Бараненков Сергей
Демочкина Полина
Тихобаев Илья

ОРКЕСТРАЦИЯ ВЕКТОРНОЙ БД ДЛЯ ПРИМЕНЕНИЯ В LLM МОДЕЛЯХ.

AGENDA

01 BACKGROUND

02 GOAL

03 PIPELINE

04 FRAMEWORK

05 DATASET

06 RESARCH

BACKGROUND

BACKGROUND

FAQ DataBase

Context

BACKGROUND

FAQ DataBase

Context

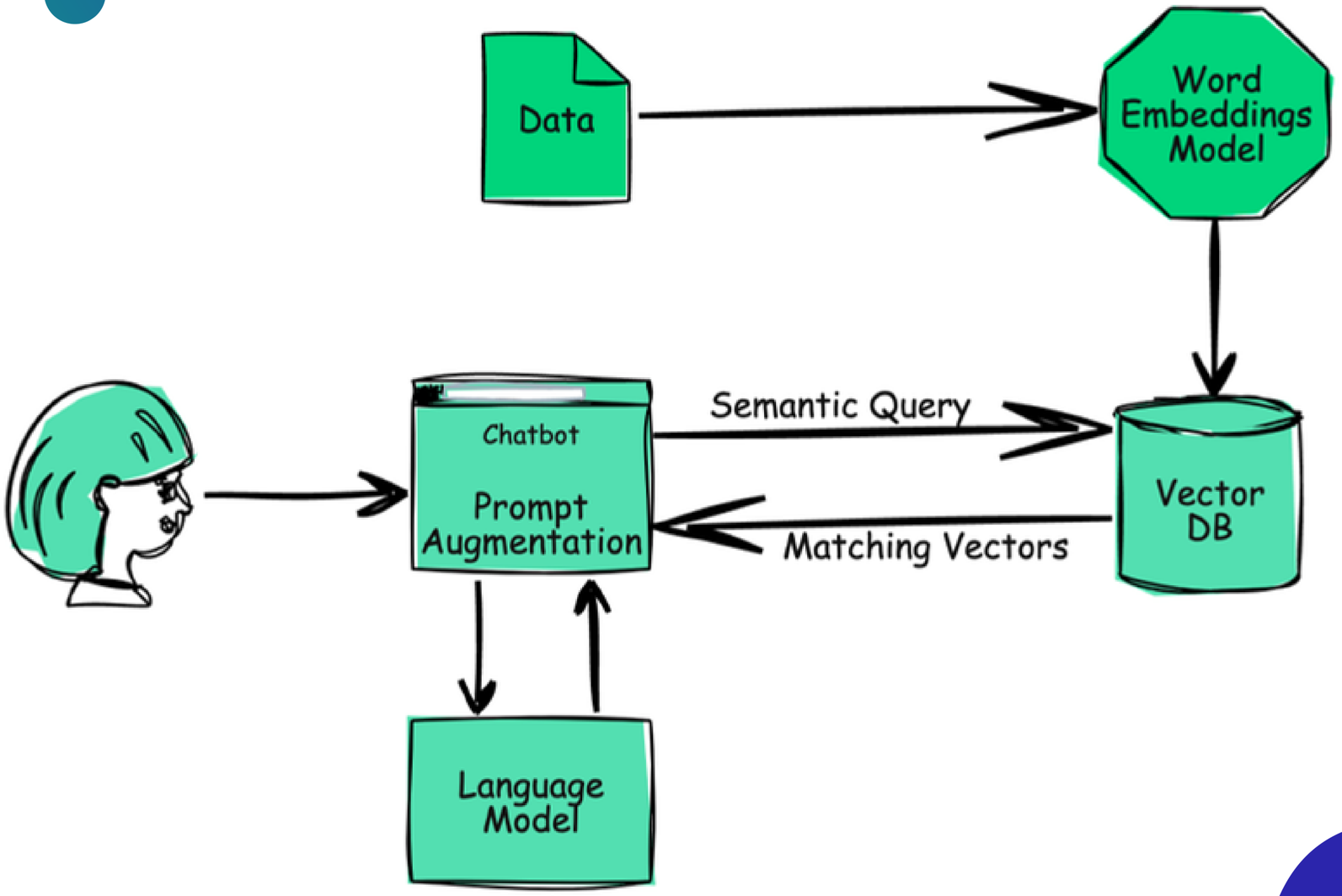
Semantic search

GOAL

Develop a chat-bot for answering Ubuntu-related questions based on a vector DB and an LLM model.

