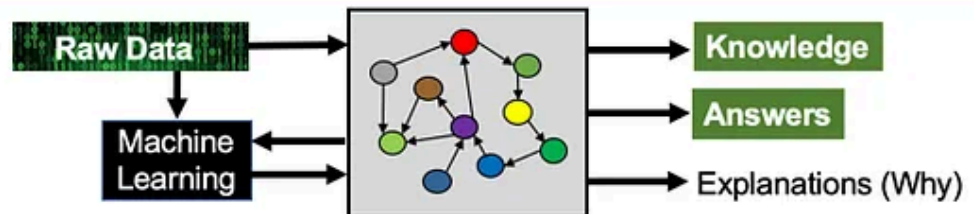
The Procedural Era



The Machine Learning Era



The Knowledge Graph Era



Knowledge Graphs are at the core of the third era of computing where we use machine learning to continuously enrich our shared knowledge

My good friend Ed Sverdlin often begins our “Introduction to Graph Databases” presentations with the question “When did computing start?”.

After he lets the audience guess for a bit, tension builds, and he reveals a picture of a cuneiform tablet from around 3000 BC. He notes knowledge representations really began when we wanted to remember things that were important to us. What were these original things? They were often a ledger of financial transactions such as “Alok owes Sumar 10 baskets of grain.” The key is that it was natural for us to store these facts in rows and columns of a table because tables were a good “natural representation” for financial transactions. These transactions records evolved into rows of symbols which represented concepts and and written languages were born.

What is interesting is that these representation stuck for over 5,000 years. Clay tablets evolved into papyrus scrolls, which evolved into the Luca Pacioli’s double entry bookkeeping system, which eventually became punch cards and then flat files in COBOL, then the tables in a row-store popular in relational databases and finally in our billion dollar Enterprise Resource Planning (ERP) systems that run many large companies today.

These tabular representations have worked well when our problem had uniform data sets. By uniform we mean that each record (row) has similar attributes with similar data types. Now Ed asks the question — *do all business problems fit well into tables?* What about data about your health? Does the electronic medical record fit well into a set of tables? We have things like list of allergies, medical conditions, your genome, and a list of the drugs you taking and your adherence to taking these drugs from your smart pill dispenser, your FitBit step count, your weight measured by your smart scale and other IoT data like your sleep patterns. Then add all the diagnostics blood tests and medical images like MRIs, CAT scans and ultrasound images, EKG measurements... I think you see our point. Not all problems fit well into tables! The more tables you have the more expensive the relational joins become.

Now let's take it one step further. How do we store the **analysis** of your patient chart? You might have a powerful AI agent scanning all your data each night as your IoT and drug adherence data arrives at the data center of your provider or accountable care organization (ACO). What do they produce? The answer is often a list of medical conditions and the probability that you have each of these conditions. And these conditions are often abstract medical concepts like "diabetes" or "COPD". Healthcare systems store these concepts in a complex hierarchy (a taxonomy) with many connections between the concepts (an ontology). Your AI doctor now can make a recommendation on your next best actions — how many steps for today and if you should try that new veggie burger. In summary — this data does not fit easily into a table. But it does fit well into a graph of connected concepts. We call this the knowledge graph.

So how do we get from today's world of 95% of our developers writing PHP/Java/Python over tables to this new era? Perhaps the best way to describe this is to think abstractly about what we are doing today break it down.

The way we describe current generation is to give it a broad descriptive name called the "Procedural Era" described in Figure 2.

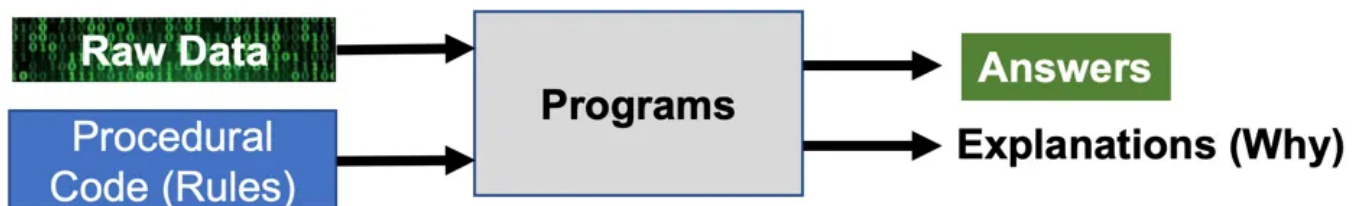


Figure 2: The Procedural Era: where we write step-by-step procedures to find answers in our raw data

This is where developers hand-code step-by-step procedures that take raw **data** and come up with **answers**. The developers need to give explicit

instructions about how to interpret the 1s and 0s on the hard drives and how to describe the results in terms of codes and relationships between these codes. If you want to ask the program why you produced a specific answer you can trace back the decision to set of specific rules that applied to your situation. These “trace backs” make the system easy to explain.

