Freddie Boateng

Old Dominion University

CS411W

Dr. Sumaya Sanober

Feb 28, 2025

1 Table of Contents

In today's world, various interfaces frequently don't speak the user's language in today's fast-paced digital environment, which causes a disconnect between how users interact with apps and how those apps work in reality. Users anticipate smooth, intuitive experiences and frequently using natural language. Yet developers encounter several obstacles when attempting to incorporate these features into their apps. This is where CueCode comes into play. This software will be built to make things easier for developers in terms of creating a safer, more scalable way for businesses to harness the power of NLP and LLMs, ensuring that AI applications can be developed with confidence and without unnecessary risks. CueCode will offer staff that are non-technical that give the ability to interact with APIs using simple language. Furthermore, allowing versatility for enterprises that utilize multiple systems and services

The lack of mature, risk-aware tooling for generating API payloads from natural language has created a significant barrier for developers and organizations looking to build AI-driven applications. CueCode aims to fill this gap by commoditizing the process of turning natural language into API payloads, allowing enterprises and SaaS platforms to seamlessly integrate NLP features into their systems—without sacrificing control or risking business logic integrity.

CueCode is not just about making things easier for developers it's about creating a safer, more scalable way for businesses to harness the power of NLP and LLMs, ensuring that AI applications can be developed with confidence and without unnecessary risks.

1.1 Key Product Features And Capabilities

Some of the features that CueCode will essentially do is have features such as user friendly interface (API client libraries) that developers can utilize. CueCode will offer staff that are non-

technical that give the ability to interact with APIs using simple language. Furthermore, allowing versatility for enterprises that utilize multiple systems and services.

This will increase efficiency allowing for quick and accurate REST API interactions based on simple language inputs. By abstracting the technical complexity of generating the API payloads, CueCode effectively bridges the gap between human input and technical execution of their requests via REST API calls.

1.2 Major Components (Hardware/Software)

According to the description, CueCode will need a specific hardware and software infrastructure in order to handle a variety of NLP tools and machine learning models. The software will make it easier to deploy, manage, and carry out the machine learning tasks, while the hardware will have to handle the processing demands of large-scale models. The software that will be used will be Ollama which will be responsible for running the large language model. It needs to be capable of efficient parallel processing and inference with minimal latency.

2 Identification Of Case Study

Steve's client needs to parse free-text descriptions "I need a 30-minute appointment with Dr. John Smith next Tuesday morning" and extract key details (e.g., appointment type, time, doctor's name) to create structured data for an API. He needs a reliable and scalable solution that integrates NLP and AI-driven text-to-API conversion into his full-stack application. The complexity lies in ensuring that the system can handle varied, unstructured text inputs in real time while ensuring the extraction is accurate and efficient.

Steve can leverage CueCode's NLP capabilities to achieve this functionality seamlessly. The system uses Spacy.io for text parsing, tokenization, and named entity recognition (NER). For example:

- **Spacy.io** can identify entities like dates ("next Tuesday"), names ("Dr. John Smith"), and appointment types ("30-minute appointment").
- The system then converts these entities into structured data that is ready for the API. It could output a JSON object like:

```
json
Copy code
{
  "appointment type": "30-minute",
  "doctor": "Dr. John Smith",
  "date": "2024-11-12",
  "time": "09:00"
}
```

Steve can easily integrate this workflow into his backend, leveraging CueCode's API endpoint that processes the raw text input and returns structured data. The system can be scalable and cloud-based, allowing Steve to handle large volumes of API requests from his web application without performance degradation. If Steve needs a more custom NLP model, he can use Ollama to tune a large language model and also fine-tune the data extraction further. Steve will benefit from this by saving time due to the fact that he utilized the NLP features for text-to-structured-data conversion.

Another scenario is Patricia Davis, who wants to book an appointment. Patricia could use a conversational AI interface powered by CueCode's Ollama software to interact with the hospital's appointment scheduling system. CueCode's solution would allow her to input her request in natural language, such as:

- "I need an appointment with Dr. John Smith in cardiology next Monday afternoon."

CueCode's NLP system, using Spacy.io, would recognize the key entities (e.g., doctor name, department, time), extract them, and convert the request into an API payload that the hospital's system can process.

- Example request: "Schedule appointment with Dr. John Smith next Monday afternoon."

