Domain Specific AI

SD 18

Jacob Duba, Conor O'Shea, Carter Cutsforth, Diego Perez, and Keenan Jacobs

Improving Code Search

Problem

- “I know what I want the code to do, but don’t know the specific keywords”
- Natural language keyword search is an unsolved problem
- Now, LMs embedded in programming environments (eg. Cursor) need to search a local codebase for more context.

Solution

- UniXcoder, an embedding model, scores 70% on the CosQA benchmark.
- Add more compute
  - Take top 10 results from UniXcoder, and rank them using a language model.

Vector comparison done more efficiently

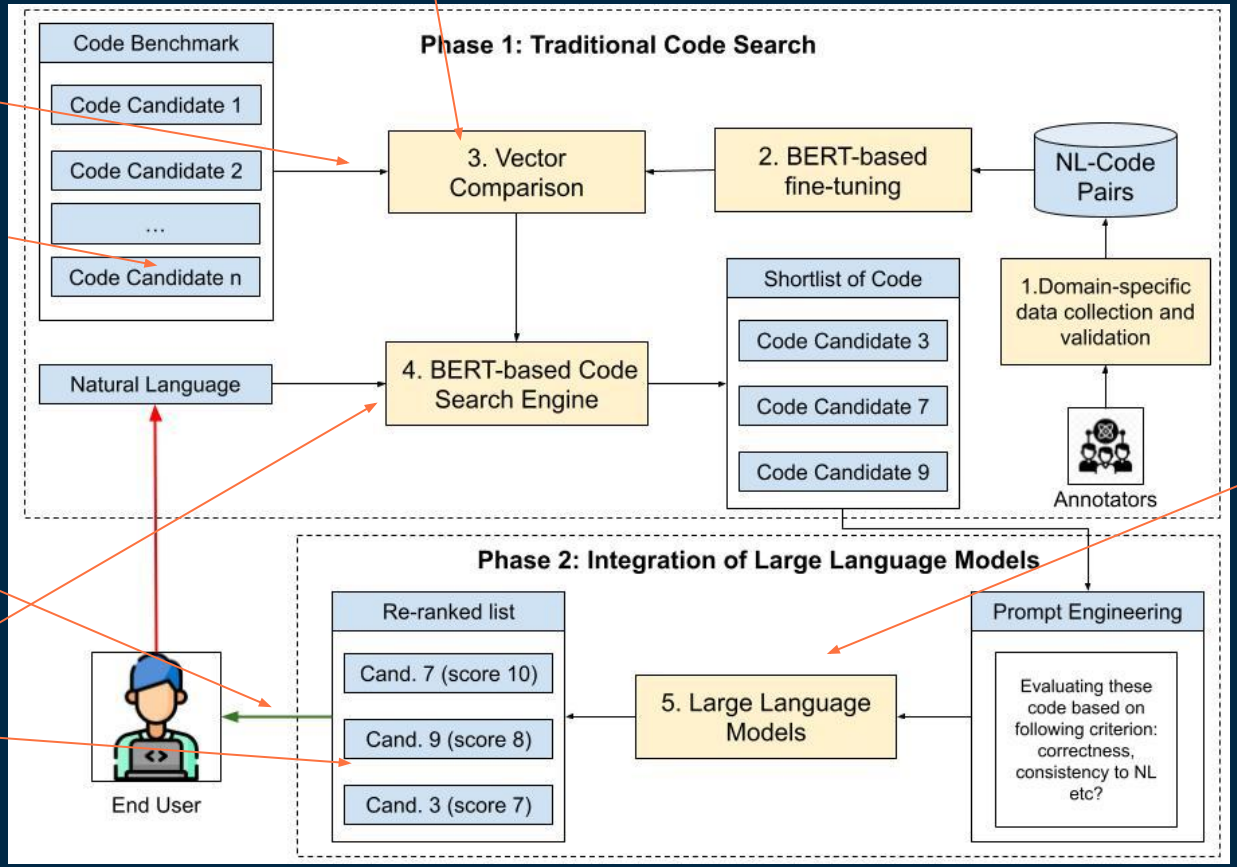
Overview of Design and Implementation

Dataset moved to SQLite, loads much faster

Generate embeddings using GPU versus CPU (~3x speed up)

New web search UI improves accessibility

In process of benchmarking UniXCoder results Vs Reranked results



Switched out external API calls to on-prem LLM to rank results
Faster
Significantly cheaper, which is useful for benchmarking

Vector Search Performance Improvements

Design and implementation

- Speed up queries by moving away from json.
- Localize the model
- Take advantage of hardware acceleration.

