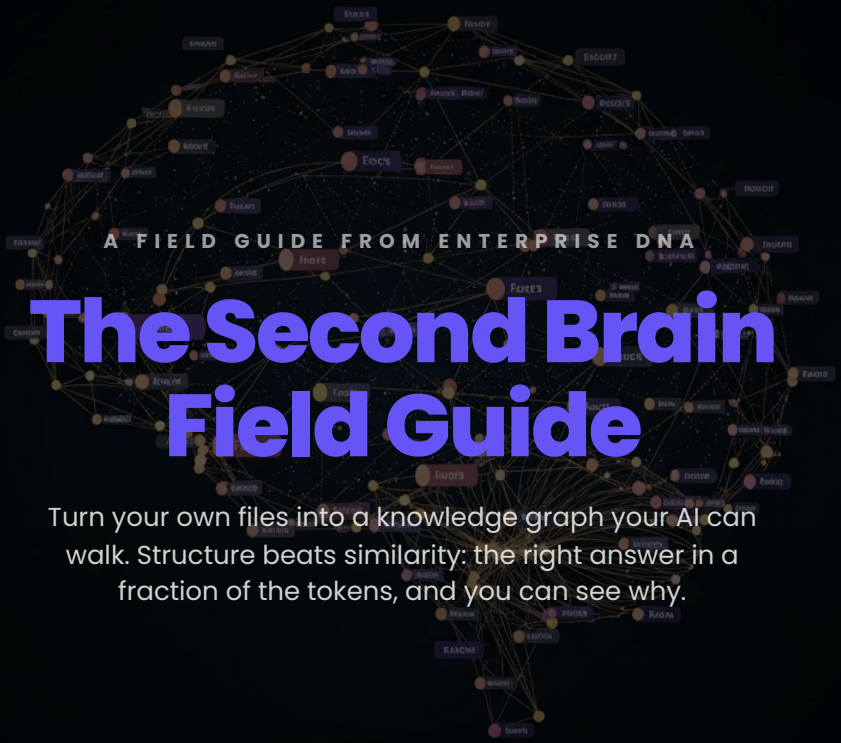
ENTERPRISE DNA



A FIELD GUIDE FROM ENTERPRISE DNA

The Second Brain Field Guide

Turn your own files into a knowledge graph your AI can walk. Structure beats similarity: the right answer in a fraction of the tokens, and you can see why.

FRONT MATTER

A note from Sam

Most people give their AI more context by pasting in more files. That works until it does not. The model forgets, it retrieves the wrong thing, and it burns tokens re-reading everything you own.

There is a better way, and it is not a bigger context window or another vector database. It is structure. You turn your files into a knowledge graph, the entities and the relationships between them, and the model walks that structure to the answer. It reads the path, not the whole library.

This is the idea Andrej Karpathy floated as an LLM wiki, and it is the idea an open source project called Graphify rode to more than 83,000 stars in 2026. Structure the knowledge once, then query it forever.

We do not write this from the outside. We already run the code version of it. Our agents read a graph of our own codebase instead of grepping through it, across more than 25 repositories. Every pattern in this guide is one we use ourselves.

Read it as a field guide. What a second brain really is, why structure beats similarity, and how to build one over your own folder this week.

Sam McKay

Founder, Enterprise DNA

CONTENTS

What is inside

A practical guide for anyone who feels their AI forgets, retrieves the wrong thing, or burns tokens re-reading everything. No PhD in retrieval required.

PART 1 · THE PROBLEM

Just give the AI everything

> *Context stuffing, grep, and vector databases fail the same way. None of them understand structure.*

The instinct is to hand the model more. Paste in the files, widen the context window, or wire up a vector database and let it retrieve. Each one hits the same wall.

Stuffing the context. You can only fit so much, and the more you paste, the more the model has to wade through to find the one line that matters. You pay for every token, and the signal gets buried in the noise.

Grepping the files. Search finds the word, not the meaning. It cannot follow a thread from one file to another, and it has no idea which of forty matches is the one you actually need.

