Comparing Natural Language Embeddings for Libc Functions as Rich Labels

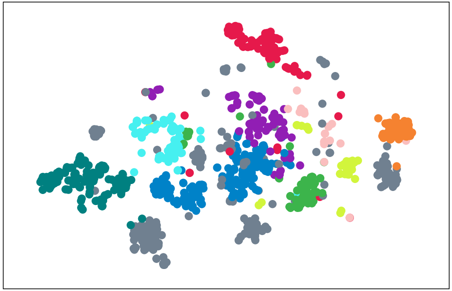
Bachelor's Thesis Defense

Ruben Triwari

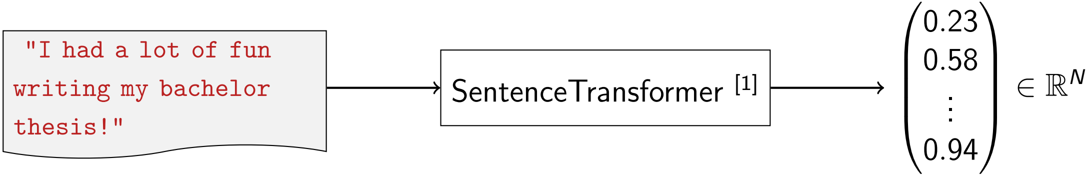
Ludwig Maximilian University Munich

February 19, 2025

Outline



Motivation



- ⇝ Encoding natural language had an important role in recent NLP advancements
- ⇝ Information described as a vector can be used in many downstream tasks
- ⇝ That serves as an motivation for encoding binary code as vector
- ⇝ That motivates using NLP tools to encode binary code

[1] Reimers and Gurevych: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, EMNLP'19

Motivation

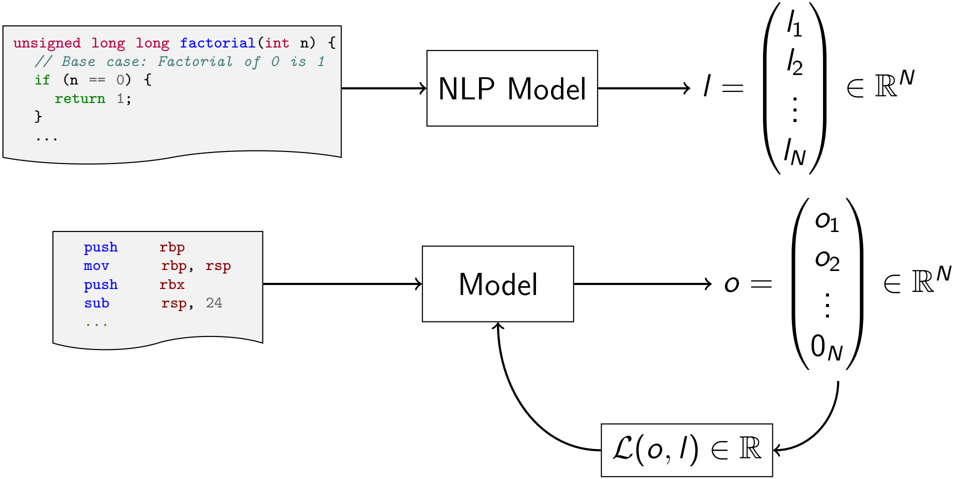
```c
unsigned long long factorial(int n) {
  // Base case: Factorial of 0 is 1
  if (n == 0) {
    return 1;
  }
  // Recursive case: n! = n * (n-1)!
  return n * factorial(n - 1);
}
```

→ Compiler →

```asm
factorial(int):
    push    rbp
    mov     rbp, rsp
    push    rbx
    sub     rsp, 24
    mov     DWORD PTR [rbp-20], edi
    cmp     DWORD PTR [rbp-20], 0
    jne     .L2
    mov     eax, 1
    jmp     .L3
.L2:
    mov     eax, DWORD PTR [rbp-20]
    movsx   rbx, eax
    mov     eax, DWORD PTR [rbp-20]
    sub     eax, 1
    mov     edi, eax
    call    factorial(int)
    imul    rax, rbx
.L3:
    mov     rbx, QWORD PTR [rbp-8]
    leave
    ret
```

