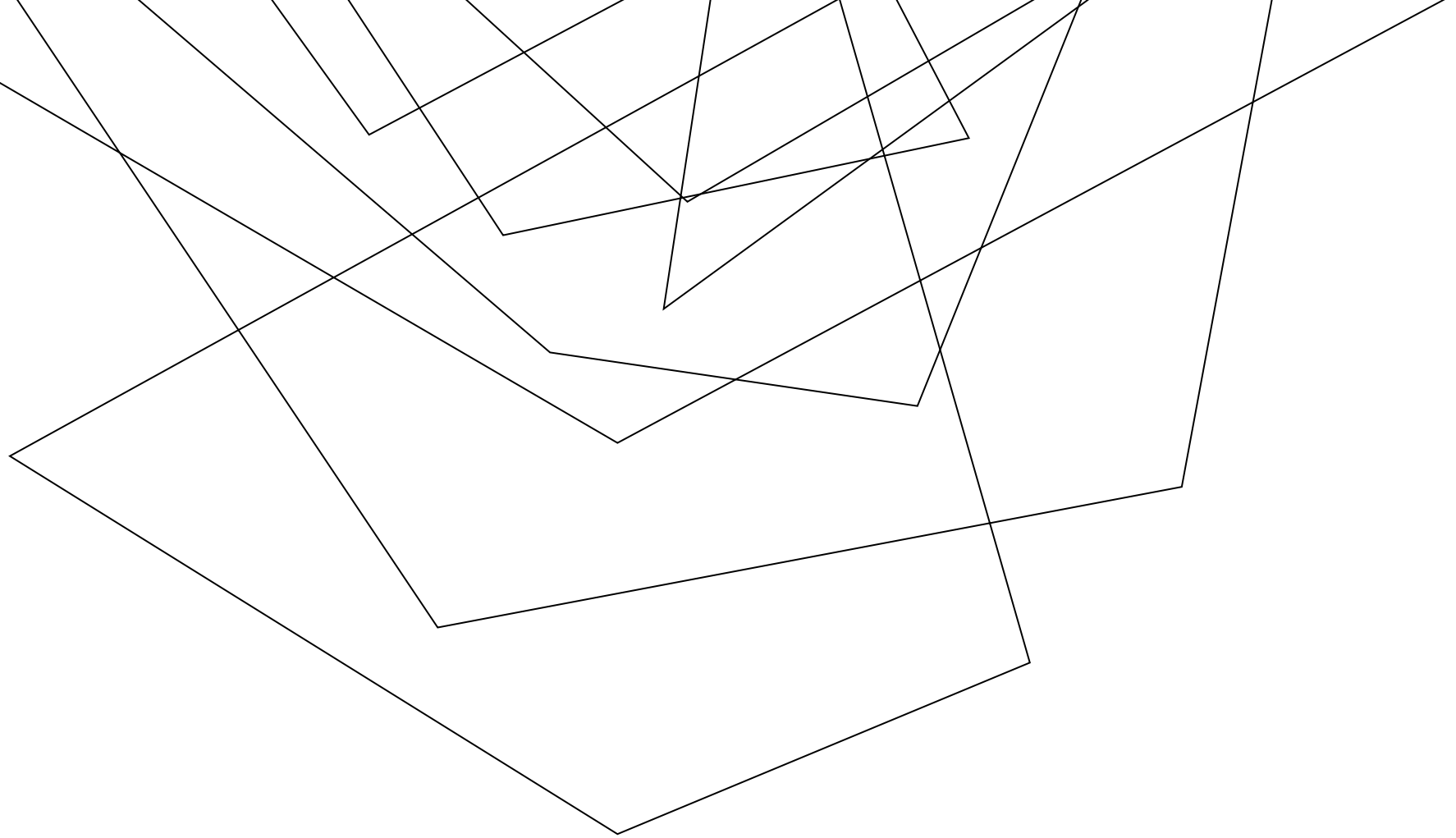
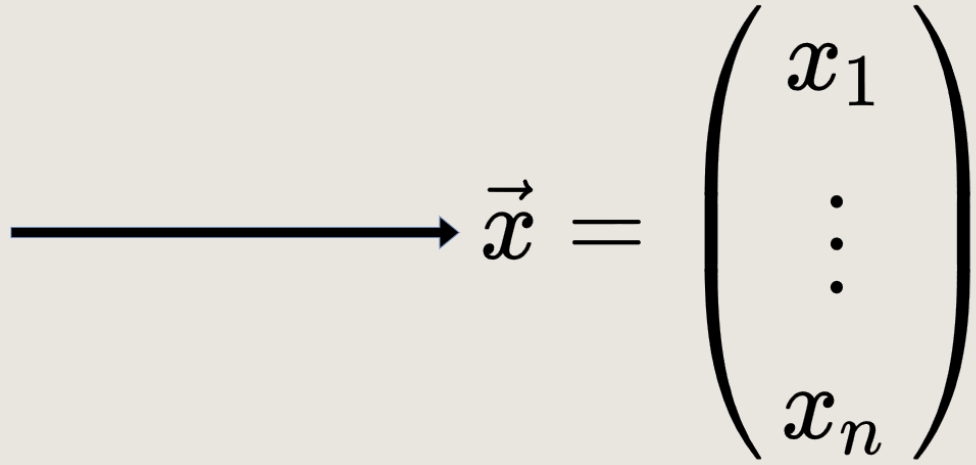


