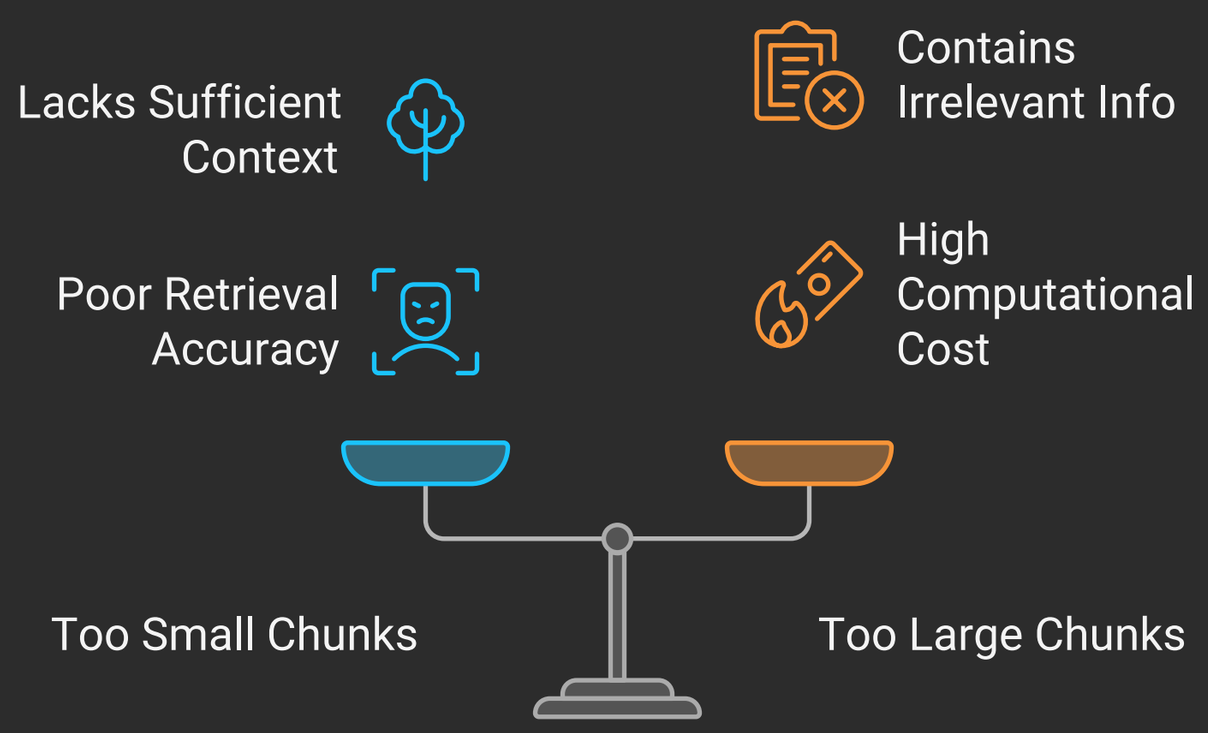
Building a Question Answering System with Vector Databases and LLMs