- The bot is hosted in Telegram
- ChatGPT is used to generate the final response
- Additional context is used during generation from a vector DB

PIPELINE


Data
Word Embeddings Model
Chatbot
Prompt Augmentation
Semantic Query
Matching Vectors
Vector DB
Language Model

FRAMEWORKS

- Semantic search based on context vectors
- Pipeline assembly

FAISS

elastic

LangChain

DATASET

	Context
A	Hi, I can not longer access the graphical **login screen** on ubuntu 12.04
B	what exactly happen?
A	I can't remember the error message, would it have auto-logged to a file or should I reboot quick?
B	you mean it won't **automaticaly start** and what happen then?
A	it just stop at a text **screen**, but I can access the command line **login** via alt F1-6, and **start** x **manually** there. I think it might me **lightdm** that's break but I'm not sure
	Candidate responses
R1	for me **lightdm** often won't start **automatically** either. It show me console tty1 instead and I have to **start lightdm manually** ✓
R2	what about sources.list ? ✗

# dialogues (human-human)	930,000
# utterances (in total)	7,100,000
# words (in total)	100,000,000
Min. # turns per dialogue	3
Avg. # turns per dialogue	7.71
Avg. # words per utterance	10.34
Median conversation length (min)	6

Ubuntu Dialogue Corpus

26 million turns from natural two-person dialogues

k kaggle.com

DATASET

folder	The folder that a dialogue comes from. Each file contains dialogues from one folder .
dialogueID	An ID number for a specific dialogue. Dialogue ID's are reused across folders.
date	A timestamp of the time this line of dialogue was sent.
from	The user who sent that line of dialogue.
to	The user to whom they were replying. On the first turn of a dialogue, this field is blank.
text	The text of that turn of dialogue, separated by double quotes (“). Line breaks (\n) have been removed.

Get access to the chatGPT plugin

Set up a vector DB and create the pipeline using LangChain

Test the semantic search feature in the vector DB

Create a Telegram bot to demonstrate the final pipeline

DOMAIN RESEARCH AND OVERVIEW

DOCUMENT VECTORIZATION - AVERAGING WORD EMBEDDINGS

Take the average of word embeddings within a sentence.

+ Simple
- It may not capture complex contextual nuances.

DOCUMENT VECTORIZATION - PRE-TRAINED MODELS LIKE BERT

+ BERT-based models consider the context of each word in a sentence, resulting in rich and contextually aware embeddings.

DOCUMENT VECTORIZATION - NEURAL NETWORK-BASED APPROACHES

Skip-Thought vectors and InferSent are examples of neural network-based sentence embedding models.

+ They are trained to predict the surrounding sentences, which encourages them to understand sentence semantics.

SENTENCE-BERT

- Attention — allows the algorithm to create the embedding by focusing only on the most important parts of the input.
- Transformers — takes less time to train.
- BERT — it is composed of Transformer blocks.
- Siamese Network — trained to compare the similarity between two inputs.

Reimers, Nils and Iryna Gurevych. “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.” Conference on Empirical Methods in Natural Language Processing (2019).

INFERSENT

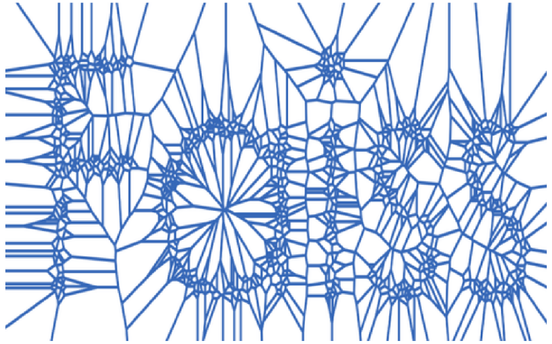
- It is trained on a large-scale supervised Natural Language Inference (NLI) task, making it suitable for tasks like paraphrase detection, semantic similarity, and sentiment analysis.
- InferSent may struggle with domain-specific data due to the nature of its training data.