CREATING RICH LABELS FOR BINARY FUNCTIONS USING SOURCE INFORMATION

Ruben Triwari

ASSEMBLY CODE EMBEDDING

```
1. square(int):
2.         push    rbp
3.         mov     rbp, rsp
4.         mov     DWORD PTR [rbp-4], edi
5.         mov     eax, DWORD PTR [rbp-4]
6.         imul    eax, eax
7.        pop     rbp
8.         ret
```

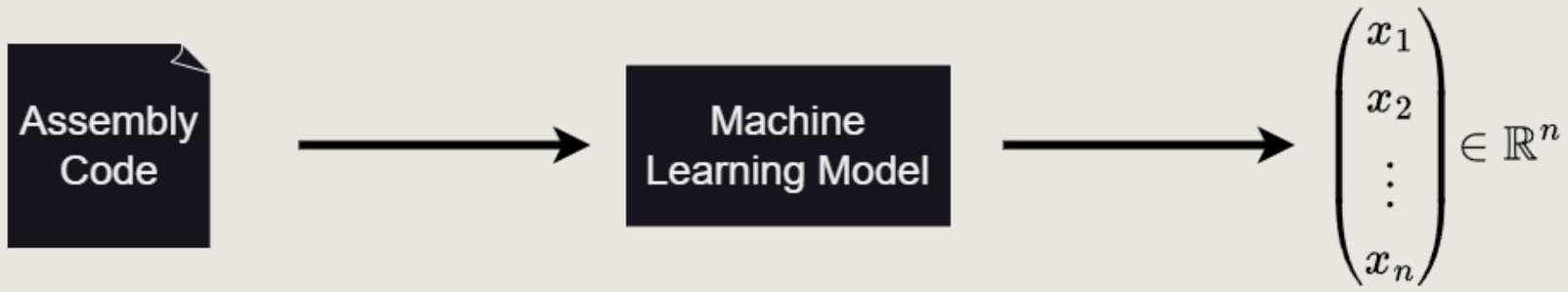


$$\vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

- Nearest Neighbor Search
- Classification with Labels (e.g. Networking, time, ...)
- XFL: eXtreme Function Labeling
- Binary Code Similarity Detection (BCSD)

⇝ Assist Reverse Engineering

GENERATING ASSEMBLY EMBEDDINGS



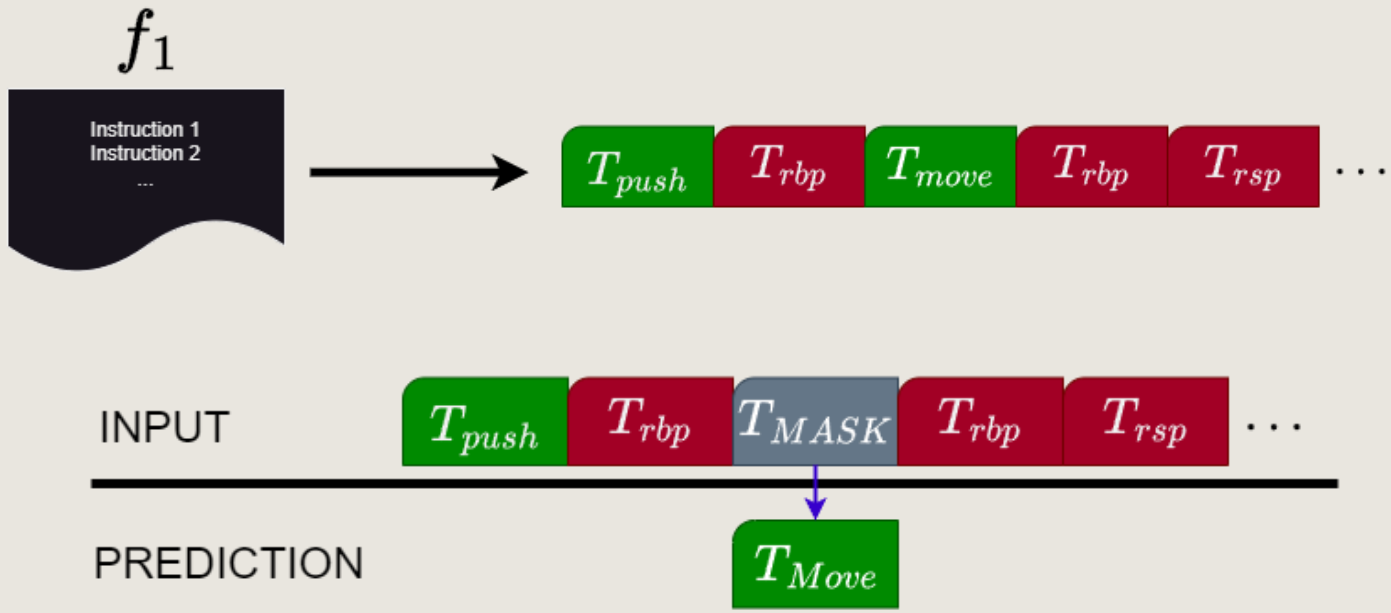
- Machine Learning Models to generate Assembly Embeddings
- Models need an objective function
  - ⇝ What is a “good” Objective function?