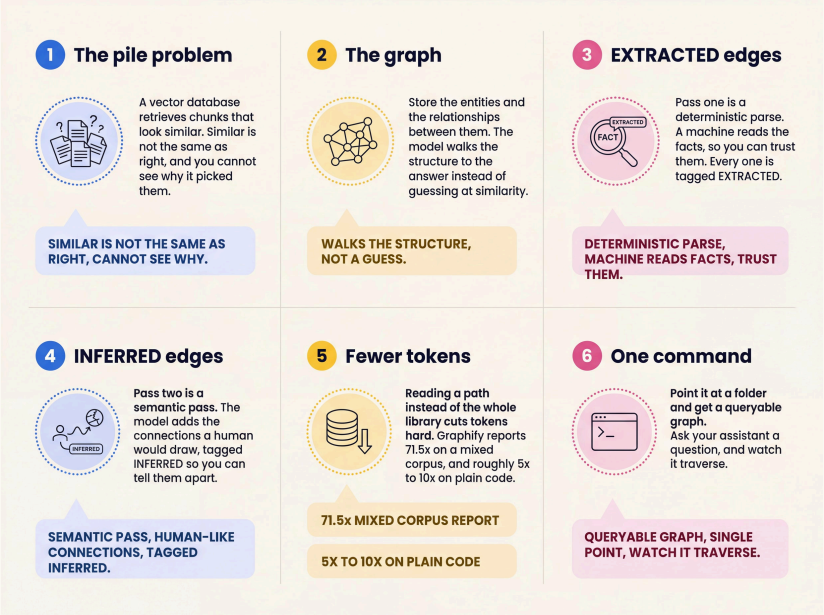
The vector database. The default answer today is retrieval by similarity. It is better than grep, but it guesses. The next page is about why that guess is weaker than it looks.

Why the vector-database default falls short

A vector database chops your files into chunks, turns each into a string of numbers, and returns the chunks that sit closest to your question. The whole bet is that similar text is the right text.

Similar is not the same as right. Ask who signed the contract and it may hand you the paragraph about signing rules, not the signature line, because that paragraph talks about signing more. It ranks by resemblance, not by whether the answer is in there.

You cannot see why. The result is a similarity score, not a reason. There is no thread to follow and no way to check the logic, so a confident wrong answer looks exactly like a right one.



PART 2 · THE IDEA

A knowledge graph, not a pile

> *Store the connections, not just the notes. A graph is something the model can walk.*

A second brain is not new. Tiago Forte popularised the idea in 2022: an external, trusted place to keep what you learn. The trouble is that a folder of notes is still a pile a person has to open and read.

The upgrade is structure. Instead of storing notes and hoping you re-find them, you store the entities and the relationships between them. Who relates to what, what depends on what, which idea generalises another. The connections become data, not something locked in your head.

Karpathy's framing. Andrej Karpathy described this as an LLM wiki. Rather than retrieving raw documents at query time, the model builds and maintains a structured, cross-referenced artifact it can read. The cross-references are already there. The work of connecting is done once, up front.

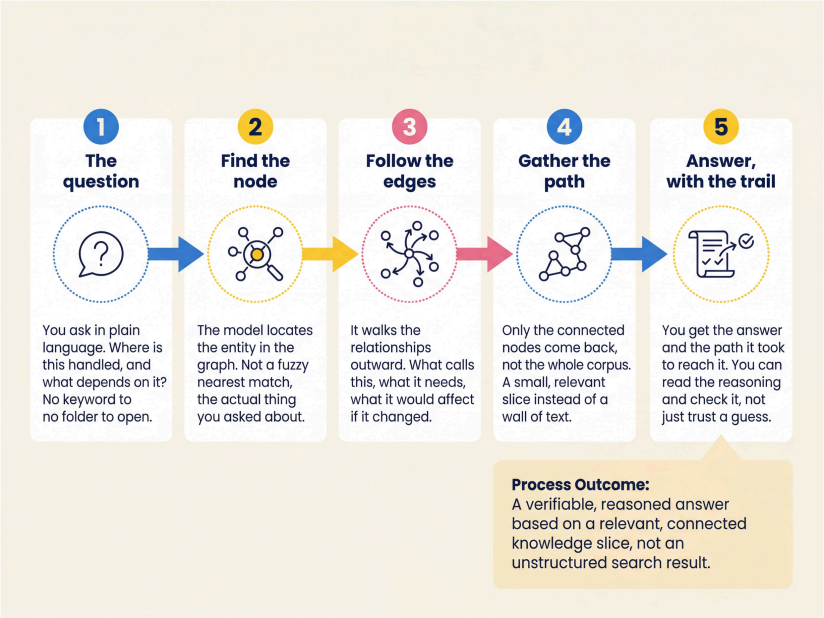
Give the model a graph and it does not guess at similarity. It walks the structure, the way you would follow a train of thought.

Structure beats similarity

Here is the difference in one question. Ask where something is handled and what depends on it.