  The system would recognize:

  - Doctor: Dr. John Smith
  - Department: Cardiology (based on context or prior input)
  - Date: Next Monday
  - Time: Afternoon

  It would then automatically generate a payload like:

```
json
Copy code
{
  "doctor": "Dr. John Smith",
  "department": "Cardiology",
  "date": "2024-11-11",
  "time": "14:00"
}
```

The system would automatically send this data to the backend, confirming the appointment without requiring Patricia to manually choose the doctor, department, or time.

For developers like Steve, it simplifies the integration of NLP-driven features into web applications. For individuals like Patricia, it provides a more efficient and user-friendly way to interact with complex systems, such as appointment booking in healthcare. The flexibility of the system also positions it for expansion into a wide range of industries in the future.

2.1 X Product Prototype Description

CueCode is a developer-oriented prototype designed to assist in converting natural language input into API requests, streamlining the process of integrating Natural Language Processing (NLP) into applications. The system will focus on transforming plain English text into structured REST API payloads, making it easier for developers to interact with APIs without needing to manually craft each request.

2.1.1 Core Features of the Prototype:

1. **Natural Language Processing (NLP) Integration**

   The prototype leverages NLP to process and interpret user input in English. Developers will input natural language queries, and CueCode will generate corresponding API requests. The system will rely on **Spacy.io** for NLP tasks, with basic functionalities to process simple to moderately complex requests.

2. **REST API Generation**

   CueCode's primary task is converting natural language into REST API payloads. It will support generating **JSON-based REST API requests**, ensuring that the API payloads are properly formatted according to the OpenAPI specifications. The prototype will not support XML or GraphQL formats.

3. **API Configuration Wizard**

   The system includes a simple configuration wizard that guides developers through setting up the API specifications. This is aimed at simplifying the process of connecting natural language inputs with specific API endpoints.

4. **Backend and Asynchronous Task Handling**

   The backend is built using the **Flask web framework** in Python, with asynchronous task processing powered by **Dramatiq**. This ensures that API requests are handled efficiently, even when multiple requests are processed simultaneously.

3 Prototype Architecture (Hardware/Software)

- **Software**
  - **Frontend**: HTML, CSS, Bootstrap 5, and JavaScript will create a responsive and user-friendly interface for developers to interact with CueCode. We have API clients like **Swagger CodeGen** which will be used to generate client code for interacting with the backend APIs.
  - **For our Application Layer**: Python and the Flask Web Framework will power the backend, while **Jinja** templating renders dynamic HTML content. Dramatiq will handle asynchronous task processing, ensuring efficient request handling.
  - Application Libraries: Spacy.io is used for natural language processing (NLP) tasks, while the Ollama Python client enables integration with other NLP tools. The OpenAPI spec validator ensures that API definitions conform to standards.
  - **Persistence Layer**: PostgreSQL stores the data, with PgVector supporting vector-based search for NLP-related tasks. pg-dramatiq is used for Postgres-based message brokering, enabling efficient background task management.

- **Hardware**
    - LLM (Large Language Models): Ollama and Llama 3.2 are used for natural language processing, enabling CueCode to interpret and generate API requests based on user input.
    - Third-Party Services: Identity services handle authentication, while transactional email services manage email communication with users (e.g., account verification, notifications).
    - **Testing**: Testing frameworks like **Jest** (for JavaScript) and **PyUnit** (for Python) ensure code quality, while **PyLint** enforces best practices by checking module boundaries.
    - CI/CD: GitHub Actions automates continuous integration and deployment, and Docker registry stores containerized application images for easy deployment.

3.1 Prototype Features and Capabilities

In real-world applications, CueCode features would be fully developed, scalable, secure, and integrated into a broader ecosystem, offering a high degree of customization and automation. On the other hand, prototype versions would be more focused on proving the concept, likely offering fewer features, less stability, and more manual processes. The goal of the prototype is to demonstrate functionality, while the real-world application aims to provide a polished, reliable, and user-friendly product.

- **Login / Authentication Real-World**:
- In production, the login/authentication system would likely include multi-factor authentication (MFA), strong encryption protocols, user role management (admin,

developer, publisher), and integration with identity management systems like OAuth or SSO (Single Sign-On).

- **Account Creation / Deletion:** Account management would be fully functional, allowing users to create, update, and delete accounts seamlessly. It would also include user verification and recovery options (e.g., password reset via email).
- **Prototype**: This may only include basic account creation with minimal features, and account deletion may not be fully automated or may be temporarily disabled during the testing phase.