OVERVIEW OF DIFFERENT APPROACHES

- PalmTree[1] & JTrans[2]: BERT[3] like model to learn embeddings
- SAFE[4]: Supervised Learning with same Source as Label

[1]: Li et al.: PalmTree: Learning an Assembly Language Model for Instruction Embedding, CCS ‘21
[2]: Wang et al.: JTrans: Jump-Aware Transformer for Binary Code Similarity, ISSTA ’22
[3]: Devlin et al.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, NAACL ‘19
[4] Massarelli et al.: SAFE: Self-Attentive Function Embeddings for Binary Similarity, DIMVA ‘19

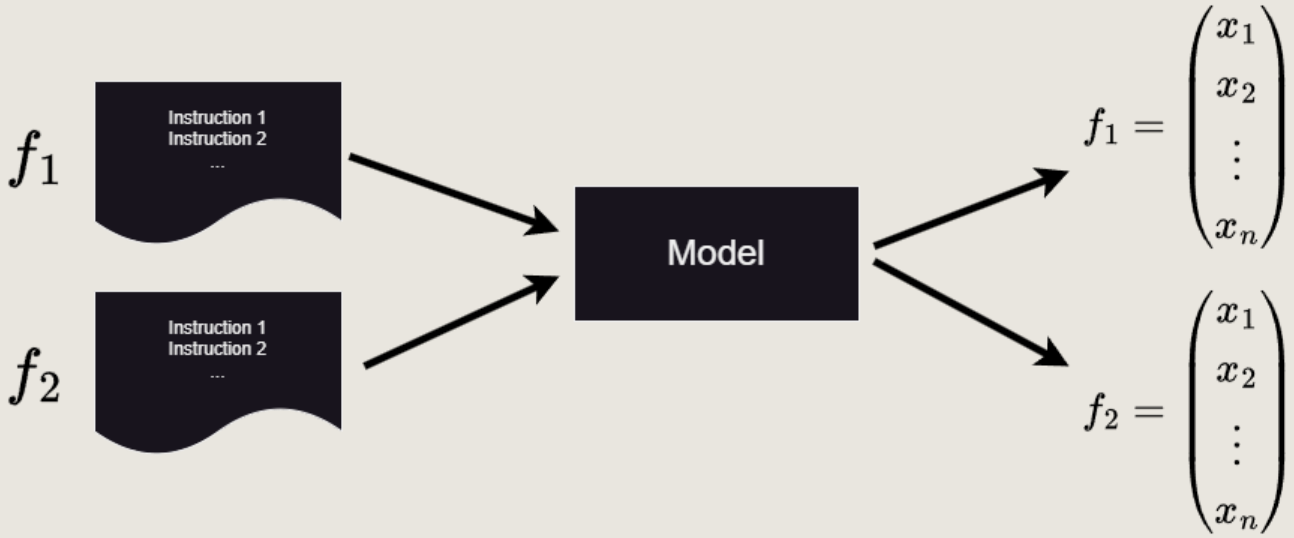
PALMTREE & JTRANS



- Self-supervised learning
- Tasks to train Model (e.g., Masking MLM)

  ⇝ State of the art in Binary Code Similarity Detection

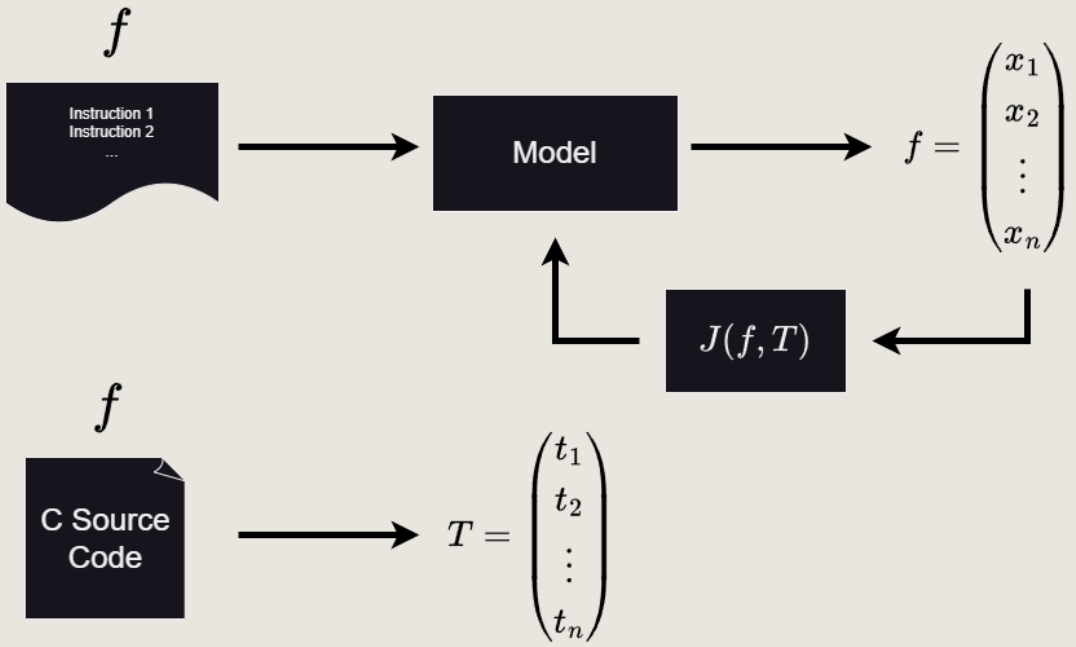
SAFE: SELF-ATTENTIVE FUNCTION EMBEDDINGS