The similarity way. A vector search returns the chunks that read like your question and stops there. If the answer spans two files, it has no way to connect them. It fetches lookalikes and leaves you to join the dots.

The graph way. The model finds the entity you asked about, then follows its edges outward. What calls it, what it needs, what it affects. Only the connected nodes come back, and the answer arrives with the path it took, so you can check the reasoning.



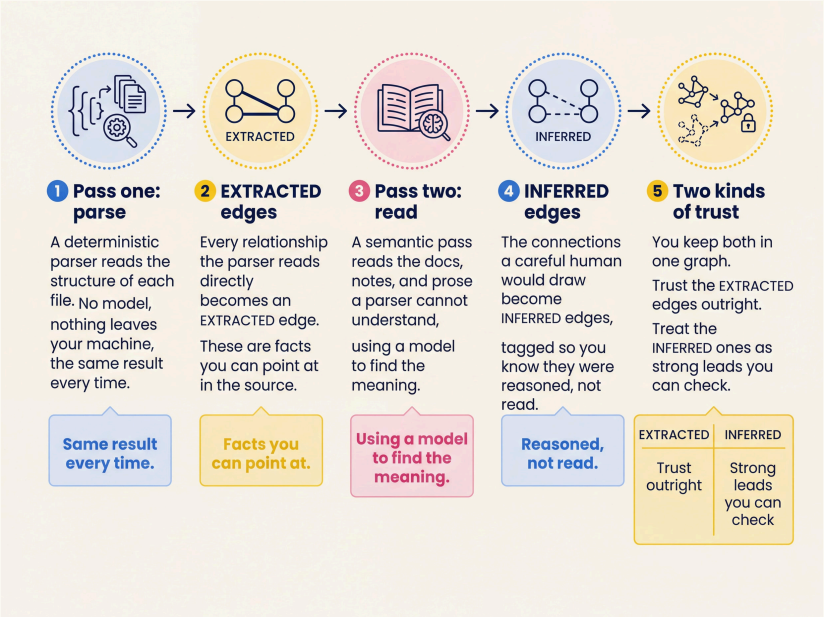
PART 3 · THE METHOD

The two-pass method

A good graph is built in two passes, and the difference between them is the difference between a fact and a good guess.

Pass one reads the facts. A deterministic parser walks your files and records the relationships it can read directly. Nothing is inferred and nothing leaves your machine. Graphify tags each of these an EXTRACTED edge, because a machine read it from the source.

Pass two draws the connections. A semantic pass uses a model to read the prose a parser cannot, and adds the links a careful human would draw. These are tagged INFERRED, so you always know which edges were read and which were reasoned.

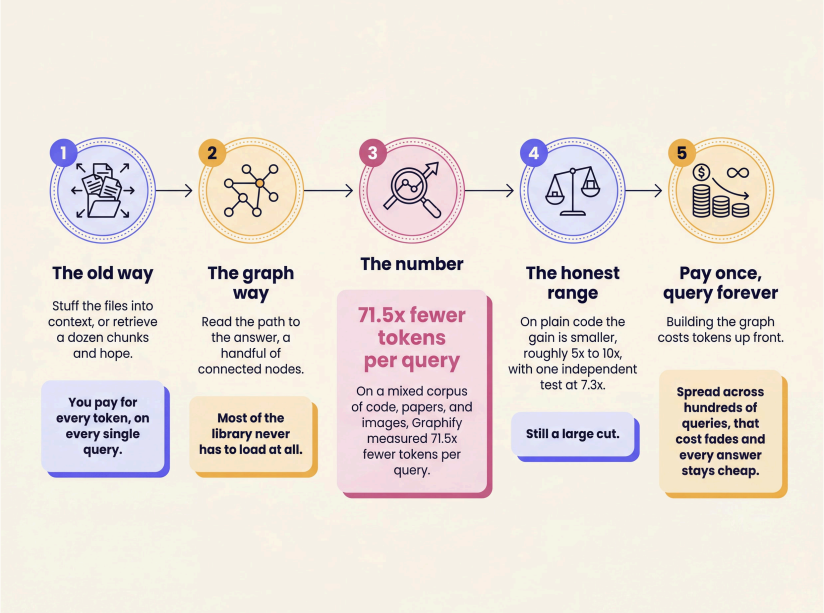


The token math

Walking a path uses far fewer tokens than reading the whole library, and that is where the payoff shows up.

The number. On a mixed corpus of code, papers, and images, Graphify measured 71.5 times fewer tokens per query than reading the raw files. On plain code the gain is smaller, roughly 5 to 10 times, with one independent test at 7.3 times. Either way, it is a large cut.