1. **Config**
    - **REST API Definition Management**
    - **Real-World**: Fully-featured management for creating, editing, and maintaining REST API definitions, integrated with version control, testing tools, and proper documentation for each API version.
    - **Prototype**: Basic API definition management, possibly with limited functionality, such as only the ability to upload or modify a small subset of configurations without extensive versioning or automation.
    - **Upload and Manage OpenAPI Specifications**
    - **Real-World**: OpenAPI spec management is robust, supporting full API documentation, automatic validation, versioning, and integration with continuous integration/continuous deployment (CI/CD) systems.

- **Prototype**: Likely simpler, allowing users to upload OpenAPI specifications without comprehensive management tools or integration into a larger development pipeline.

2. REST API Configuration Wizard

- **Real-World**: Fully interactive, easy-to-use wizard with error handling, step-by-step guidance, and the ability to customize various API settings based on user needs.
- **Prototype**: Might be basic with limited functionality, providing only rudimentary guidance and lacking some customization or error-handling features.

3. Runtime

Process Natural Language and Turn it into REST API Payloads

Real-World: The NLP system would be highly refined, capable of accurately interpreting natural language input and generating complex API payloads, with minimal errors and fast response times.

Prototype: The NLP feature may be in the early stages, with a narrower scope, fewer supported inputs, and potentially less accuracy or slower performance.

Map Natural Language to Customer's Data Entities via Search or API Call

Real-World: This feature would have robust mapping capabilities, able to understand customer-specific data models, handle various data formats, and make real-time API calls for dynamic data retrieval.

Prototype: Mapping may be limited to simple examples, with less flexibility in mapping customer-specific data entities, possibly requiring manual intervention for complex cases.

4. Client Libraries

Integrate Application with CueCode's NLP API

Real-World: Client libraries would be fully documented, support multiple programming languages, and integrate seamlessly with a variety of frameworks and tools used by developers.

Prototype: Libraries may be basic, limited to one or two languages, and may lack comprehensive documentation or compatibility with multiple development environments.

5. NLP Monitoring

- **Trace, Debug, and Report on Translation Requests in CueCode**

Real-World: Fully integrated monitoring and logging system that offers detailed traceability of NLP requests, including debugging tools, error reporting, and user-friendly dashboards for real-time system performance.

Prototype: Monitoring features may be minimal, offering only basic logs or limited tracing with less detailed information, possibly only usable by developers during testing phases.

6. Marketplace

Share CueCode API Configurations with Other Users

Real-World: The marketplace would be fully developed, allowing users to share API configurations, with robust review systems, version control, and access management to ensure the security and reliability of shared resources.

Prototype: Sharing features might be basic or even absent, with limited ability for users to share configurations or a lack of user-friendly tools for collaboration.

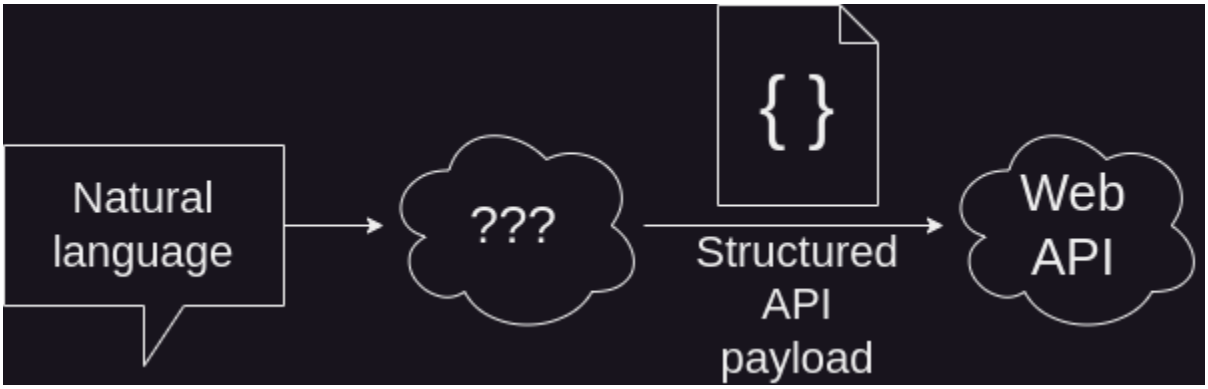
3.2 Prototype Development Challenges

The CueCode prototype faces several developmental challenges that must be addressed during its design and implementation. Accurately interpreting natural language is difficult by nature, particularly when the input has different phrasing, tone, or ambiguity. It is challenging to create an NLP system that can comprehend a variety of natural language requests and translate them into exact API payloads. Another challenge we could face are limited language support. The prototype only supports English. Expanding to multiple languages or regional dialects would require additional resources and complex localization techniques, which aren't feasible in the initial version. The prototype might have trouble managing a lot of data or processing several requests at once, especially as it combines asynchronous task handling with natural language processing.