Get Dan McCreary's stories in your inbox

Join Medium for free to get updates from this writer.

Subscribe



Remember me for faster sign in

Next, let's look at the second era: The Machine Learning Era. The process of training a machine learning algorithm is describe in Figure 3.



Figure 3: The Machine Learning Era: where data and answers are fed in and the outcome is a “black box” model with 10 million weights but without explanation of why decisions were made.

This era has become incredibly popular in the last seven years with the development of deep learning algorithms and the use of GPUs to train these networks. Unlike the procedural era, we don't write explicit if-the-else rules for each byte of data in the input. We provide a training set of answers and the machine “learns” a set of complex rules. For example we might “train” a small remote-control car by recording how it should react as it drives around a race track. It looks at the lines on the road and responds with the right speed and steering commands. The rules are typically stored as a set of

weights that are applied to input data as it moves through a network. The problem is that there are many weights, typically ten million in a typical image recognition problem. You can't really ask the system to explain why it made various decisions. It can only tell you the numerical weights it used to arrive at the answer.

Now let's come to the third era of computing, the Era of the Knowledge Graph which is captured in Figure 4.

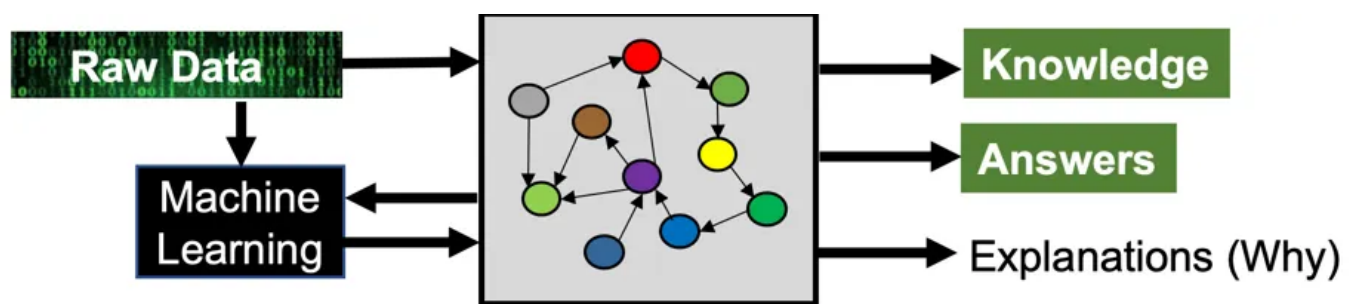


Figure 4: The Knowledge Graph Era: where machine learning continuously reads raw data, combines this with existing knowledge and produces new knowledge, answers and explanations

This is where we combine the best of the first two eras of computing to produce a system that not only learns from complex data, but it also can explain its decisions. On the left we still use machine learning to harvest raw data and look for patterns in this data. Machine learning finds relevant information (people, places and things) in our images, text and sound and then converts this to new entries in our knowledge graph along with confidence weights. This data can then be checked for consistency and quality by graph algorithms. What comes out of the graph is new knowledge, answers and explanations of why we made specific decisions. Our knowledge graph becomes a repository of semantically precise verticis and relationships with confidence weights retained from the machine learning processes. The knowledge enrichment processes are not perfect and can

easily add false assertions if new facts are not curated by subject matter experts.

In many ways, this is a much more accurate representation of how our brains process information. No tables, no if/then/else code. Just real-world concepts and the relationships between them that can be understood by graph traversal algorithms.

However, just like in the real world, if we are constantly exposed to fake news we tend to start to believe it. So their needs to be humans in the loop to verify the new assertions. This verification can be difficult and expensive without crowdsourcing when your knowledge graphs reach the billion-vertex levels.

Machine Learning

+

Knowledge Graphs

+

AI

+

Procedural Programming

+

**Written by Dan McCreary**

3.1K followers · 879 following

Follow

AI researcher helping educational organizations use AI to build Microsims in intelligent textbooks. Fan of Systems Thinking, STEM, microcontrollers, & robots.

