Exploring Latent Program Spaces for Program Synthesis

Kendall Tauser

University of Oklahoma

December 5th, 2025

- This presentation operates under the assumption that we want computers that can generate code and ultimately program themselves.

- This presentation operates under the assumption that we want computers that can generate code and ultimately program themselves.
- The current overwhelming paradigm is to throw data and transformers at the problem and hope that it just works out.

- This presentation operates under the assumption that we want computers that can generate code and ultimately program themselves.
- The current overwhelming paradigm is to throw data and transformers at the problem and hope that it just works out.
- This has the major downfall of being very expensive on multiple fronts (data, compute, syntax).

- There exists tooling that defines programming languages rigorously.

- There exists tooling that defines programming languages rigorously.
- One technique for exploiting this for generating programs is with reinforcement-learning based grammar-expansion. (Bunel et al. 2018; Simmons-Edler, Miltner, and Seung 2018)

- There exists tooling that defines programming languages rigorously.
- One technique for exploiting this for generating programs is with reinforcement-learning based grammar-expansion. (Bunel et al. 2018; Simmons-Edler, Miltner, and Seung 2018)
- Want some sort of continuous representation of partially expanded programs that can provide an estimate of where we have been.

- There exists tooling that defines programming languages rigorously.
- One technique for exploiting this for generating programs is with reinforcement-learning based grammar-expansion. (Bunel et al. 2018; Simmons-Edler, Miltner, and Seung 2018)
- Want some sort of continuous representation of partially expanded programs that can provide an estimate of where we have been.
- Can we utilize embeddings as context when performing RL?

- There exists tooling that defines programming languages rigorously.
- One technique for exploiting this for generating programs is with reinforcement-learning based grammar-expansion. (Bunel et al. 2018; Simmons-Edler, Miltner, and Seung 2018)
- Want some sort of continuous representation of partially expanded programs that can provide an estimate of where we have been.
- Can we utilize embeddings as context when performing RL?
- The bigger question is, how do we know if the embedding is any *good*?

Big idea: Can we quantify how good embeddings are at retaining program structure?

This presentation consists of four main sections:

- Some background on grammars and embeddings.
- Introduction to the lang-explorer framework, its architecture, and its features.
- Overview of experiments conducted and results.
- Discussion, limitations, and future work.

A formal grammar is an object typically represented as the tuple $\mathcal{G} = (\mathcal{V}, \mathcal{T}, \mathcal{S}, \mathcal{P})$.

- $\mathcal{V}$ is the set of non-terminal symbols.
- $\mathcal{T}$ is the set of terminal symbols.
- $\mathcal{S} \in \mathcal{V}$ and is the start symbol.
- $\mathcal{P}$ denotes the set of productions used for expansion, each element is of the form $\alpha A \beta \rightarrow \alpha \gamma \beta$, where $\alpha, \beta \in \{\mathcal{V} \cup \mathcal{T} \cup \epsilon\}^*$, $\gamma \in \{\mathcal{V} \cup \mathcal{T} \cup \epsilon\}^+$ and $A \in \mathcal{V}$.

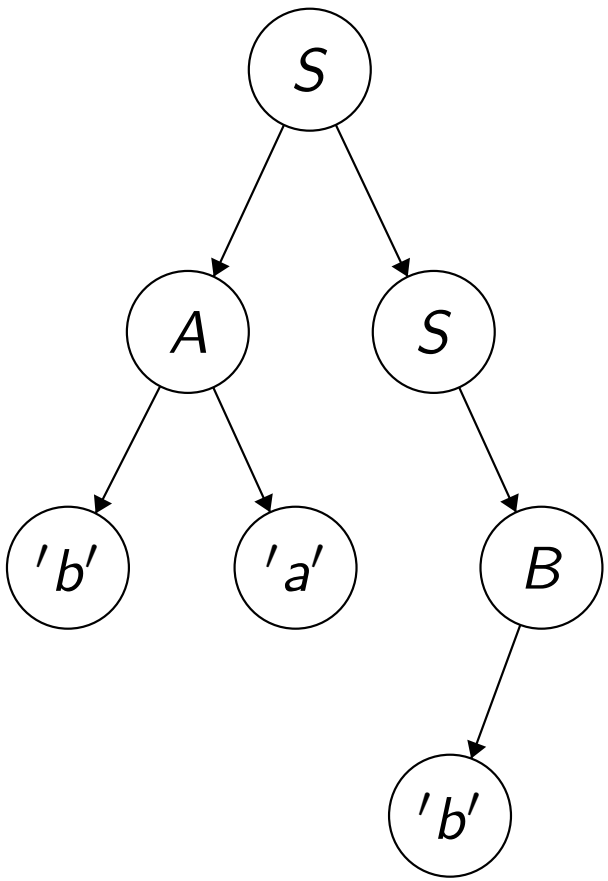
One can "expand" such grammars to form sentences, which are instances of programs within the space of all programs defined by a grammar.