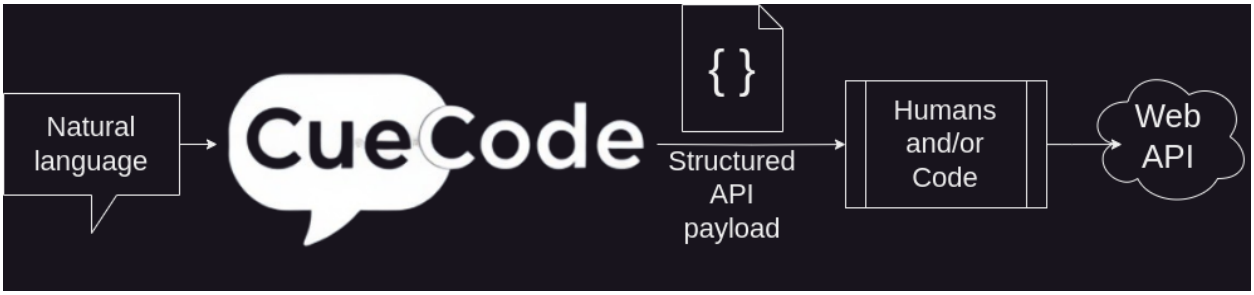
4 List Of Tables

Feature	CueCode	OpenAI Functions	Google Natural Language API	Spacy.io	LangChain	GenKit	Phone AI Alexa, Siri,...
Entity recognition	✔		✔	✔		P	✔
Plug and Play	✔				P	P	P
Retrieval Augmented Generation	✔	✔			✔	✔	
API call generation as a service	✔	P	P		P		P