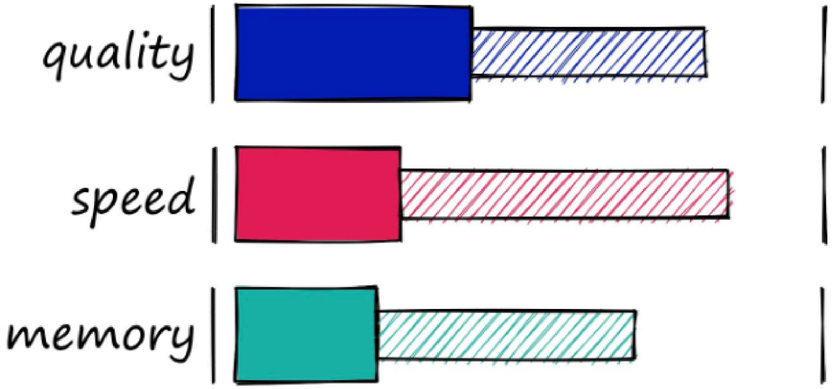
Alexis Conneau et al, “Supervised Learning of Universal Sentence Representations from Natural Language Inference Data.” In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 670–680.

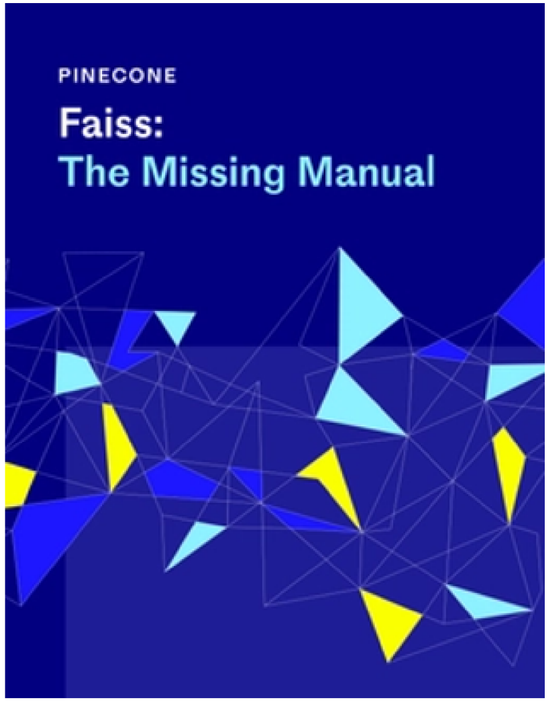
TYPES OF VECTOR DB

FAISS

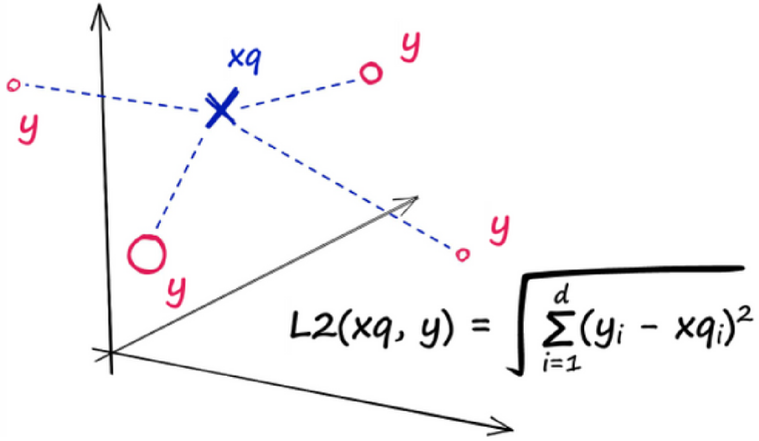
- Flat
- Inverted File
- Hierarchical Navigable Small Worlds (HNSW)