- f_1 and f_2 same source code ⇝ minimize distance
- f_1 and f_2 different source code ⇝ maximize distance

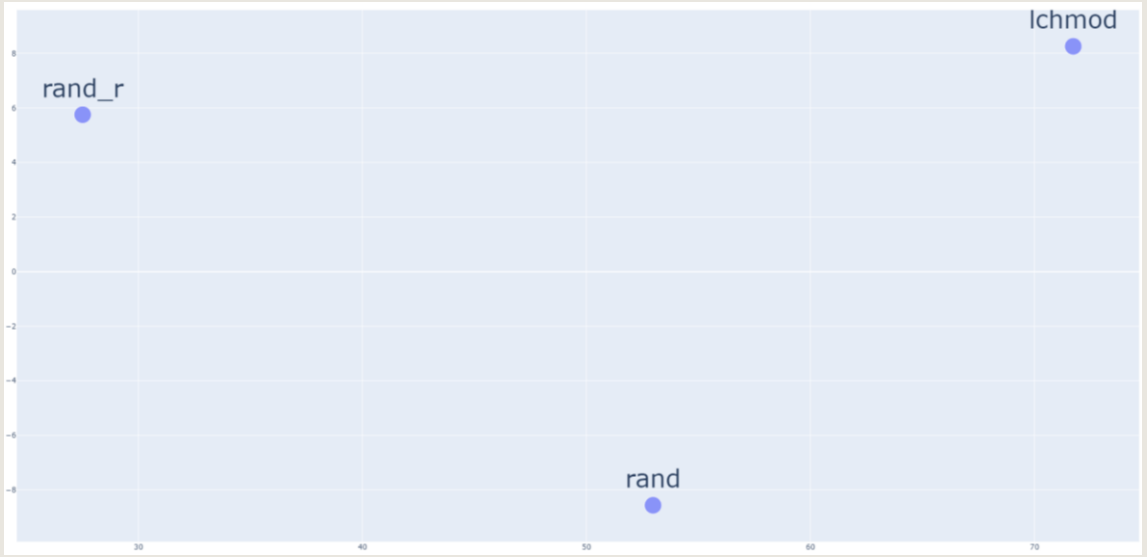
⇝ Similar functions are potentially far away from each other

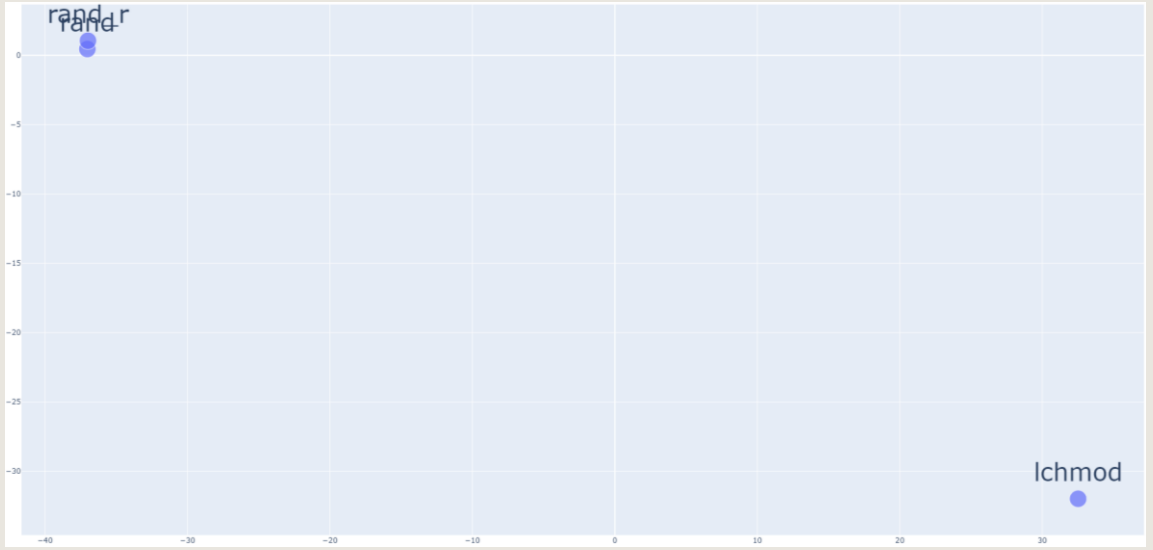
OUR APPROACH



- Supervised Learning
- Using C Source Code to generate “ground truth”

GROUND TRUTH





- Semantic similar functions ⇝ Vectors close to each other
- Semantic similar functions ⇝ Vectors far away from each other