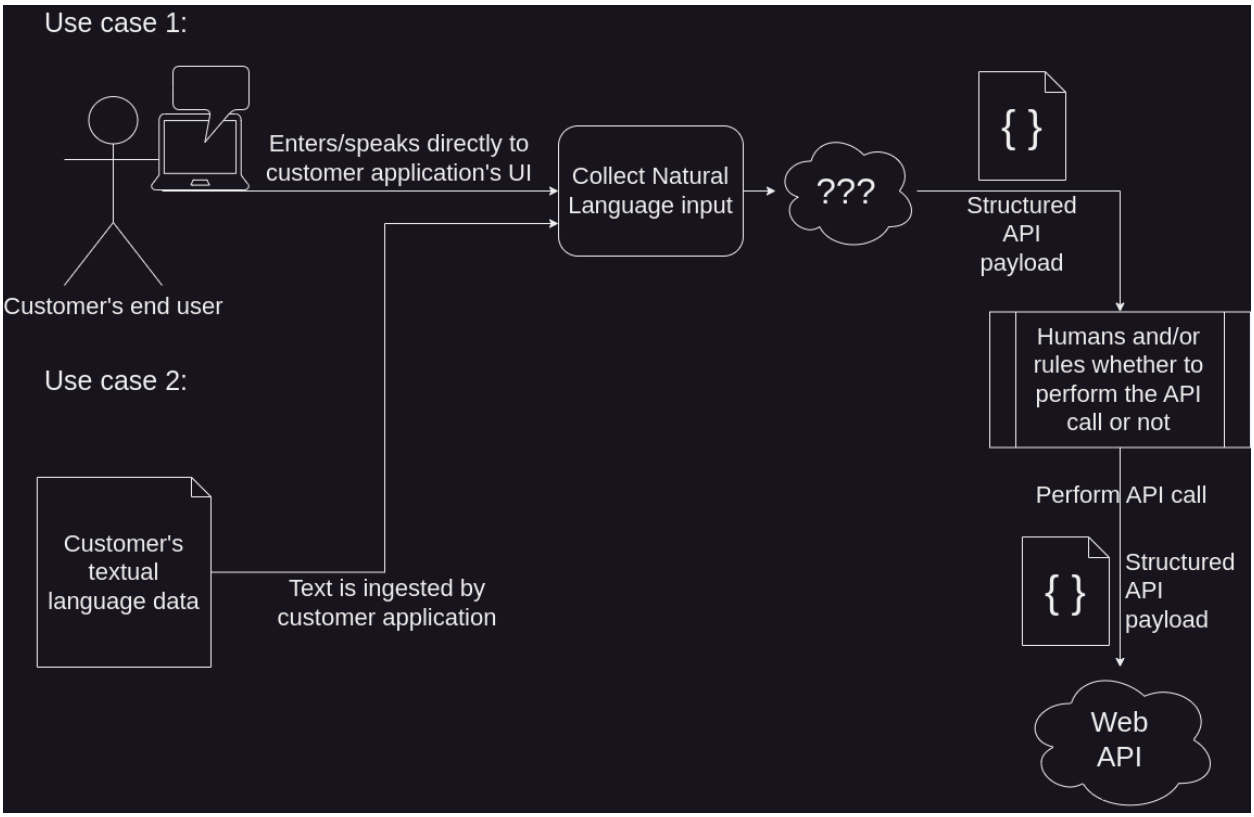
5 List Of Figures



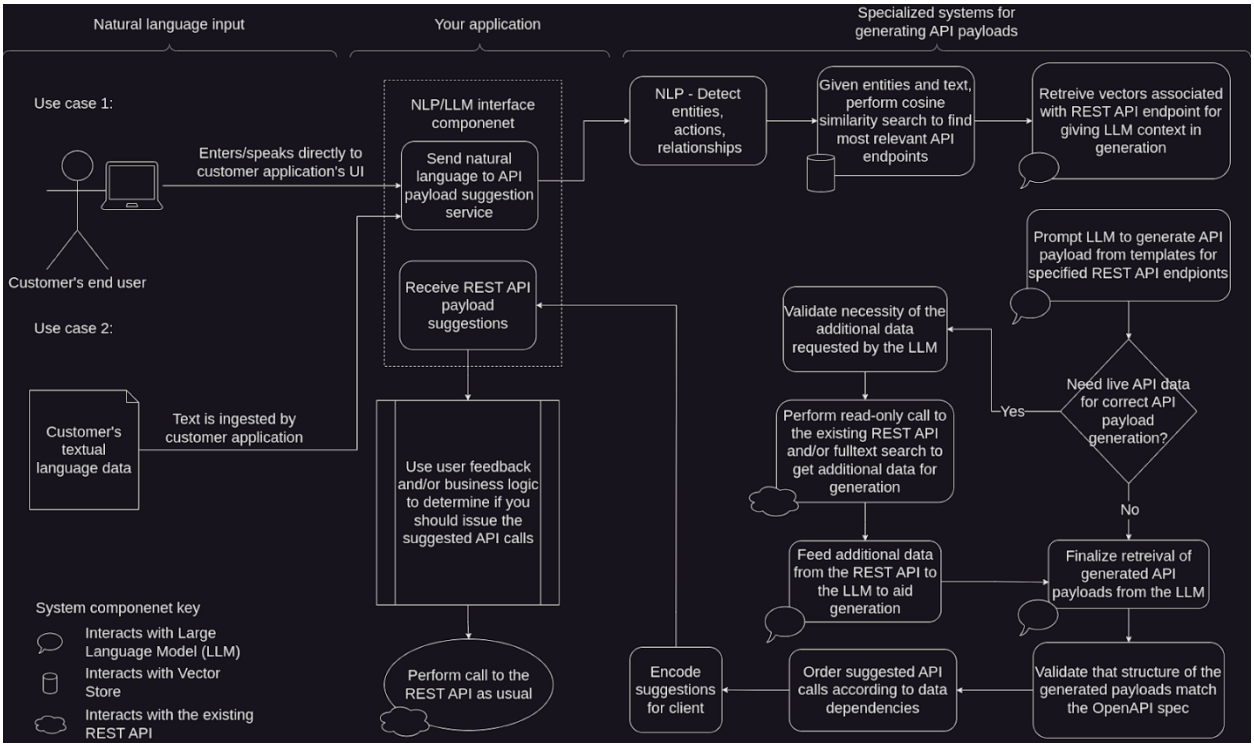
Conceptual diagram of turning natural language into Web API payloads, with the NLP component left a mystery.