Obstacles

- Access to remote machine hasn’t gone as planned
- Cuda vs mps development (thank you PyTorch)

Remaining work

- Better use of the GPUs.

Performance results (generating 10,000 embeddings)

Before:

- Data.json = 248MB
- Embeddings generated in ~97 min

After:

- embeddings.db = 47.7MB
- Embeddings generated in 20:55 min

Performance results (nl search and LLM ranking)

Ran 100 times, through its “dataset” of 10,000 embeddings

Before:

- Average of **9.27s**

After:

- Average of **4.69s**

Accelerating LLM Calls (Carter)

Design and implementation

Obstacles:

- Slow Api Call Speeds (35+ seconds)
- Network Connectivity Problems (drops)
- Rate Limits

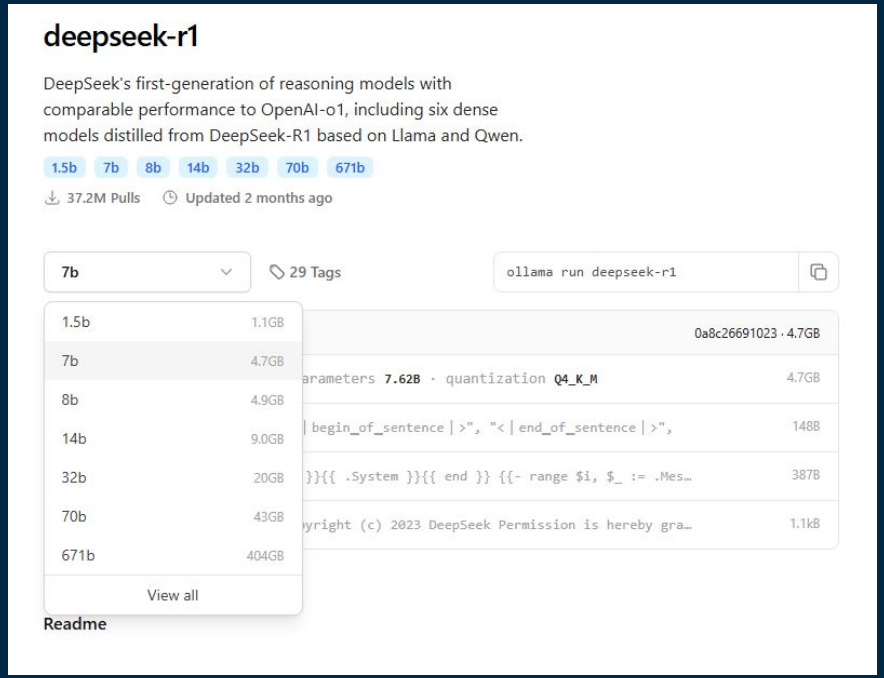
Accelerating LLM Calls

Solution: Local Hosting



Accelerating LLM Calls

avg 40% increase in call speed!



Accelerating LLM Calls

Challenges:

- Speed vs Accuracy
- Model Hallucinations
- Hardware Limits

How to Overcome:

- Using better models
- Better Prompting
- Lab Computer

Benchmarking Against UniXcoder (Conor)

Design and Implementation

- Rework of accuracy assessment
- Feed LLM the top 10 most accurate in order to get new best
  - Match with corresponding URLs
  - Must determine best code segment to give LLM

$$\text{Average}(1/\text{rank})$$

Benchmarking Against UniXcoder (Conor)