  ⇝ Similar functions are grouped together

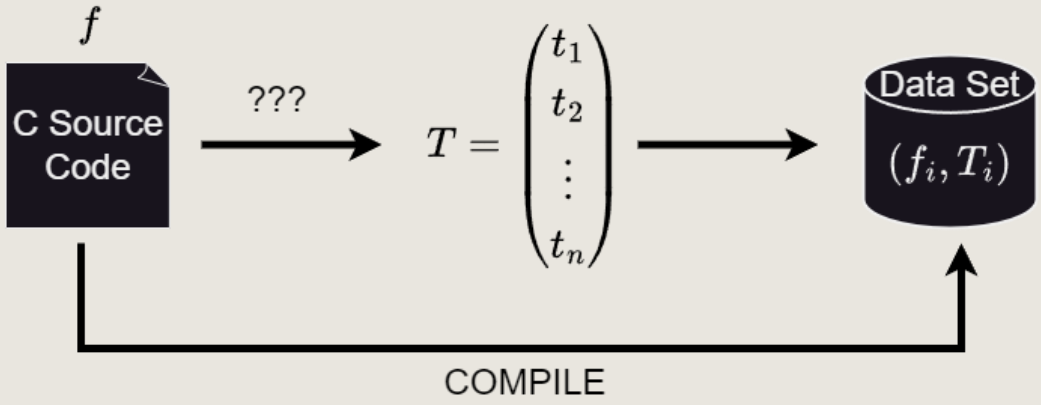
SENTENCE TRANSFORMER[1]

- Encoding Sentences semantically to Vectors
- Groups sentences with similar semantic

  ⇝ Exactly what we want for our ground truth

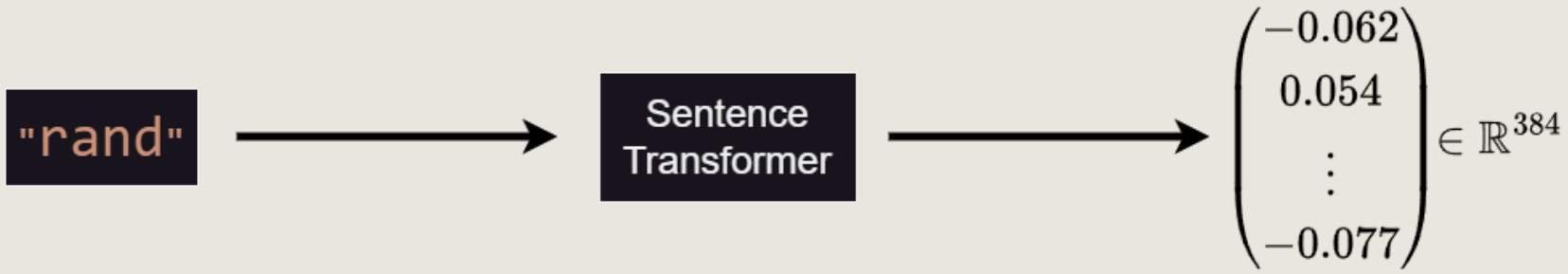
[1]: Sentence Transformer: https://www.sbert.net/