⇝ Compiler removes all information that is in natural language

Motivation


unsigned long long factorial(int n) {
// Base case: Factorial of 0 is 1
if (n == 0) {
return 1;
}
...
NLP Model
$l = \begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ l_N \end{pmatrix} \in \mathbb{R}^N$
push rbp
mov rbp, rsp
push rbx
sub rsp, 24
...
Model
$o = \begin{pmatrix} o_1 \\ o_2 \\ \vdots \\ 0_N \end{pmatrix} \in \mathbb{R}^N$
$\mathcal{L}(o, l) \in \mathbb{R}$

Research Objectives

- Compare different approaches encoding additional information in the source code into machine readable format
  1. Embed function names with SentenceTransformer
  2. Embed function comments with SentenceTransformer
  3. Embed Code Llama [2] code summaries with SentenceTransformer

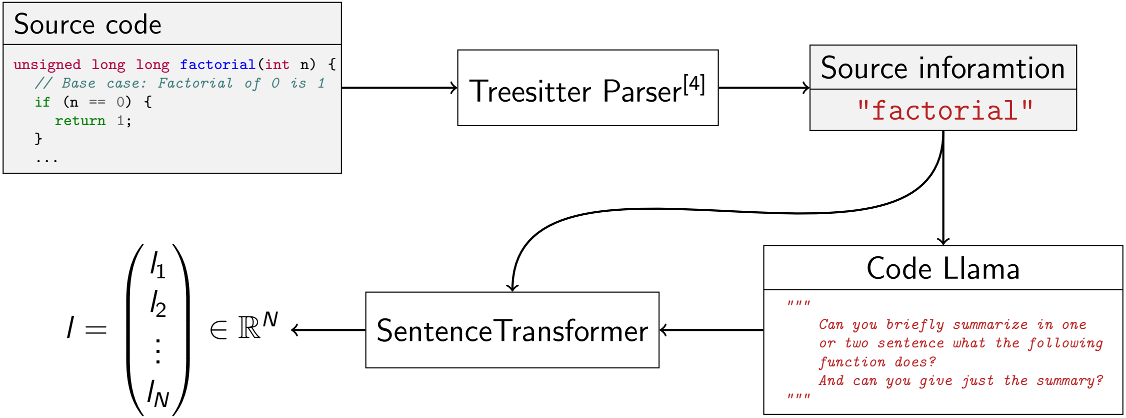
  $\rightsquigarrow$ Intuition is that Code Llama explanation will yield "good" embeddings
- Compare NLP approach to the existing Code2Vec [3] Model
- Propose a new way comparing embedding spaces.