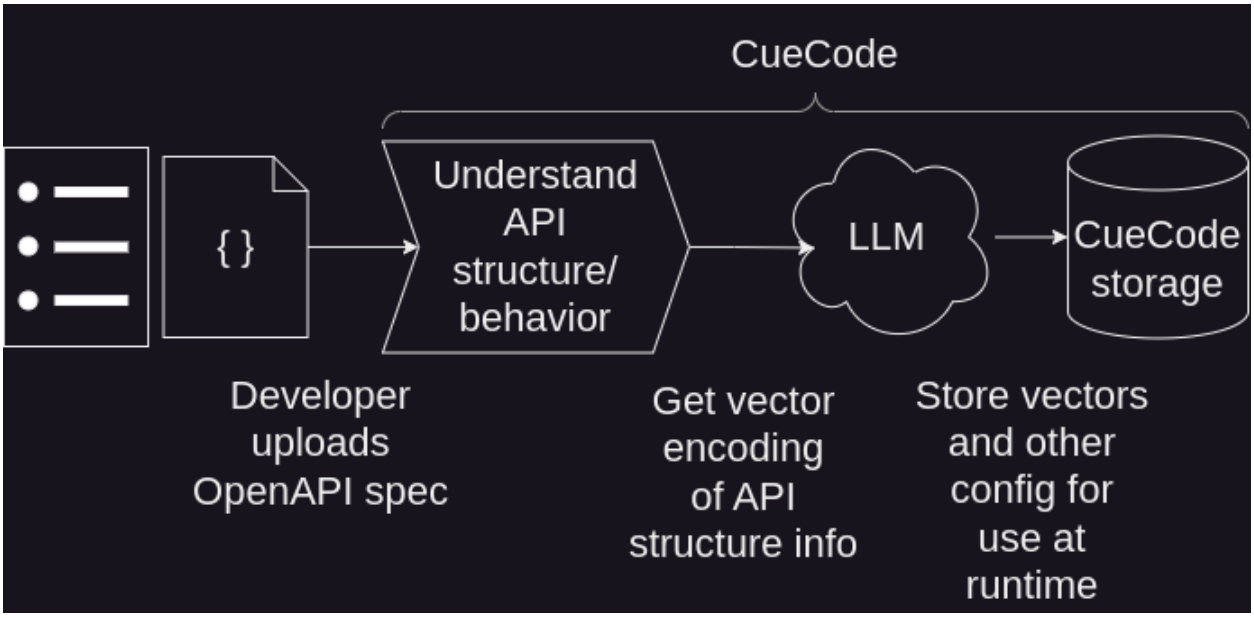
Conceptual diagram of turning natural language into Web API payloads, with CueCode shown as the NLP component.



Conceptual diagram of two example use cases where a customer can use CueCode to include validation of generated API payloads.



Current process flowchart for engineering REST API generation with OpenAPI specifications, showing the two example customer use cases from other slides. Major system components involved at each process step are labeled with icons.



6 Conclusion

CueCode provides the right application of developer tooling and service delivery to elevate your application's user experience from blunt form entry to a natural and delightful interaction. This project presents the opportunity to work in an emerging set of technologies. I hope this presentation has been informative and whets your appetite to join the project. Thank you for your time and attention.

7 Reference Page

Pgvector, pgvector. (2024). [C]. pgvector. *https://github.com/pgvector/pgvector* *(Original work published 2021)*

Prabhakaran, S. (2018, October 22). Cosine Similarity - Understanding the math and how it works? (With python). Machine Learning Plus. *https://www.machinelearningplus.com/nlp/cosine-similarity/*

Rao, P. (2024, January 24). Turbo-charge your spaCy NLP pipeline. Medium. *https://towardsdatascience.com/turbo-charge-your-spacy-nlp-pipeline-551435b664ad*

Prabhakaran, S. (2018, October 22). Cosine Similarity - Understanding the math and how it works? (With python). *Machine Learning Plus*. https://www.machinelearningplus.com/nlp/cosine-similarity/

Rao, P. (2024, January 24). *Turbo-charge your spaCy NLP pipeline*. Medium. https://towardsdatascience.com/turbo-charge-your-spacy-nlp-pipeline-551435b664ad

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models* (arXiv:2201.11903). arXiv. http://arxiv.org/abs/2201.11903

What Is NLP (Natural Language Processing)? | IBM. (2021, September 23). https://www.ibm.com/topics/natural-language-processing

Why Visual Studio Code? (n.d.). Retrieved October 22, 2024, from https://code.visualstudio.com/docs/editor/whyvscode