$\langle S \rangle$ ::= <A><S> | <B>

$\langle A \rangle$ ::= 'a' | 'ba'

$\langle B \rangle$ ::= <B> | 'b'

Above: Example of a simple grammar with corresponding expansion tree to the right. The final program is 'bab'.



Embeddings

Models typically operate on tensors. Thus it can be useful to encode your domain-specific data as a tensor (namely a vector) for use within a model. This leads us to embeddings.

Embeddings

Models typically operate on tensors. Thus it can be useful to encode your domain-specific data as a tensor (namely a vector) for use within a model. This leads us to embeddings.

- The simplest example is one-hot encoding.

Object A →	1	0	0	...	0	0
Object B →	0	1	0	...	0	0

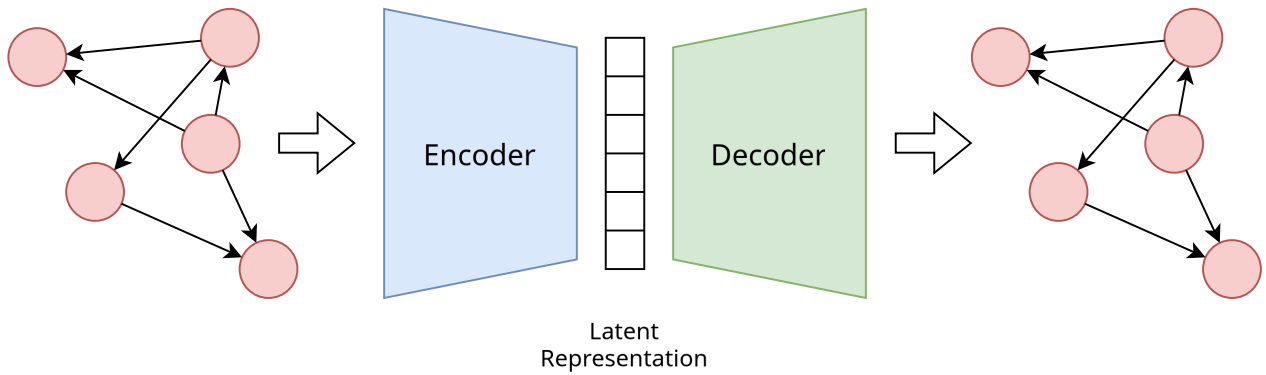
Embeddings

Models typically operate on tensors. Thus it can be useful to encode your domain-specific data as a tensor (namely a vector) for use within a model. This leads us to embeddings.

- The simplest example is one-hot encoding.

Object A →	1	0	0	...	0	0
Object B →	0	1	0	...	0	0

- Autoencoders are a way to learn the vectors from data.



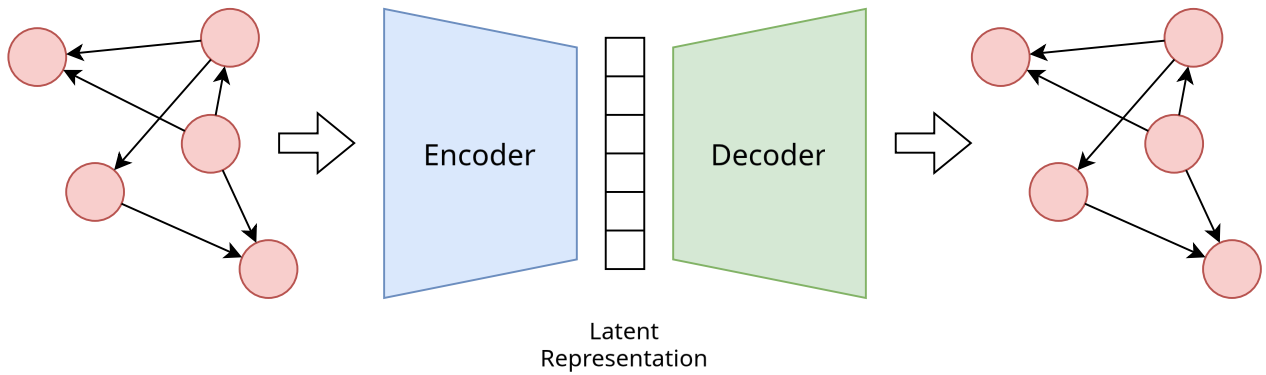
Embeddings

Models typically operate on tensors. Thus it can be useful to encode your domain-specific data as a tensor (namely a vector) for use within a model. This leads us to embeddings.