  $\rightsquigarrow$ To prove intuition

[2] Rozière et al.: Code Llama: Open Foundation Models for Code, 24

[3] Alon et al.: code2vec: Learning Distributed Representations of Code, POPL'19

Architecture



[4] Official website: `https://tree-sitter.github.io/tree-sitter/`

Evaluation with t-SNE

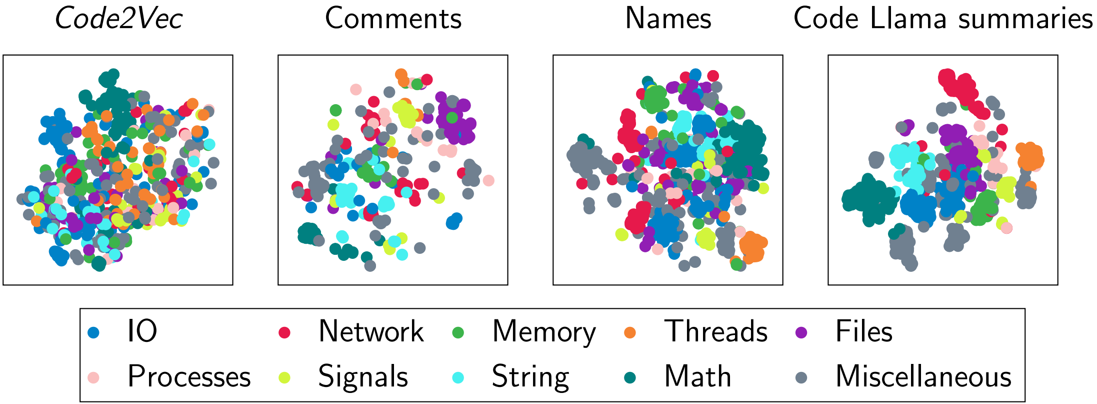


Figure: Depicted are the *t-SNE* output vectors with perplexity $P = 30$.

Expert Survey

"fmaximum_numl"

1. fminimum_magl
2. fminimuml
3. fminimum_mag_numl
4. fminimum_numl

○ Yes
○ No

Figure: Positive example

"execl"

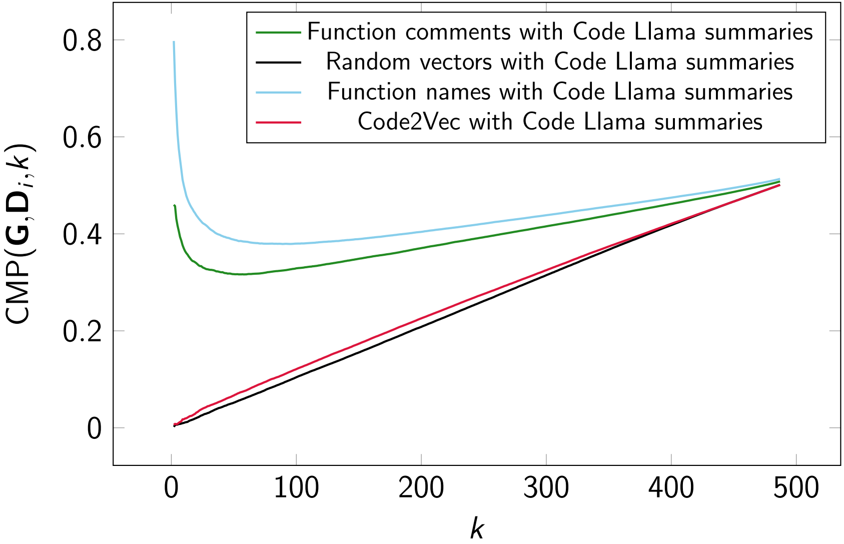
1. j1l
2. exp10l
3. exp2l
4. expm1l

○ Yes
○ No

Figure: Negative example

Expert survey results				
Method	Code Llama summaries	Function names	Function comments	Code2Vec
Score	0.596	0.532	0.433	0.321

Embedding space comparison



Function names

Abbreviations can potentially confuse the SentenceTransformer:
Example function `lchmod`:

$$\text{l} \leftrightarrow \text{link}, \quad \text{ch} \leftrightarrow \text{change}, \quad \text{mod} \leftrightarrow \text{file mode}.$$

Nearest neighbors in function space:

$$\texttt{lchmod} \leftrightarrow \left(\texttt{lcong48}, \ \ \texttt{fchmodat}, \ \ \texttt{coshl}, \ \ \texttt{cacoshl}\right)$$

$\rightsquigarrow$ In categories:

$$\texttt{files} \leftrightarrow \left(\texttt{math}, \ \ \texttt{files}, \ \ \texttt{math}, \ \ \texttt{math}\right)$$

function names

Example function `lchmod`:

$$\text{l} \leftrightarrow \text{link}, \quad \text{ch} \leftrightarrow \text{change}, \quad \text{mod} \leftrightarrow \text{file mode}.$$