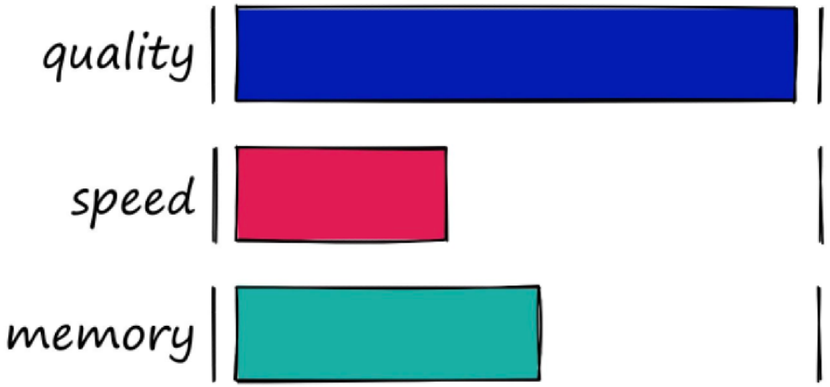


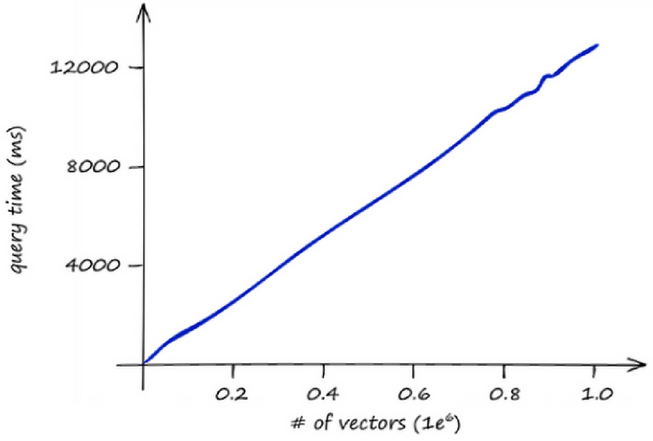




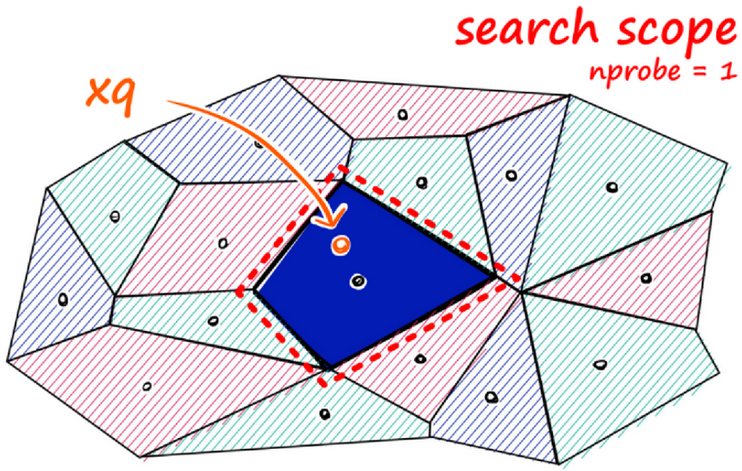
FLAT INDEX

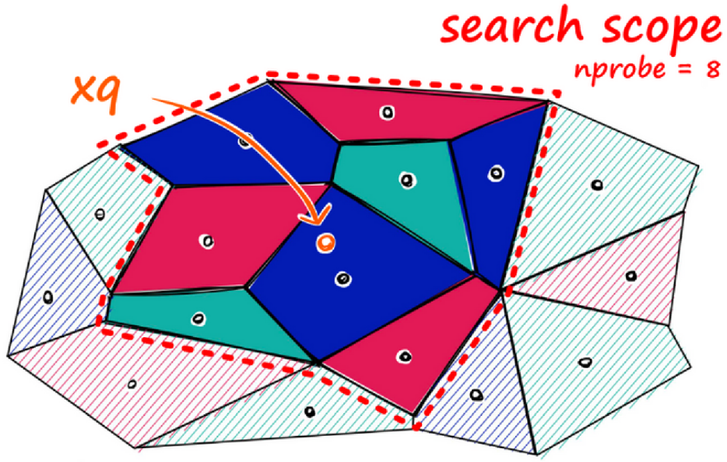

xq
y
y
y
y
$L2(xq, y) = \sqrt{\sum_{i=1}^{d}(y_i - xq_i)^2}$

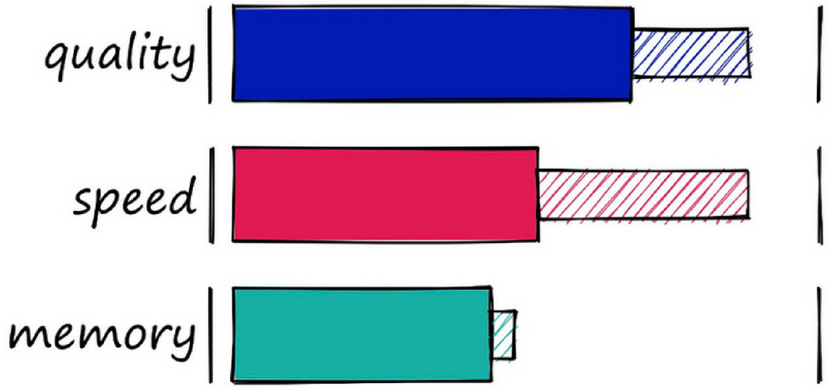

quality
speed
memory


12000
8000
4000
query time (ms)
0.2
0.4
0.6
0.8
1.0
# of vectors (1e⁶)

