How ARD works

What is ARD

- ARD: Natural Language search over a set of ‘Agentic Resources’
  - MCP tools, Skills, …
  - Data sources!
  - agenticresourcediscovery.org
  - E.g., github.com/agentfinder, https://huggingface.co/blog/agentic-resource-discovery-launch
- ARD is actually a protocol, not a system

One way to implement ARD

- Catalog of ARD entries
- Each has an NL description
- Store vector embedding
- Given query, retrieve top 20/30/50 nearest neighbours
- Use a small language model to rerank these against the query (pure NL, no embeddings in this stage)
- Many different models work.
- A lot of tweaking to parallelize calls, keep cost and latency low, etc.
- Return values are a set of ‘ARD entries’

ard-finder: agent_finder.py ard-finder: registry/index.py ard-finder: docs/life-of-a-query.md

ARD + OKF

- OKF: a simple way of describing a data source.
- Data is the most important agentic resource
- Encode OKF in JSON-LD as an ARD entry and let any agent find relevant data sources.

ARD + OKF on NeuralWeb

- Endpoint with about 20k+ ARD entries for OKF data sources
- ARD finds tables, not numbers.
- One document per published table.
- Demo 1: What is the poverty rate in Detroit
  - https://ard.neuralweb.dev/?q=poverty+rate+in+detroit
- Same question at API level:
  - https://ard.neuralweb.dev/ask?query=poverty+rate+in+Detroit
- API (all endpoints):
  - https://ard.neuralweb.dev/?format=json

Source vs table

- A source is a publisher.
- A table is one variable from a source.
- Census: one source, hundreds of tables.