- The simplest example is one-hot encoding.

Object A →	1	0	0	...	0	0
Object B →	0	1	0	...	0	0

- Autoencoders are a way to learn the vectors from data.

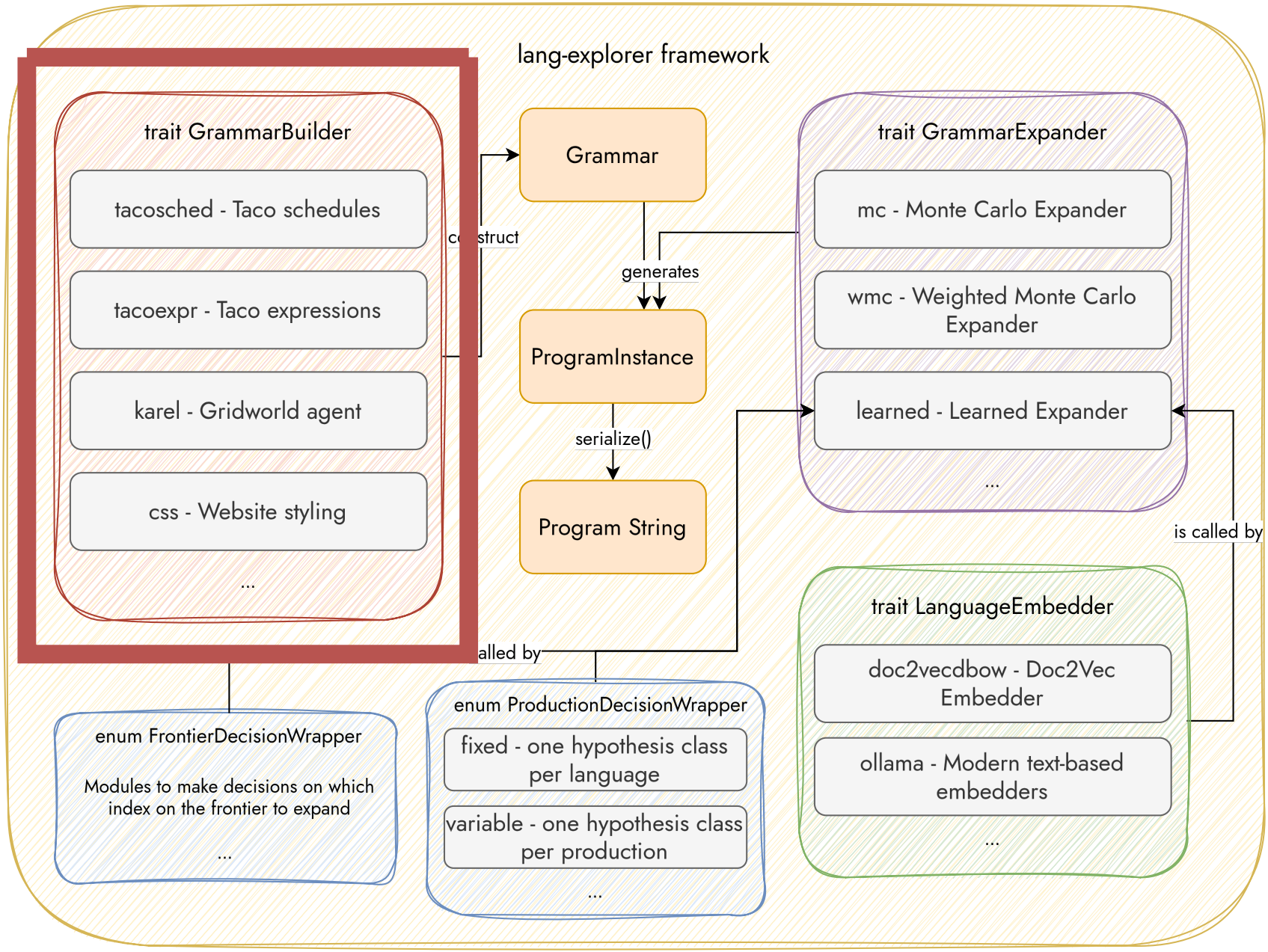


- There also exist more nuanced embedding techniques, which we will cover.