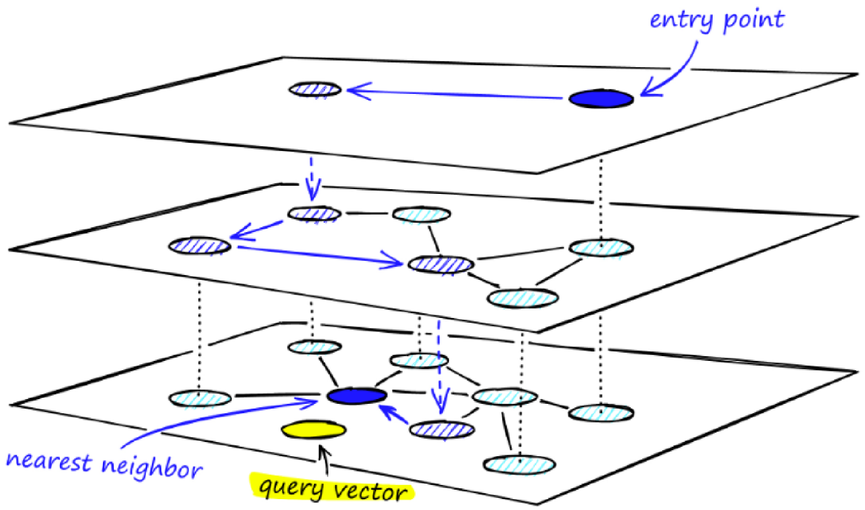
INVERTED FILE INDEX

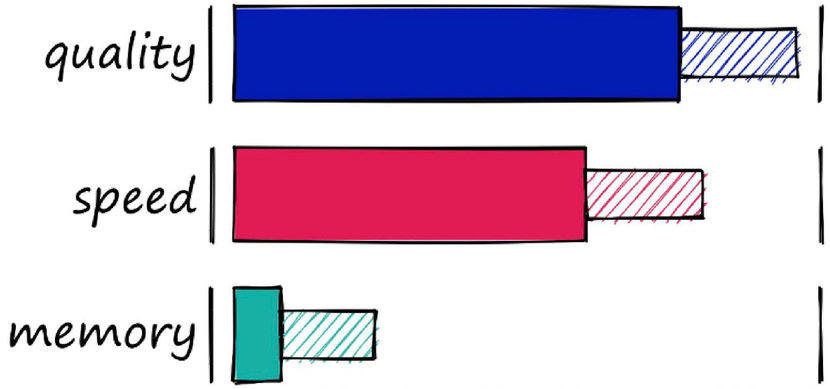


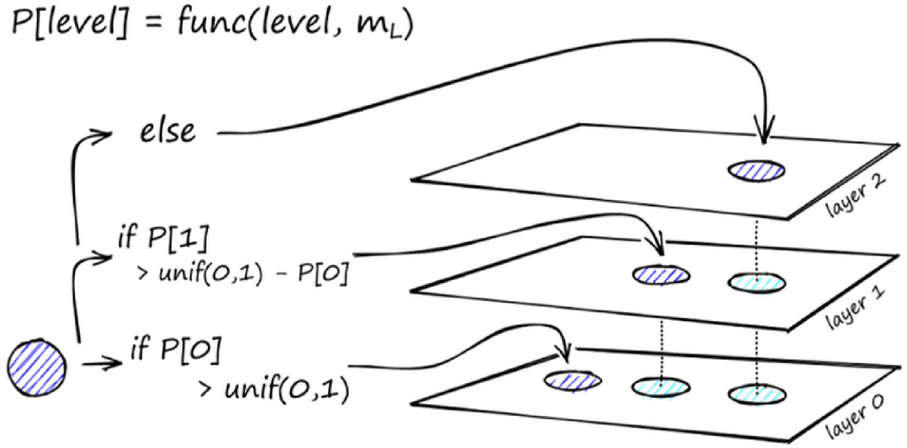




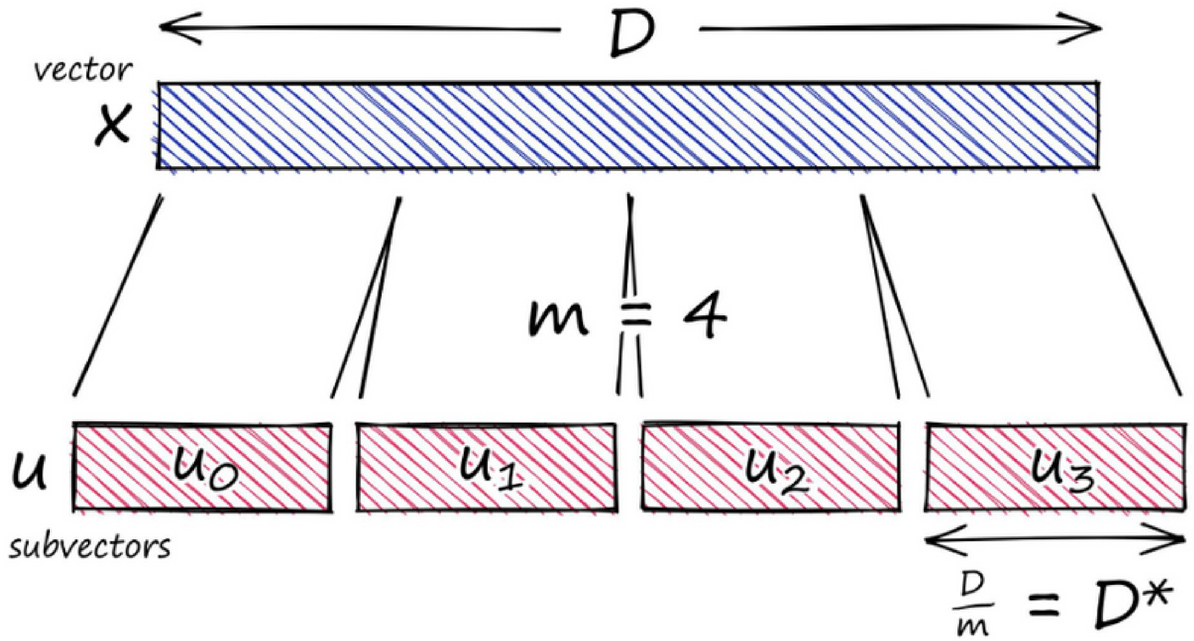
HNSW INDEX