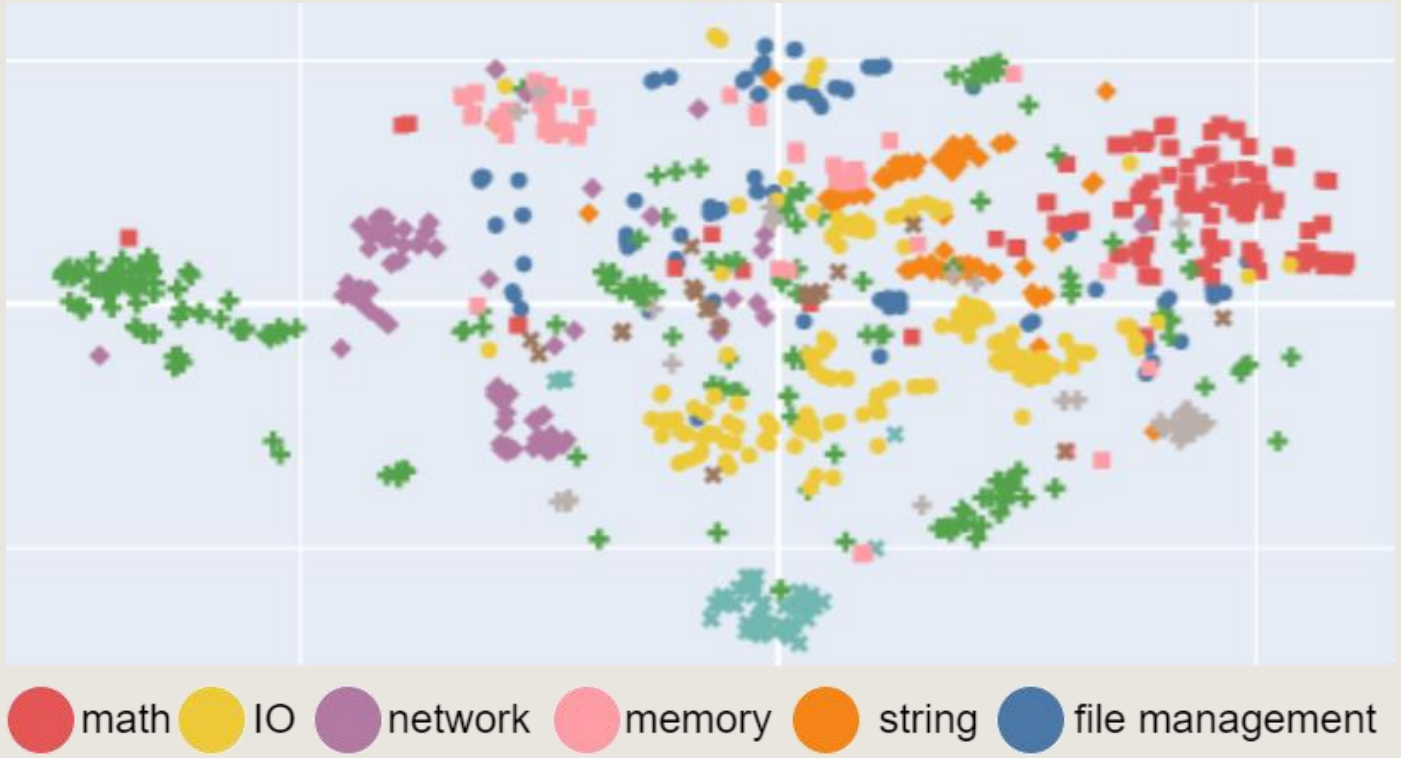
MY PART



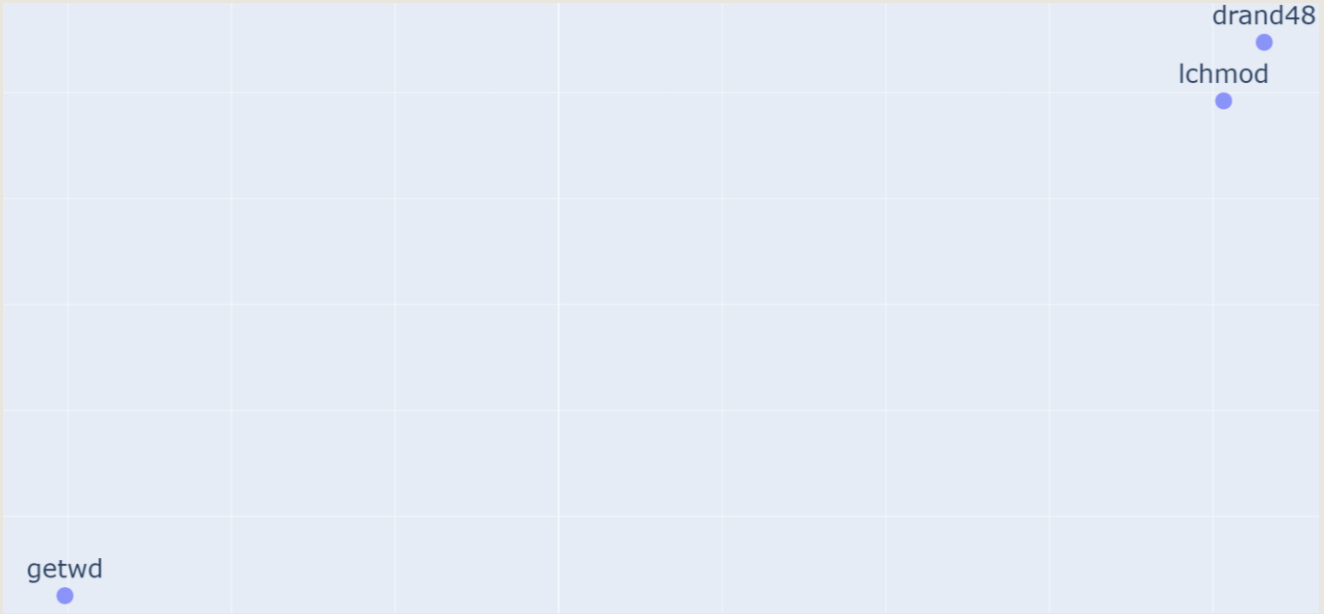
- Sentence Transformer on C Source Code Names
- Sentence Transformer on C Source Code Comments
- Sentence Transformer on LLM Code Summaries generated