Pay once, query forever. Building the graph costs tokens up front. Spread that across hundreds of queries and the cost fades, while every answer stays short, fast, and cheap.



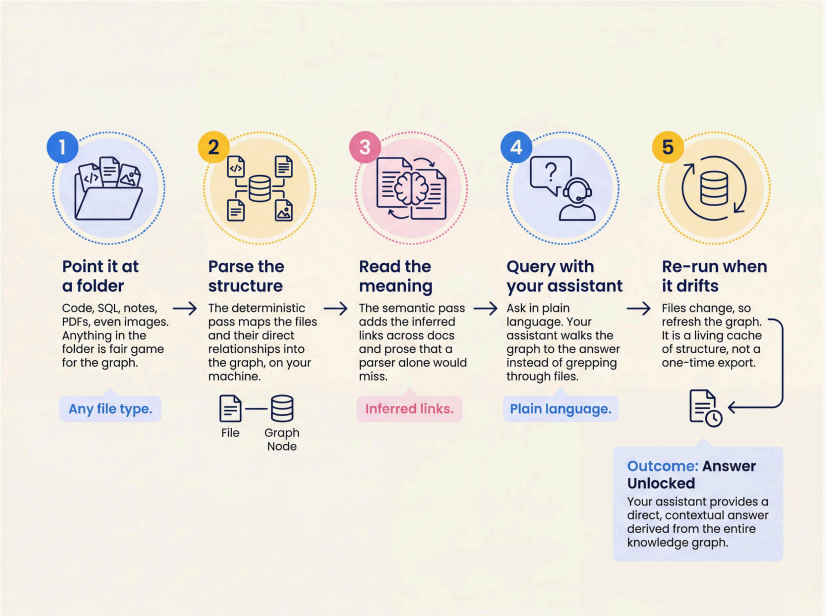
PART 4 · DO IT YOURSELF

Do it on your own folder

The whole thing runs from one command. You point it at a folder and get a queryable graph out.

What goes in. Code, SQL, notes, PDFs, even images. The parser maps the structure, the semantic pass adds the inferred links, and the two land in a single graph on your own machine.

How you use it. Ask your assistant a question in plain language. Instead of grepping the files, it walks the graph to the answer. Tools like Graphify run right inside the assistant you already use.



Living with your second brain

A second brain is only useful if it stays useful. The habits are simple.

Query it first. Before you open a folder or search your notes by hand, ask the graph. The point is to stop wading through files and start asking questions of the structure.

Keep it fresh. Files change, so the graph has to keep up. Re-run extraction on what changed, not the whole thing. The EXTRACTED edges refresh cheaply on write. The heavier inferred pass can run on a schedule.

Trust it in layers. Lean on the EXTRACTED edges outright. Treat the INFERRED ones as strong leads worth a glance. The tags tell you which is which, so you always know how much weight an answer can bear.

Structure it once. Then it works for you every day, instead of the other way around.

PART 5 · THE PROOF

We did it to our own codebase

> *We do not just teach this pattern. We run our company on it.*

The code-only version of this is called CodeGraph, and we have been running it at scale. It turns our codebase into a structural graph our agents read instead of grepping.

More than 25 repositories. Across four domains, every repo is indexed as a graph our agents query on demand. When an agent asks what calls a function and what it would break, it gets the answer from the graph in one step, with the path shown.

PART 6 · WHERE THIS GOES

From a second brain to an operating layer

> *A personal second brain is where this starts. A team's is an operating layer.*

Once your own files are structured and queryable, the same idea scales. Your tools, your data, and your agents can share one structured memory instead of each holding a slice.

The same bet, sized up. A second brain answers your questions from your files. An operating layer answers the business's questions from its systems, with a person on the gate for anything that matters. Same structure-first idea, wired into the work.

Where we fit. Omni by Enterprise DNA is where you wire your structured knowledge, your tools, and your agents into one cockpit. We built it for ourselves first, and we run on it every day.

Start where it is easy

Pick one folder you already know well. Build the graph, ask it a few real questions, and watch it walk to the answer. That is the whole idea, and it is closer than it looks.

Web: enterprisedna.co

Try it: search Graphify and point it at a folder today

ENTERPRISE DNA

A NOTE IN CLOSING

Stop giving your AI more to read. Give it structure to walk. A knowledge graph over your own files is the difference between an assistant that wades through everything and one that goes straight to the answer, and shows you the path.

FROM INSIDE THIS GUIDE.

Structure it once. Query it forever.