Indexing inside github-semantic-search

Compiled by D.Gueorguiev, from Damien Beneviste’s lecture material, 10/5/2025

Indexing happens in the backend, in module [backend/app/indexing/indexer.py](https://github.com/dimitarpg13/github-semantic-search/blob/main/backend-orig/app/indexing/indexer.py).

We start with some parsed data ( code, markdown ) and we need to derive a vector of it.

The vector representation needs to be captured by some metric e.g. cosine similarity, Euclidean distance, norm, etc.

A diagram of a keyword and a keyword

AI-generated content may be incorrect.

Figure : depicted are the 3 possible ways to index the data

The first way to project the parsed code into a vector representation is via dense vector representation. For obtaining this dense vector representation of the code fragment we are going to use LLM.

Besides using dense representation for vector embeddings we can use sparse representations which are useful in capturing specific tokens. Thus, the sparse representation is used to augment the dense representation of vector embeddings.

A diagram of a fashion model

AI-generated content may be incorrect.

Figure : Vector Representation with causal language models

# References

[1] [Dense and Sparse Embeddings: A Comprehensive Overview, M. Lokhandwala, Aug 30, 2024](https://mlokhandwalas.medium.com/dense-and-sparse-embeddings-a-comprehensive-overview-c5f6473ee9d0)

[2]

[3]

# Appendix

## Sparse vs Dense Representations of Vector Embeddings

Sparse embeddings are a type of embedding where the majority of values in the vector are zeros. These embeddings are generally high-dimensional, with most dimensions inactive or zero. Sparse embeddings were among the earliest forms of word representation in NLP, seen in techniques like one-hot encoding and bag-of-words models.

Key Characteristics of Sparse Embeddings

-High Dimensionality: sparse embeddings involve vectors with thousands and even millions of dimensions. For example