C SOURCE CODE NAMES



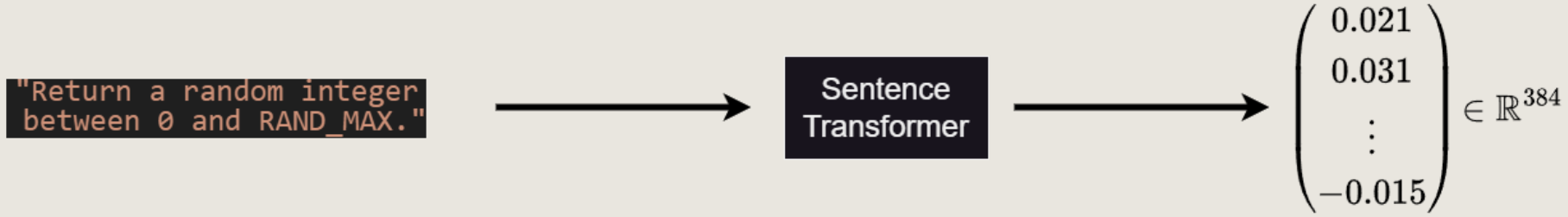


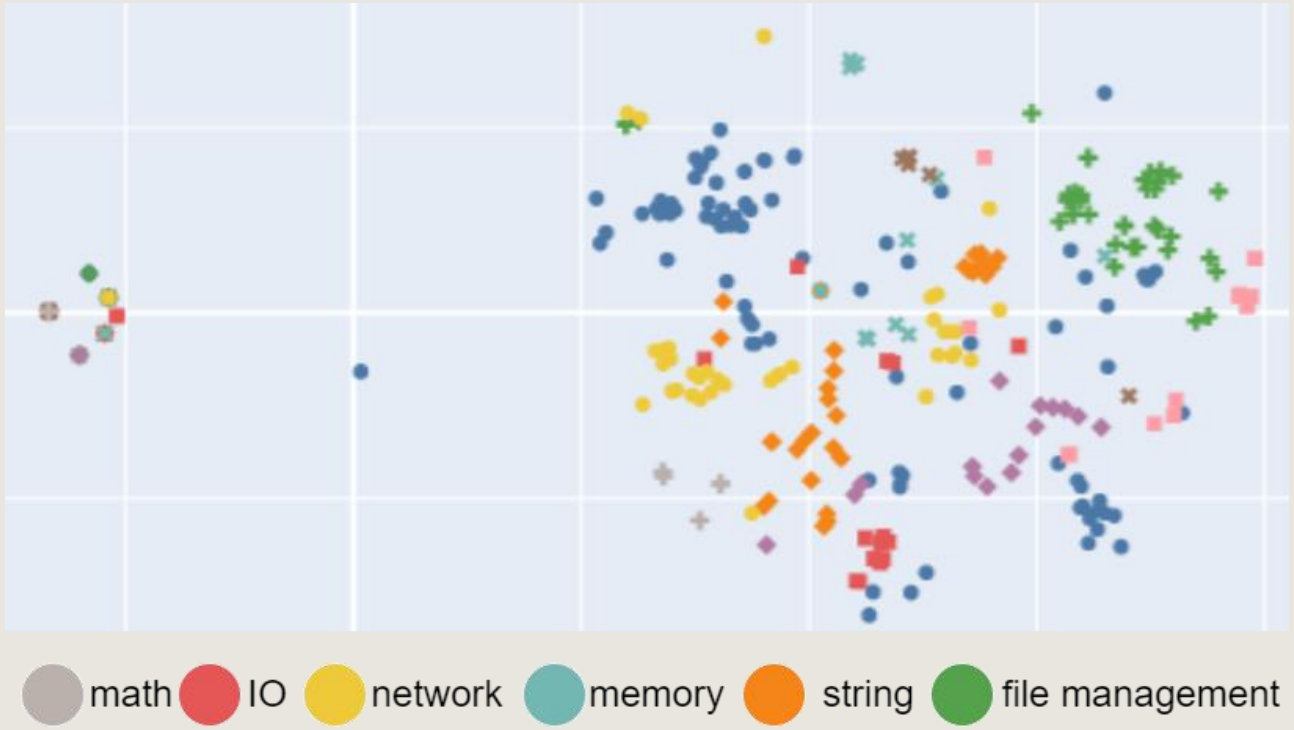
LIMITATION



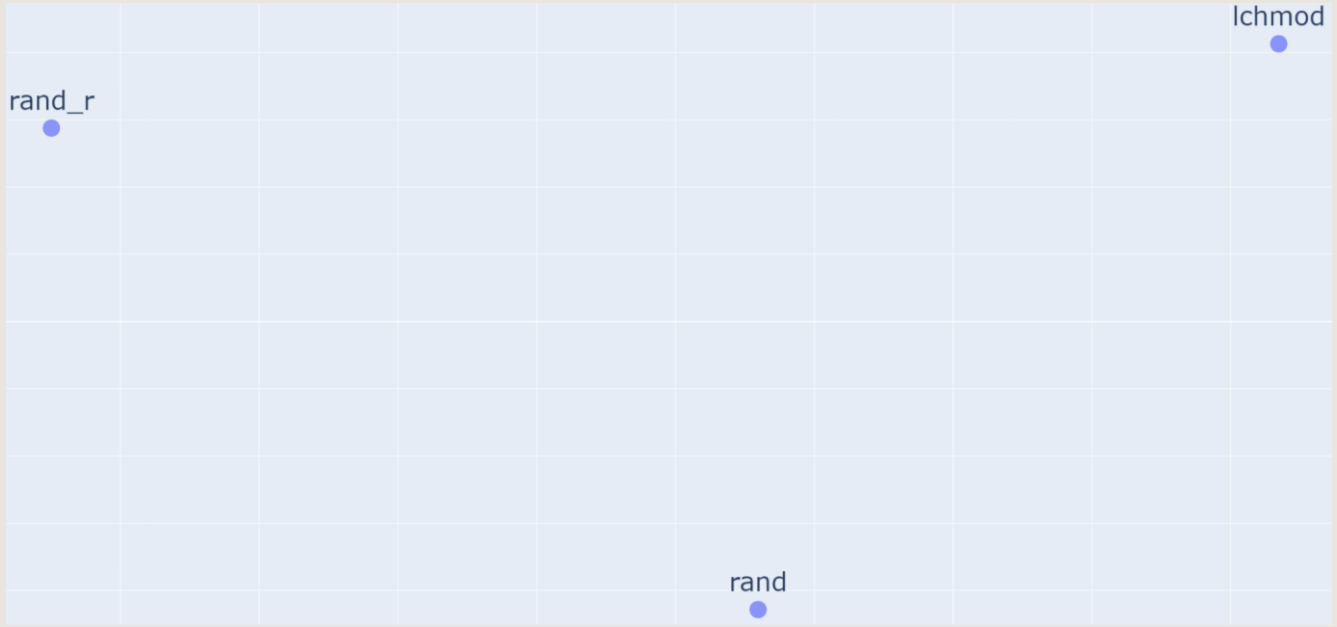
- “lchmod”, “getwd”: file operation
- “drand48”: Generates a random Number
  - ⇝ “lchmod” and “getwd” should be close
  - ⇝ Abbreviations can potentially confuse the sentence transformer