Model	Clone Detection				Code Search		
	POJ-104	BigCloneBench			CosQA	AdvTest	CSN
	MAP@R	Recall	Precision	F1-score	MRR		
RoBERTa	76.67	**95.1**	87.8	91.3	60.3	18.3	61.7
CodeBERT	82.67	94.7	93.4	94.1	65.7	27.2	69.3
GraphCodeBERT	85.16	94.8	95.2	95.0	68.4	35.2	71.3
SYNCOBERT	88.24	-	-	-	-	38.3	74.0
PLBART	86.27	94.8	92.5	93.6	65.0	34.7	68.5
CodeT5-base	88.65	94.8	94.7	95.0	67.8	39.3	71.5
UniXcoder	**90.52**	92.9	**97.6**	**95.2**	**70.1**	**41.3**	**74.4**
-w/o contras	87.83	94.9	94.9	94.9	69.2	40.8	73.6
-w/o cross-gen	90.51	94.8	95.6	95.2	69.4	40.1	74.0
-w/o comment	87.05	93.6	96.2	94.9	67.9	40.7	72.6
-w/o AST	88.74	92.9	97.2	95.0	68.7	40.3	74.2
-using BFS	89.44	93.4	96.7	95.0	69.3	40.1	74.1
-using DFS	89.74	94.7	94.6	94.7	69.0	40.2	74.2

Benchmarking Against UniXcoder (Conor)

Obstacles

- Confusion over what accuracy we should be checking
- What code should the LLM receive?
- Interacting with entire benchmarks that are not ours
    - Have to learn the pre-existing code and how to adapt it

Benchmarking Against UniXcoder (Conor)

Remaining Work and Current Progress

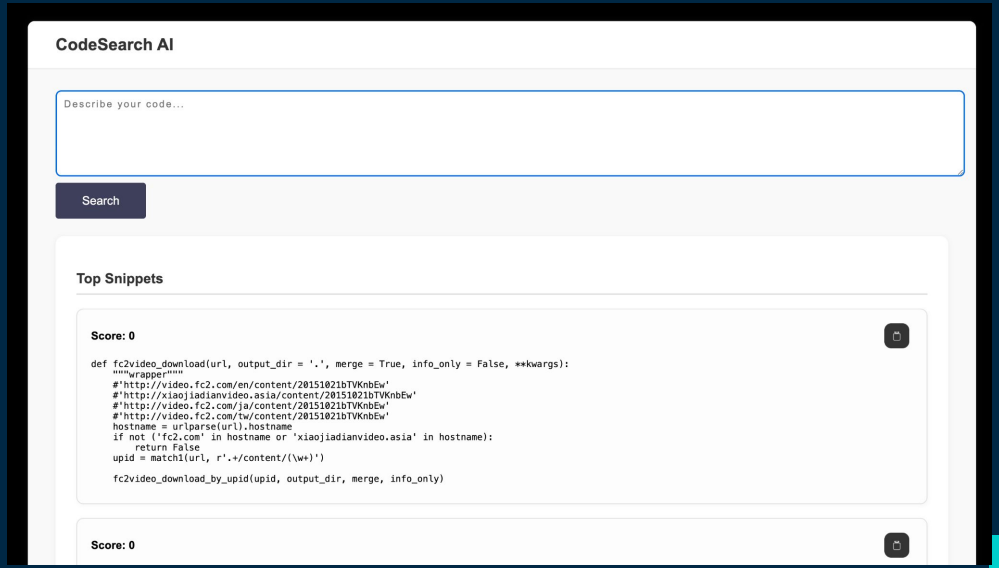
- Finish our benchmarking
    - Should see improvements from base UniXcoder search
- Benchmark with the different provided datasets
- Benchmark with different LLMs as well

Frontend: Design and Implementation (Keenan)

- Built with Flask and HTML/CSS/JS
- Clean, minimalist UI
- Users enter a code prompt and receive relevant snippets
- Displays both pre-ranking (vector similarity) and post-ranking (LLM-evaluated) results
- Supports:
  - Copy-to-clipboard button
  - Loading animation
  - Responsive layout