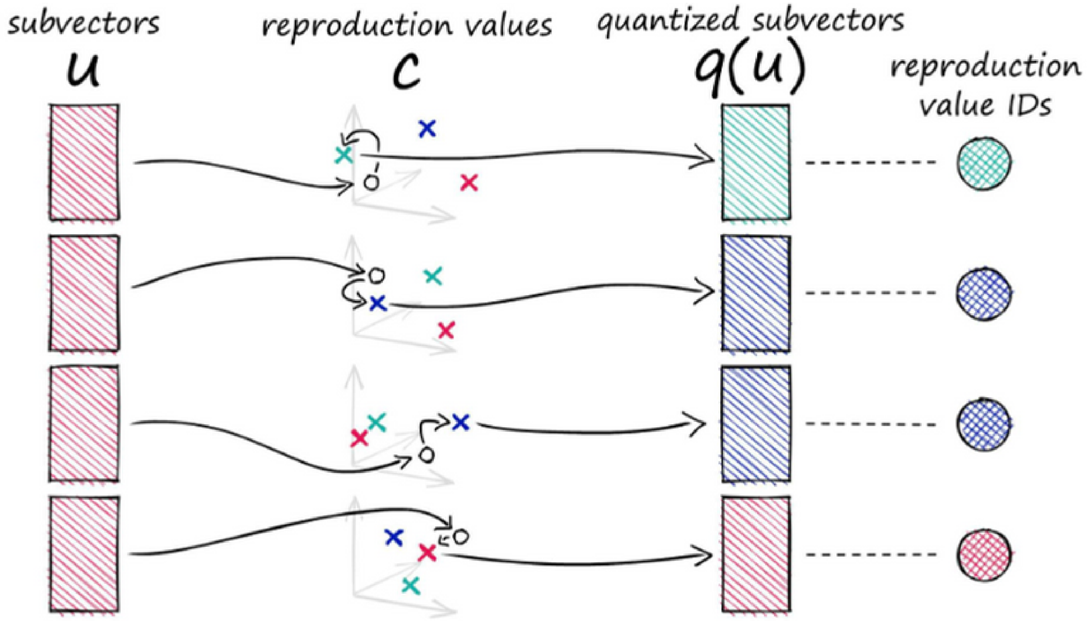

entry point
nearest neighbor
query vector


quality
speed
memory

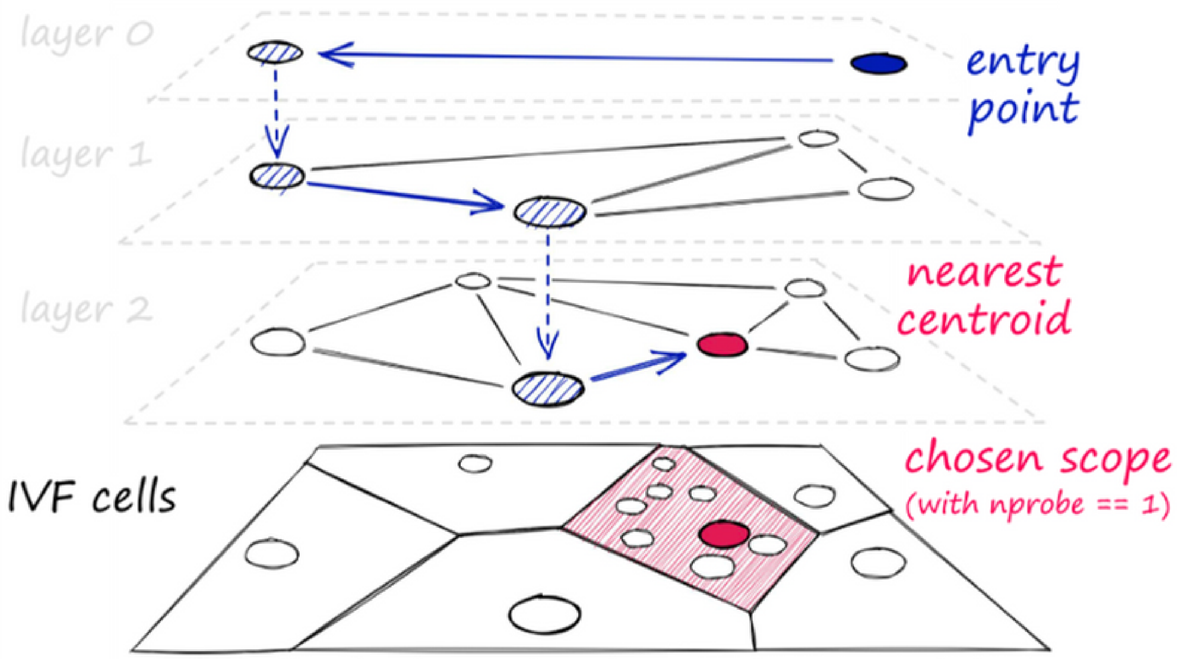

P[level] = func(level, m_L)
else
if P[1]
> unif(0,1) - P[0]
if P[0]
> unif(0,1)
layer 2
layer 1
layer 0

PRODUCT QUANTIZATION





COMPOSITE INDICES



Index	Recall	Search Time	Memory
IVF4096_HNSW,Flat	90%	550µs	523MB
IVF4096_HNSW,PQ32 (PQ)	69%	550µs	43MB
OPQ32,IVF4096_HNSW,PQ32 (OPQ)	74%	364µs	43MB

