Name: mcp-358-cswg-workshop-a-multi-agent-ai-tool-for-machine-design
Title: Workshop: A Multi-Agent AI Tool for Machine Design
Status: Draft -- anyone can edit. To request edit access, go to
http://mcp.infrastructures.org/unlock/mcp-358

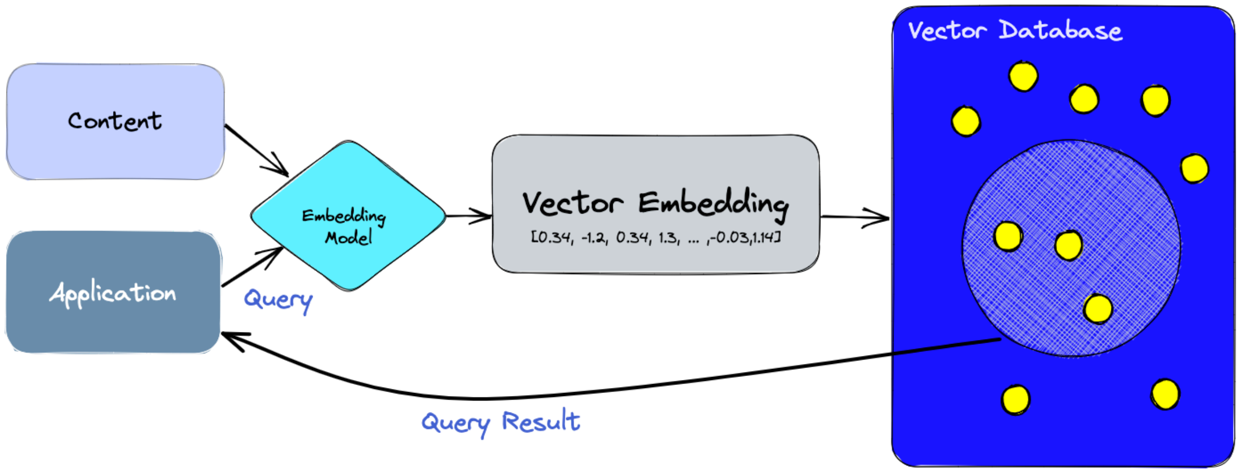
See the [MCP index](#) to create or find documents, or [mcp-0-readme](#) for an overview.
The headers above are machine-readable; please preserve format.

When: 2023-11-21 14:00 Pacific
Who: Steve, Donaldo, Richard

Workshops git repository: https://github.com/ciwg/workshops

- demo
    - XXX show pong game first next time
        - run pong_8.py
        - show pong.sh
        - discuss how org design (members and how they route messages to each other) turns out to matter (of course)
        - epiphany: consensus formation in self-organizing groups matters in multi-agent systems, whether human or AI (of course)
    - echo “foo” | grok tc
        - prints token count of “foo” on stdout
        - says “1”
    - echo “foo” | grok msg “you are a sequence continuer”
        - sends system message “you are a sequence continuer”
        - sends user message “foo”
        - prints “bar” on stdout
    - lots of conversation about how LLM servers work:
        - GPT-4 primary API, for instance, is a “chat continuation” service: you must send it a complete chat, and it adds the next message as its response
    - both grokker and ChatGPT function by maintaining local context:
        - for each query (prompt) they summarize it in a synthesized chat short enough to fit within the token limit, then send that synthesized chat to the GPT LLM backend servers
        - in the case of ChatGPT web UI, the “local context” is whatever is in the same chat session
        - in the case of grokker, the “local context” is whatever you’ve added to the grokker repository in the current directory. You add a file to local context by running:

            - “grok add filename”
        - in both cases (either ChatGPT web UI or grokker), the tool collects the text from local context that is most similar to the text in your query (prompt), and uses that collected text to synthesize the earlier messages in the synthesized chat.
            - the definition of “most similar” is normally “shortest distance between two points in an n-dimensional space”, where the two points represent the prompt text and a chunk of local context text.
                - see “embeddings”, a whole ‘nother conversation, but the short version is that any piece of text can be turned into a set of numbers that represent a point in space
            - the tool sorts by shortest distance first, and sends the chunks at the top of the list as message(s) in the synthesized chat
        - both ChatGPT and grokker maintain a vector database that contains all of the local context
            - the entire chat session in ChatGPT’s case
            - all files added in grokker’s case
        - A Beginner's Guide to Vector Embeddings - references some great visuals we can use
            - https://www.pinecone.io/learn/vector-database/



- a better multi-agent algorithm
    - fact: an “agent” is a single sequence of chat messages
        - this sequence must fit in the token limit
    - fact: any conversation must fit within the token limit
    - fact: BUT an agent’s chat message sequence can be synthesized from a much larger local context, using e.g. grokker’s vector database

        - BUT if you do this, you lose some context based on how well earlier messages get summarized
        - TANSTAAFL -- “there ain’t no such thing as a free lunch”
    - inference: one benefit of a multi-agent system is these multiple contexts
        - more agents lets you maintain more context without summarizing
    - hypothesis: a better algorithm is cooperative, promise-based, self-organizing
        - ref Mark Burgess’ Promise Theory work
- next week proposal:
    - Workshop: AI-Aided Consensus Formation in Self-Organizing Groups
- postmortem
    - a rolling slide deck for the first 10 minutes would help recap and focus
        - “rolling” as in previous topics, not a whole new slide deck every time
        - possibly using a markdown-source slide deck so it’s able to be version controlled, e.g.
            - https://remarkjs.com/
            - https://revealjs.com/
    - TODO Steve this week (two weeks more likely, considering Thanksgiving holidays/family) POC a promise-based consensus algo
        - ref Mark Burgess’ Promise Theory work
        - simple rules producing complex behavior
            - e.g. simple code producing complex behavior
        - the general idea is a bulletin board
        - plan B is a market board model
        - plan C is give up and finish implementing chatdev model
        - but if plan A works, then that dovetails better with the general goals of CSWG
            - gives us a platform in which to test consensus-formation algorithms while using AI-generated simulated human actors
        - Donaldo: devil’s advocate argument:
            - if we can move the hierarchy into the machines, then the people are freer to act independently
            - this could be true (Steve has been wondering the same thing for 40 years, but…)