Frontend: Design Principles

- Minimal distractions: black background with white content container
- Mobile-friendly and accessible
- Snippet display designed for quick readability
- Interactive elements:
  - Search bar with hover styling
  - Visual feedback during loading and copying



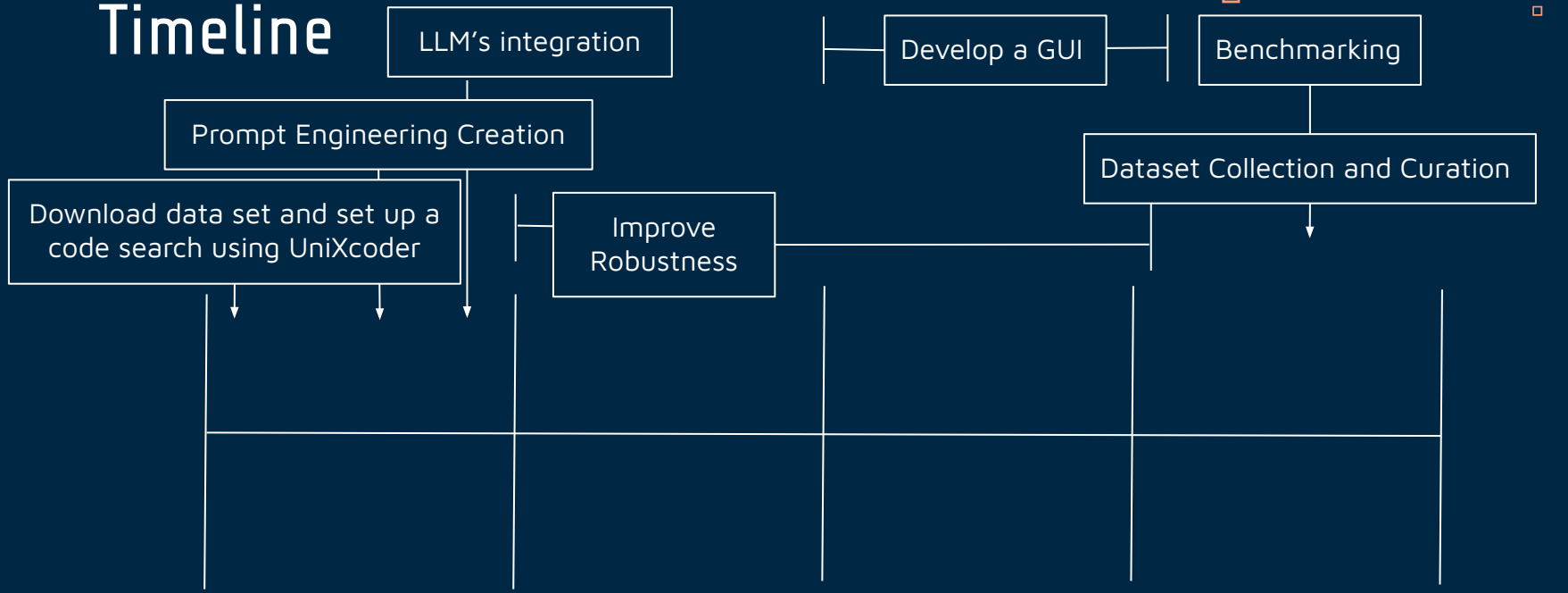
Frontend: Obstacles

- Not too many
- Learning Flask for the first time
- Learned how Flask looks for templates/ and static/ directories by default
- Improved directory structure to keep frontend code modular

Frontend: What's Left

- Side-by-side layout for pre/post rankings (currently vertical)
- Consider syntax highlighting with a library like highlight.js
- More UI polish: transitions, score color-coding, mobile tweaks
- Possibly integrate Dr. Phan's dataset instead of CodeSearchNet

Timeline


LLM's integration
Develop a GUI
Benchmarking
Prompt Engineering Creation
Dataset Collection and Curation
Download data set and set up a code search using UniXcoder
Improve Robustness
Demo 1
Spring Break
Demo 2
Final Project

Demo