C SOURCE CODE COMMENTS





LIMITATION



- “rand_r” comment: “`This algorithm is mentioned in the ISO C…` “
- “rand” comment: “`Return a random integer between 0 and RAND_MAX.`”

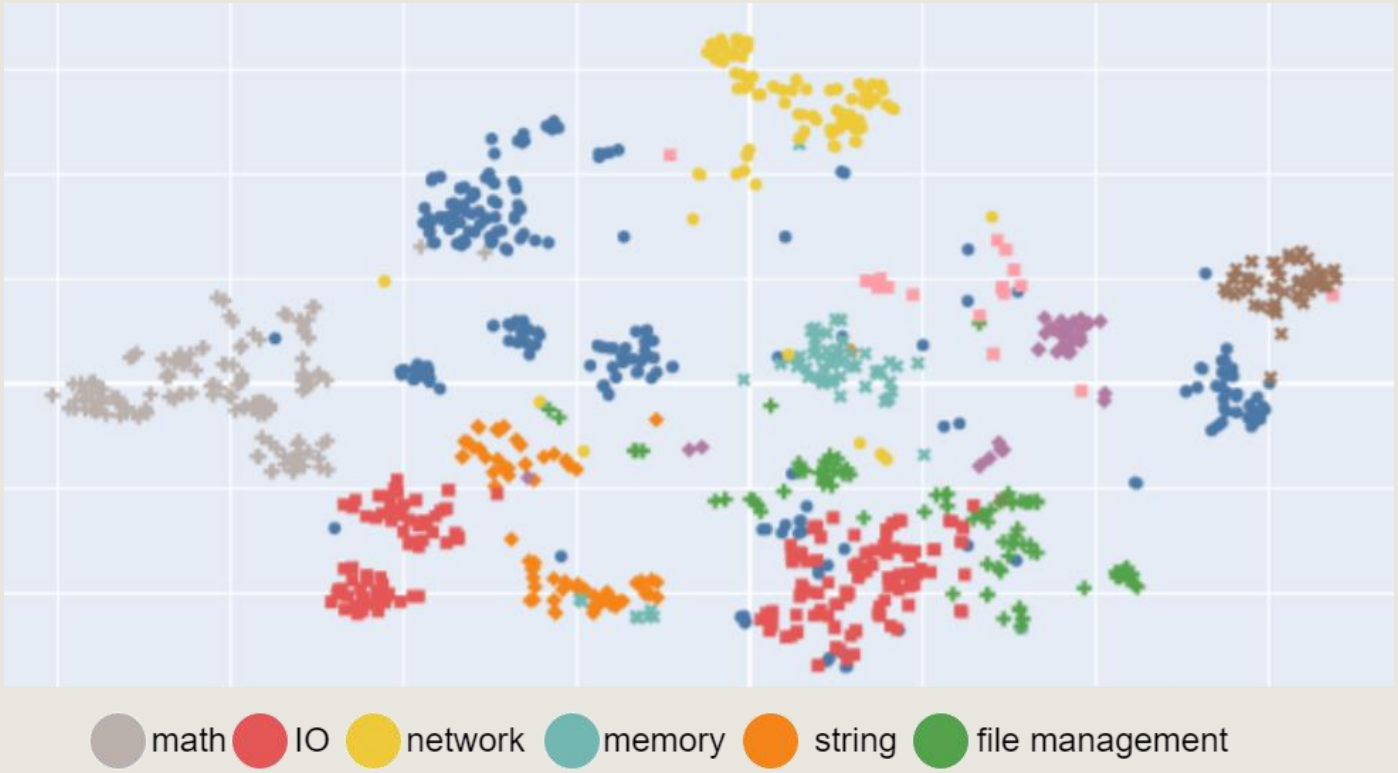
  ⇝ Comments are not Always summarizing the source code

  ⇝ Comments are not guaranteed to even exist

LLAMA

- Meta’s Large Language Model
- Open Source ⇝ runs on our server
- Prompt used: “Can you briefly summarize in one two sentence what the following function does? And can you give just the summary?\n”

LLAMA GENERATED CODE SUMMARIES



EVALUATION

- Qualitative evaluation:
  - Visualize Label Space with different Dimensionality Reduction Techniques (t-SNE, UMAP, MDS, Kernel-PCA)
  - Compare Automatic Clustering with DBSCAN/HDBSCAN[1] to manual inspection
- Quantitative evaluation:
  - Train Models with my data set
  - Performance in downstream task

[1]: Ester et al.: DBSCAN: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, KDD'96

DISCUSSION

- What would be a better Prompt?
- C Embeddings with Code2Vec[1]
- Use Code Llama instead of standard Llama
- Alias problem in “Glibc” data set
- Other useful source code information I could extract?
- Combining vectors of different source code information?

[1]: Alon et al.: Code2Vec: Learning Distributed Representations of Code, POPL ‘19