Methods I

- To create embeddings, we first need a set of languages to use.
- More specifically, it would be useful to have a set of languages where we can build a grammar, and then expand it to construct programs.

Methods II


lang-explorer framework
trait GrammarBuilder
tacosched - Taco schedules
tacoexpr - Taco expressions
karel - Gridworld agent
css - Website styling
...
Grammar
struct
generates
ProgramInstance
serialize()
Program String
trait GrammarExpander
mc - Monte Carlo Expander
wmc - Weighted Monte Carlo Expander
learned - Learned Expander
...
is called by
alled by
trait LanguageEmbedder
doc2vecdbow - Doc2Vec Embedder
ollama - Modern text-based embedders
...
enum FrontierDecisionWrapper
Modules to make decisions on which index on the frontier to expand
...
enum ProductionDecisionWrapper
fixed - one hypothesis class per language
variable - one hypothesis class per production
...

Language 1: TACO Expressions

The Tensor Algebra Compiler (Kjolstad et al. 2017) is a compiler to take tensor expressions (with optional scheduling directives) and create a C program that implements such operation.
One example:

$$A(x, y) = B(a, b) * C(c, d)$$

or

$$Y(y) = A(a, b) * X(x) + B(b)$$

Additionally, scheduling directives can be instantiated. These alter the loop structure of the computed kernel in various ways.
Some examples:

Additionally, scheduling directives can be instantiated. These alter the loop structure of the computed kernel in various ways.
Some examples:

- split() splits a loop into an outer and inner loop, based on some loop factor.

Additionally, scheduling directives can be instantiated. These alter the loop structure of the computed kernel in various ways.
Some examples:

- split() splits a loop into an outer and inner loop, based on some loop factor.
- reorder() switches the order of two loops.

Additionally, scheduling directives can be instantiated. These alter the loop structure of the computed kernel in various ways.
Some examples:

- split() splits a loop into an outer and inner loop, based on some loop factor.
- reorder() switches the order of two loops.
- unroll() takes a loop and forces all iterations of the loop to be explicitly defined.

$$y(l) = A(i,j) * x(k)$$ TACO compiles to

```
for (int32_t l = 0; l < y1_dimension; l++) {
  double yval = 0.0;
  for (int32_t i = 0; i < A1_dimension; i++) {
    for (int32_t j = 0; j < A2_dimension; j++) {
      int32_t jA = i * A2_dimension + j;
      for (int32_t k = 0; k < x1_dimension; k++) {
        yval += A_vals[jA] * x_vals[k];
      }
    }
  }
  y_vals[l] = yval;
}
```

$$y(l) = A(i,j) * x(k)$$
$$+ \ split(i, i0, i1, 5)$$ TACO compiles to

```
for (int32_t l = 0; l < y1_dimension; l++) {
    double yval = 0.0;
    for (int32_t i0 = 0; i0 < ((A1_dimension + 4) / 5); i0++) {
      for (int32_t i1 = 0; i1 < 5; i1++) {
        int32_t i = i0 * 5 + i1;
        if (i >= A1_dimension)
          continue;

        for (int32_t j = 0; j < A2_dimension; j++) {
          int32_t jA = i * A2_dimension + j;
          for (int32_t k = 0; k < x1_dimension; k++) {
            yval += A_vals[jA] * x_vals[k];
          }
        }
      }
    }
    y_vals[l] = yval;
  }
```

$$y(l) = A(i,j) * x(k)$$
$$+ \ split(i, i0, i1, 5) \ + \ unroll(i1, 5)$$ TACO compiles to

```
for (int32_t l = 0; l < y1_dimension; l++) {
  double yval = 0.0;
  for (int32_t i0 = 0; i0 < ((A1_dimension + 4) / 5); i0++) {
      #pragma unroll 5
      for (int32_t i1 = 0; i1 < 5; i1++) {
        int32_t i = i0 * 5 + i1;
        for (int32_t j = 0; j < A2_dimension; j++) {
          int32_t jA = i * A2_dimension + j;
          for (int32_t k = 0; k < x1_dimension; k++) {
            yval += A_vals[jA] * x_vals[k];
          }
        }
      }
    }
    y_vals[l] = yval;
  }
```

Language 3: Karel I

- Karel is a small DSL originally used in (Bunel et al. 2018) as their language of choice for expansion.
- Controls an agent in a gridworld, capable of picking up and placing flags at the position it currently occupies in the gridworld.
- Implemented as a sequence of instructions.