Nearest neighbors in code llama summary space:

$$\texttt{lchmod}$$

$$\leftrightarrow \left(\texttt{fchmodat}, \quad \texttt{fchownat}, \quad \texttt{euidaccess}, \quad \texttt{\_\_file\_change\_detection\_for\_stat}\right)$$

$\rightsquigarrow$ In categories:

$$\texttt{files} \leftrightarrow \left(\texttt{files}, \quad \texttt{files}, \quad \texttt{files}, \quad \texttt{files}\right)$$

function comments

Comments are not always directly about the code:
Example functions `rand` and `rand_r`:

rand $\leftrightarrow$ Return a random integer between 0 and RAND_MAX.
rand_r $\leftrightarrow$ This algorithm is mentioned in the ISO C standard, here extended for 32 bits.

$\rightsquigarrow$ Cosine distance in comment and llama summary space

$$d_{\text{comment}}(\texttt{rand}, \texttt{rand\_r}) = 0.8544 \qquad d_{\text{llama}}(\texttt{rand}, \texttt{rand\_r}) = 0.2216.$$

Future Work

- Code Llama
  1. Is it necessary to use a large Model with 70B parameters?
  2. Can Large Language Models produce deterministic output for this application?
  3. Is there a better Prompt?
- Comments
  1. Use inline Comments

Conclusion

- Best strategies ranked:
  1. Code Llama summaries
  2. Function names
  3. Function comments
  4. Code2Vec
- Code Llama summary vectors for C source code downstream tasks
- Code Llama summary vectors can now be used to train a Model
- $\texttt{CMP}(A, B, k)$ function can be used to compare two embedding spaces from the same features Space
- Evaluation methods can be used to compare different Large Language Models to each other

Embedding space comparison

$$\texttt{compare}(u, v)_k = \frac{1}{G_k} \sum_{i=1}^{k} \frac{\texttt{score}_k(u_i, i, v)}{log_2(i+1)} \in [0, 1]$$

where

$$u, v \in \mathbb{N}^k : \text{Neighbor ranking of the same vector in diffrent spaces},$$

$$\texttt{score}_k(l, i, v) = \begin{cases} 1 & , \exists j \in \mathbb{N} : l = v_j \wedge i = j \\ \frac{1}{2} & , \exists j \in \mathbb{N} : l = v_j \wedge i \neq j \\ 0 & , \text{otherwise} \end{cases} \quad , G_k := \sum_{i=1}^{k} \frac{1}{log_2(i+1)}.$$

Embedding space comparison

$$\texttt{CMP}(A, B, k) = \frac{1}{N}\sum_{i=1}^{N} \texttt{compare}_k(NN_k(A_i, A), NN_k(B_i, B))$$

where

$A, B \in \mathbb{R}^{N \times l}$: Embedding space with N vectors of length l

$\texttt{NN}_k(A_i, A)$: k nearest neighbors from vector with index i in A

$k \in \mathbb{N}$: Amount of vectors we include in one neighborhood relation

Future Work

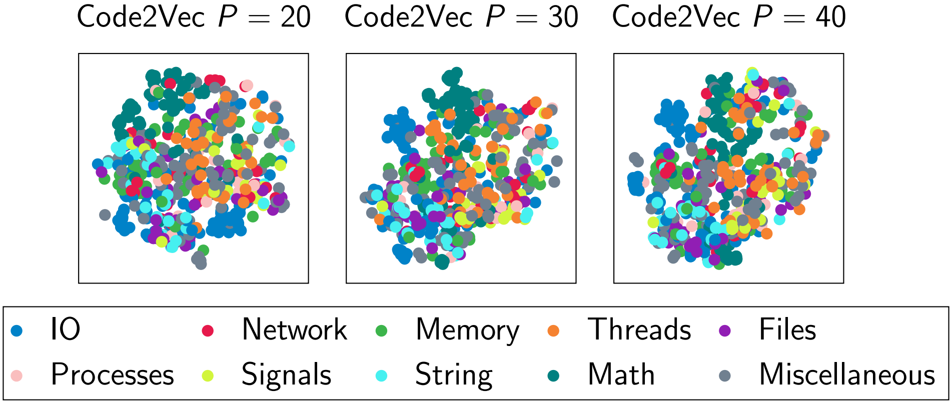
$$\texttt{CMP}(A, B, k) = \frac{1}{N}\sum_{i=1}^{N} \texttt{compare}_k(NN_k(A_i, A), NN_k(B_i, B))$$