REFERENCES

- Y. Malkov et al., Approximate Nearest Neighbor Search Small World Approach (2011), International Conference on Information and Communication Technologies & Applications
- Y. Malkov et al., Scalable Distributed Algorithm for Approximate Nearest Neighbor Search Problem in High Dimensional General Metric Spaces (2012), Similarity Search and Applications, pp. 132-147
- Y. Malkov, D. Yashunin, Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs (2016), IEEE Transactions on Pattern Analysis and Machine Intelligence
- J.Briggs, The Missing Manual, https://www.pinecone.io/learn/series/faiss/hnsw/
- Faiss, https://github.com/facebookresearch/faiss
- FAISS: Быстрый поиск лиц и клонов на многомиллионных данных, https://habr.com/ru/companies/okkamgroup/articles/509204/

LangChain

Preparation:

- Split text on fragments of valid length (512-2048 tokens). Langchain has a convenient method `RecursiveCharacterTextSplitter`.
- Create embeddings from these fragments.
- Save embeddings to vector DB.

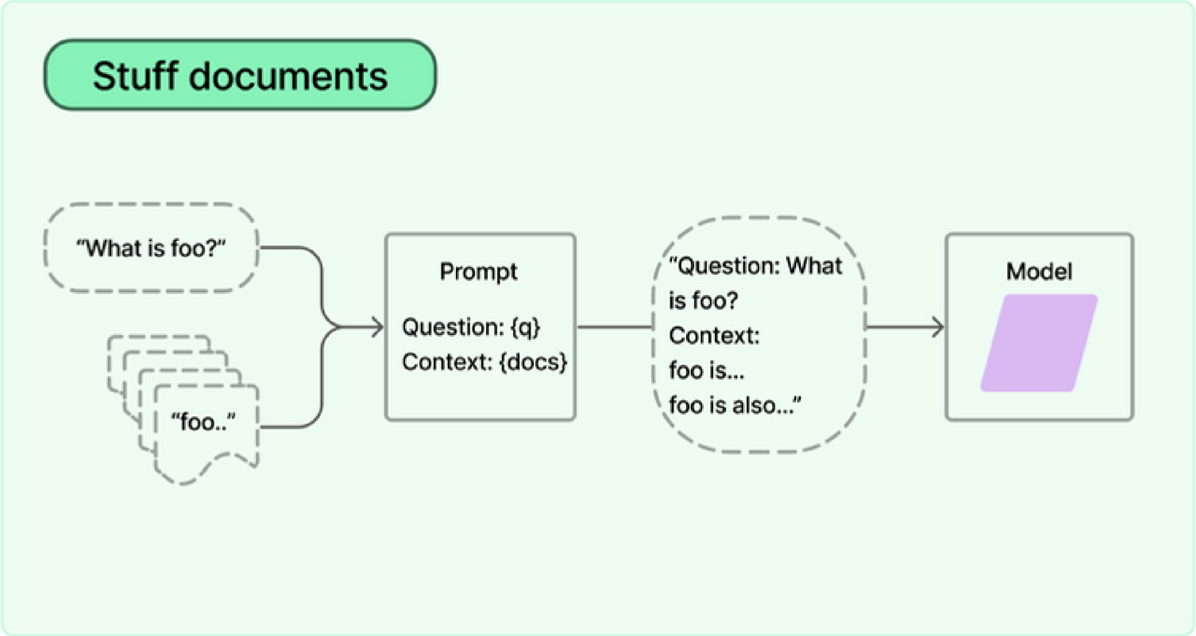
LangChain

What needs to be implemented:

- LangChain adapter for YandexGPT.
- Implement Retrieval-Augmented Generation.
- Convenient API which will just obtain questions and return answers.

RETRIEVAL-AUGMENTED GENERATION IMPLEMENTATION

We will use chains and StuffDocumentsChain method. It takes the most relevant documents retrieved from vector DB, concatenate them and save it to a variable (variable ‘context’ in our example).
Then another chain use this variable and template to create prompt and pass it to the LLM.



RETRIEVAL-AUGMENTED GENERATION IMPLEMENTATION

Example of how a prompt is generated. **Context** – found relevant text fragments. **Query** – user's question.

```
document_variable_name = "context"
template = """
Пожалуйста, посмотри на текст ниже и ответь на вопрос, используя
информацию из этого текста.
Текст:
-----
{context}
-----
Вопрос:
{query}"""
prompt = langchain.prompts.PromptTemplate(
    template=template, input_variables=["context", "query"])
```