- def run(): ifelse(markersPresent()): turnRight() else: move()
- def run(): while(markersPresent()): pickMarker()

Language 4: CSS

- Common language for styling elements in webpages.
- Structured as blocks of properties that are applied to *selected* objects in the DOM.

```
<selector> <selector> ... {
  <property>: <setting>,
  <property>: <setting>,
                ...
}
...
```

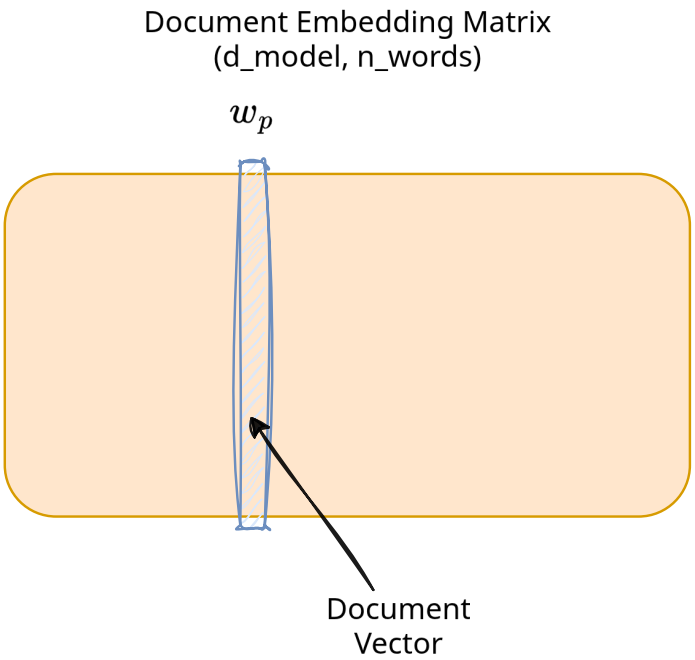
Doc2vec (Le and Mikolov 2014) and the variant graph2vec (Narayanan et al. 2017) are the canonical embedding framework used for experiments.

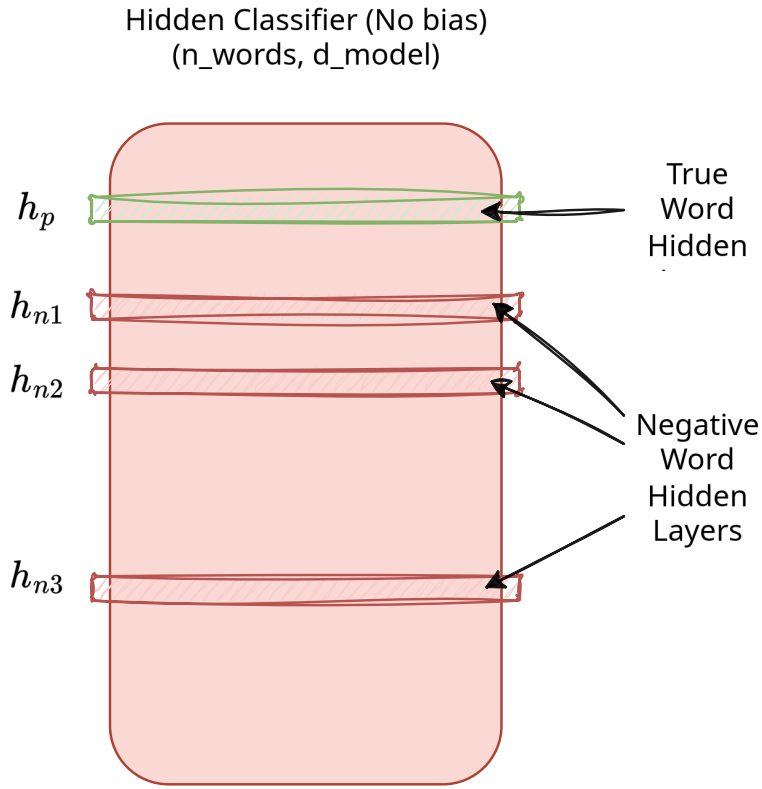
Doc2vec (Le and Mikolov 2014) and the variant graph2vec (Narayanan et al. 2017) are the canonical embedding framework used for experiments.
Doc2vec looks to predict words from a document to learn embeddings.