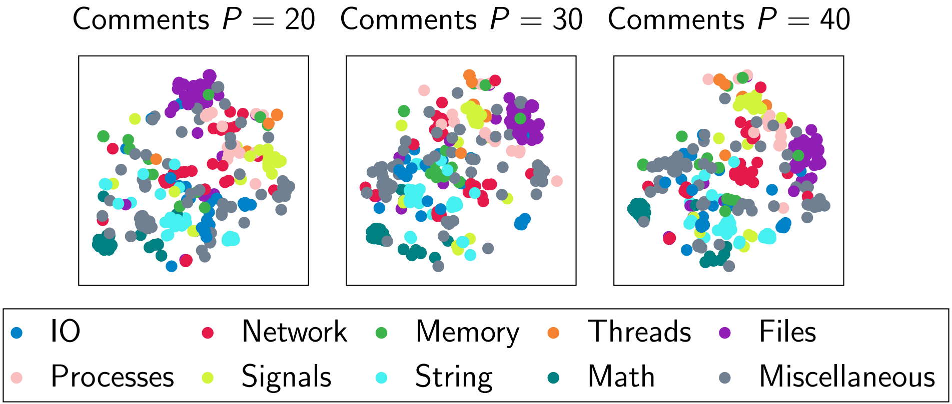
$$\texttt{compare}(u, v)_k = \frac{1}{G_k}\sum_{i=1}^{k} \frac{\texttt{score}_k(u_i, i, v)}{log_2(i+1)} \in [0, 1]$$

- $\texttt{CMP}(A, B, k) \in [0, 1]$ function
  1. Is there an optimal value for k?
  2. Is there a better way to generate a neighborhood? (instead of K-Nearest-Neighbor)
  3. Is there a better way to aggregate the compare functions?

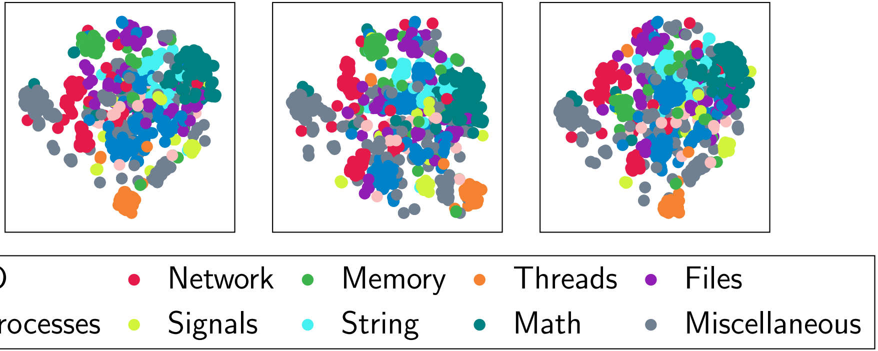
Code2Vec

- Also dependent on the function names in the data set
- Bad results could be Explained by:
  1. Small data set
  2. C instead of Java $\rightsquigarrow$ potential engineering mistakes
  3. Quality of names in the data set


Code2Vec $P = 20$
Code2Vec $P = 30$
Code2Vec $P = 40$
IO
Network
Memory
Threads
Files
Processes
Signals
String
Math
Miscellaneous


Comments $P = 20$
Comments $P = 30$
Comments $P = 40$
IO
Network
Memory
Threads
Files
Processes
Signals
String
Math
Miscellaneous

Names $P = 20$ Names $P = 30$ Names $P = 40$



Code Llama $P = 20$ Code Llama $P = 30$ Code Llama $P = 40$