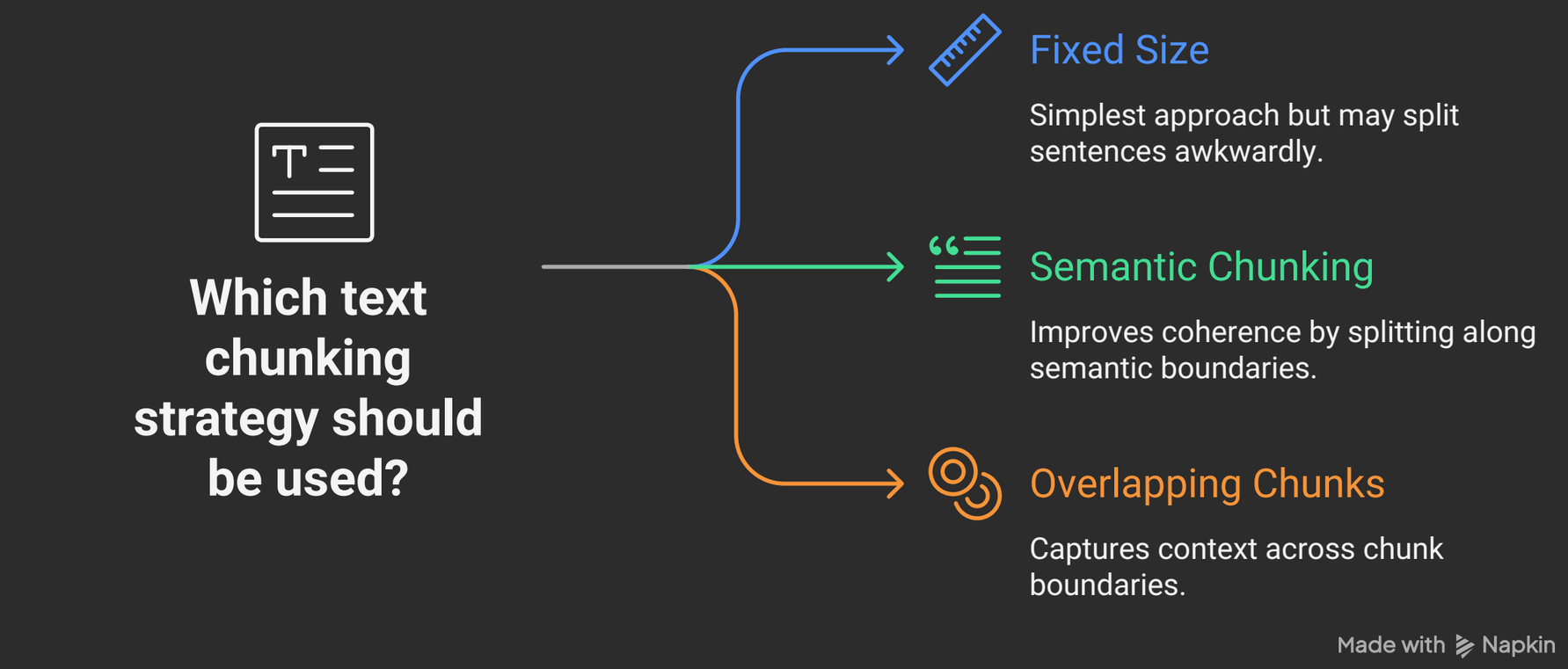
This process of building a question answering system that leverages vector databases and large language models (LLMs).
The system works by chunking a large text corpus, embedding these chunks into a vector space, storing the embeddings in a vector database like Pinecone, retrieving relevant chunks based on a user query, and finally, using an LLM to generate an answer based on the retrieved context.

