Links to essential technical resources

Steps generally being followed for processing web data in sequence:

1. Get web data
2. Convert into markdown
3. Clean markdown if needed
4. Chunk markdown data
5. Embed data
6. Prepare schema in vector database
7. Prepare metadata in sync with database schema
8. Store metadata and embedded data.

Working with Markdown and Python

https://www.linode.com/docs/guides/how-to-use-python-markdown-to-convert-markdown-to-html/

https://www.honeybadger.io/blog/python-markdown/

Chunking

https://www.linkedin.com/pulse/harnessing-vector-databases-chunking-strategies-ai-robyn-le-sueur-sqesf/

https://medium.com/@zilliz_learn/pandas-dataframe-chunking-and-vectorizing-with-milvus-90e72b9baea5

https://zilliz.com/learn/beginner-guide-to-website-chunking-and-embedding-for-your-genai-applications

https://successive.tech/blog/rag-models-optimizing-text-input-chunking-splitting-strategies/

Embedding

https://www.turing.com/kb/guide-on-word-embeddings-in-nlp

https://www.deepset.ai/blog/the-beginners-guide-to-text-embeddings

https://sbert.net/

Milvus

https://dotcommagazine.com/2024/04/milvus-a-comprehensive-guide/

https://milvus.io/docs/schema-hands-on.md

https://help.tessell.com/en/articles/8462644-how-can-documents-be-stored-in-milvus

https://milvus.io/docs/index-explained.md

Transformers Background and Working

Evolution of Statistical Model to NLP to GenAI – A must before jump to Transformer based models.

The Illustrated Transformer

RAG

https://humanloop.com/blog/rag-architectures

https://www.deepset.ai/blog/customizing-rag

Choosing between LlamaIndex and Langchain

LlamaIndex focusses on RAG based applications and specialise in preprocessing content for RAG based applications. So, preprocessing can be done with LlamaIndex and agents can be developed using langchain

https://datasciencedojo.com/blog/llamaindex-vs-langchain/

https://www.f22labs.com/blogs/langchain-vs-llamaindex-detailed-comparison-guide/

Feature	LangChain	LlamaIndex
Primary Focus	General-purpose LLM workflows (chatbots, agents, etc.)	Optimized for RAG and document retrieval
Ease of Setup	Simple for basic RAG (Example)	More streamlined for advanced RAG pipelines (Example)

Feature	LangChain	LlamaIndex
Performance	Good for hybrid search and multi-modal workflows	Faster for hierarchical document indexing/querying (e.g., PDFs, research papers)
Customization	Highly flexible (supports custom chains, tools, agents)	Specialized for document structuring (e.g., summaries, graphs)
Community Support	Larger ecosystem (1000+ integrations)	Growing focus on RAG optimizations

When to Choose LangChain

1. **Multi-Step Workflows**:

Use LangChain if you need complex pipelines (e.g., RAG → agent → external API calls).

2. **Hybrid Search**:

LangChain supports Milvus's hybrid search (vector + keyword/BERT) natively.

3. **Multi-Modal Use Cases**:

Better for applications combining text, images, and structured data.

When to Choose LlamaIndex

1. **Structured Document Retrieval**:

LlamaIndex excels at parsing complex documents (e.g., PDFs with tables) and creating hierarchical indexes.

2. **Optimized RAG Performance**:

Benchmarks show **~15% faster query latency** for text-heavy corpora due to advanced chunking and re-ranking.

3. **Pre-Built Data Connectors**:

Built-in support for 100+ data sources (Slack, Notion, etc.) simplifies ingestion.