1. Text Chunking

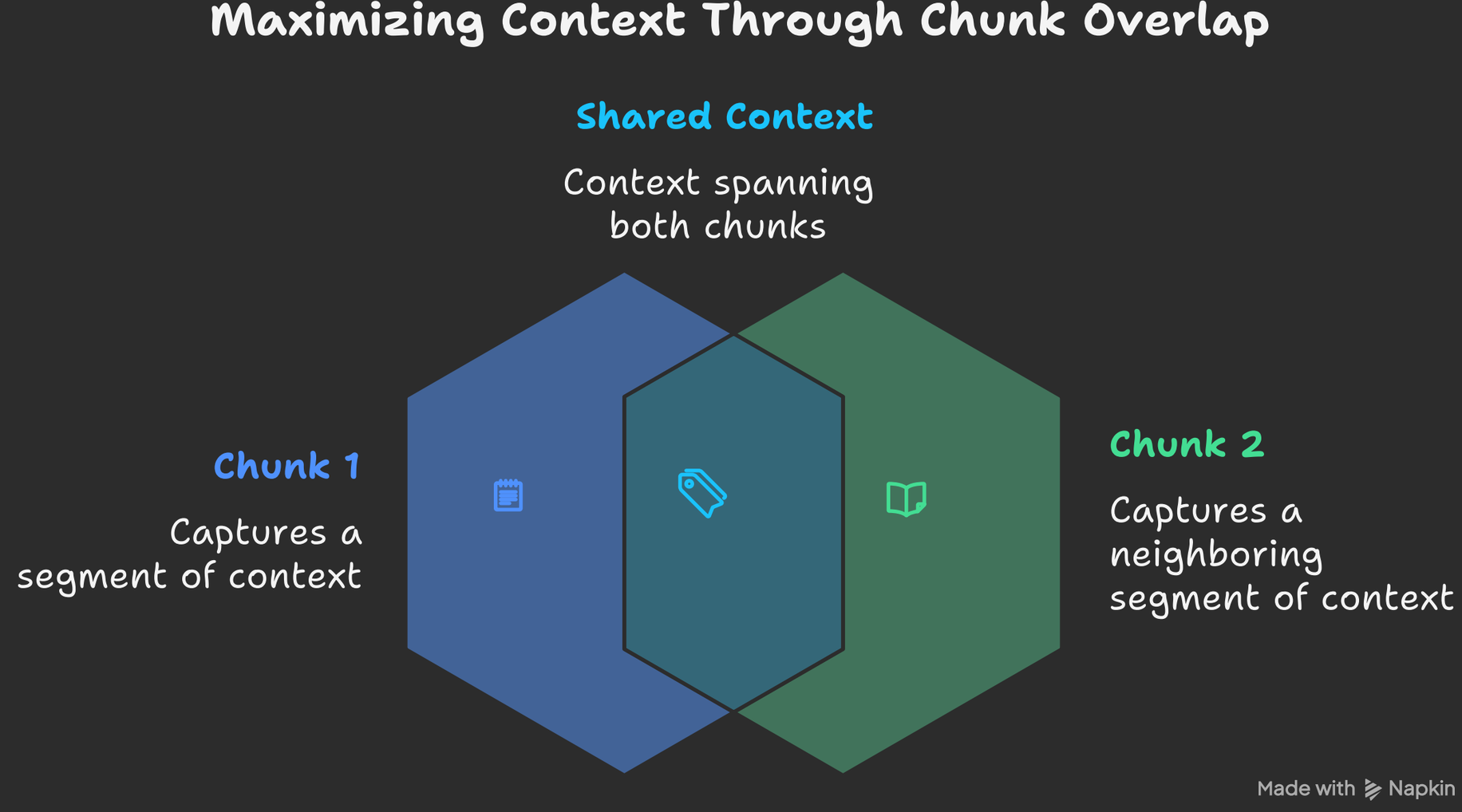
The first step is to divide the large text corpus into smaller, manageable chunks. The size of these chunks is a crucial parameter that affects both the embedding quality and the retrieval performance.



Chunking Strategies:



- **Fixed Size:** Divide the text into chunks of a fixed number of words or characters. This is the simplest approach but might split sentences or paragraphs awkwardly.
- **Semantic Chunking:** Attempt to split the text along semantic boundaries, such as paragraphs, sections, or chapters. This can improve the coherence of the chunks. Libraries like nltk or spaCy can be used for sentence splitting and paragraph detection.
- **Overlapping Chunks:** Create chunks that overlap with each other. This can help to capture context that spans across chunk boundaries.



2. Embedding Chunks

Once the text is chunked, each chunk needs to be converted into a vector representation (embedding). These embeddings capture the semantic meaning of the text and allow for efficient similarity search.

Embedding Models:

- **OpenAI Embeddings:** OpenAI provides embedding APIs that can be used to generate high-quality embeddings. These are generally very performant but require an OpenAI API key and incur costs.
- **Other Embedding Models:** Other options include models from Hugging Face Transformers or custom-trained embeddings.

3. Storing Embeddings in a Vector Database

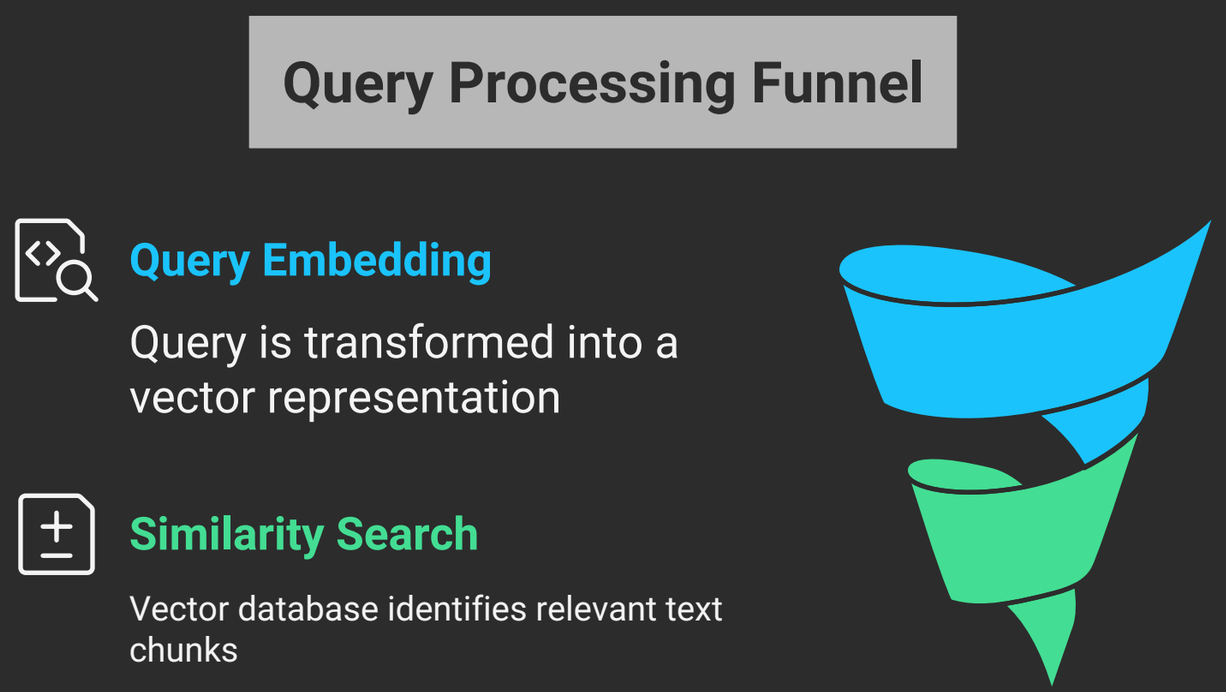
The generated embeddings are then stored in a vector database. Vector databases are designed for efficient similarity search, allowing you to quickly find the chunks that are most relevant to a given query.

Vector Database Options:

- **Pinecone:** A fully managed vector database service. It offers excellent performance and scalability.
- **Weaviate:** An open-source vector database that can be self-hosted or used as a managed service.
- **FAISS:** A library developed by Facebook AI Research for efficient similarity search. It can be used to build a custom vector database.

4. Retrieving Relevant Chunks

When a user submits a query, the system needs to retrieve the chunks that are most relevant to the query. This is done by embedding the query using the same embedding model used for the text chunks and then performing a similarity search in the vector database.



5. Generating Answers with an LLM

Finally, the retrieved chunks are passed to an LLM, along with the user query, to generate an answer. The LLM uses the retrieved context to provide a more informed and accurate response.

LLM Options:

- **OpenAI GPT Models:** Powerful language models that can generate high-quality text. Requires an OpenAI API key.
- **Hugging Face Transformers:** A library that provides access to a wide range of pre-trained language models.
- **Other LLMs:** Other options include models from Google AI, Cohere, and AI21 Labs.

Overview

- Gather raw documents (txt, PDF, HTML, etc.).
- Load + clean text (remove headers/footers, normalize whitespace).
- Chunk text into manageable pieces (token- or char-based).
- Convert each chunk → embedding vector (embedding model).
- Create a vector index (Pinecone / Chroma / Weaviate / Milvus) and **upsert** vectors with metadata.
- At query time: embed the query → search top-K vectors → pass retrieved text to an LLM with a prompt template (context + question) → produce answer.
- Iterate: tune chunk size/overlap, top_k, re-ranking, prompt, and caching.