Doc2Vec/Graph2Vec

Doc2vec (Le and Mikolov 2014) and the variant graph2vec (Narayanan et al. 2017) are the canonical embedding framework used for experiments.
Doc2vec looks to predict words from a document to learn embeddings.
Graph2vec predicts graph labels instead of words, derived from the Weisfeiler-Lehman graph kernel.

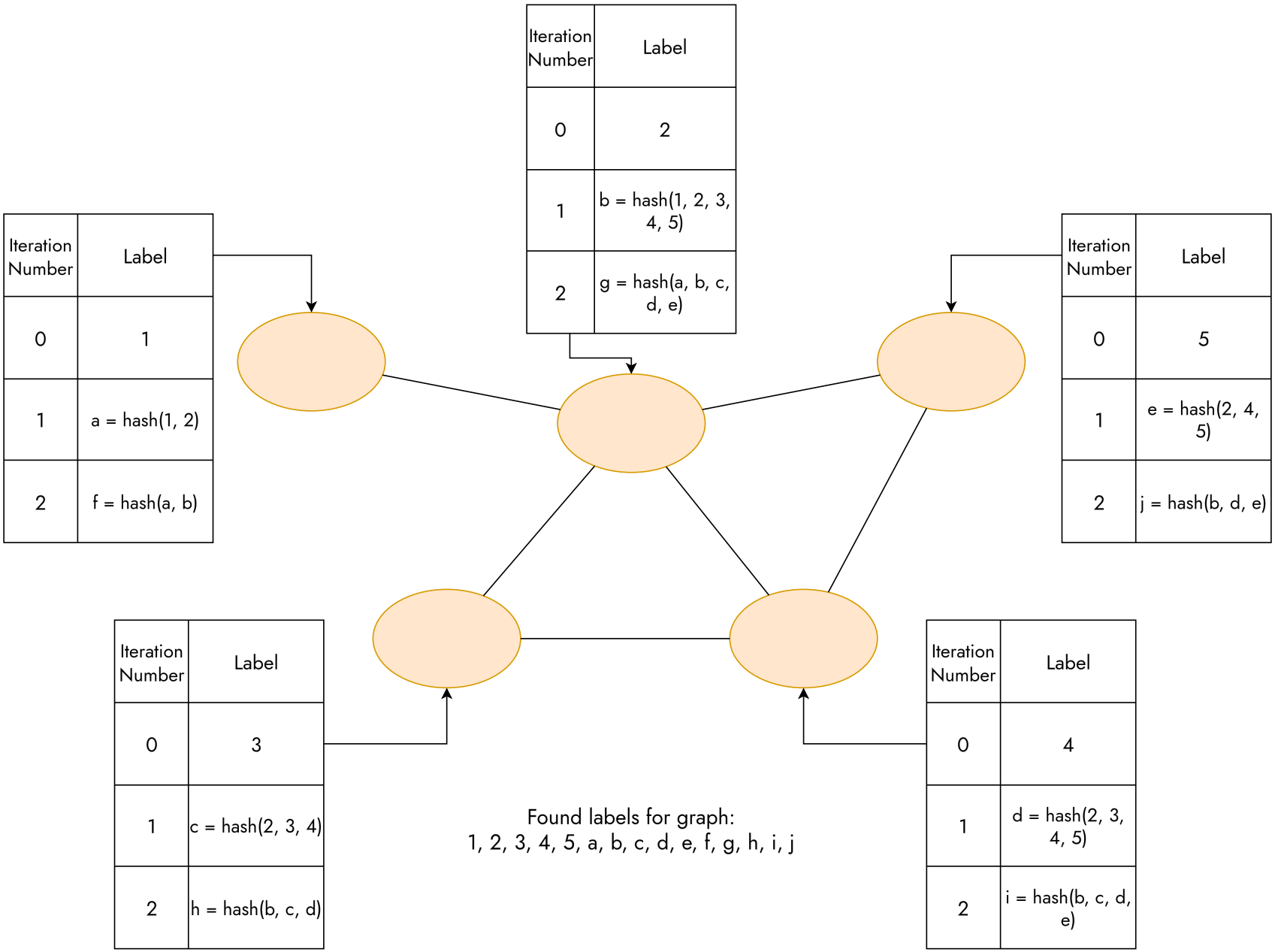
Doc2Vec Objective





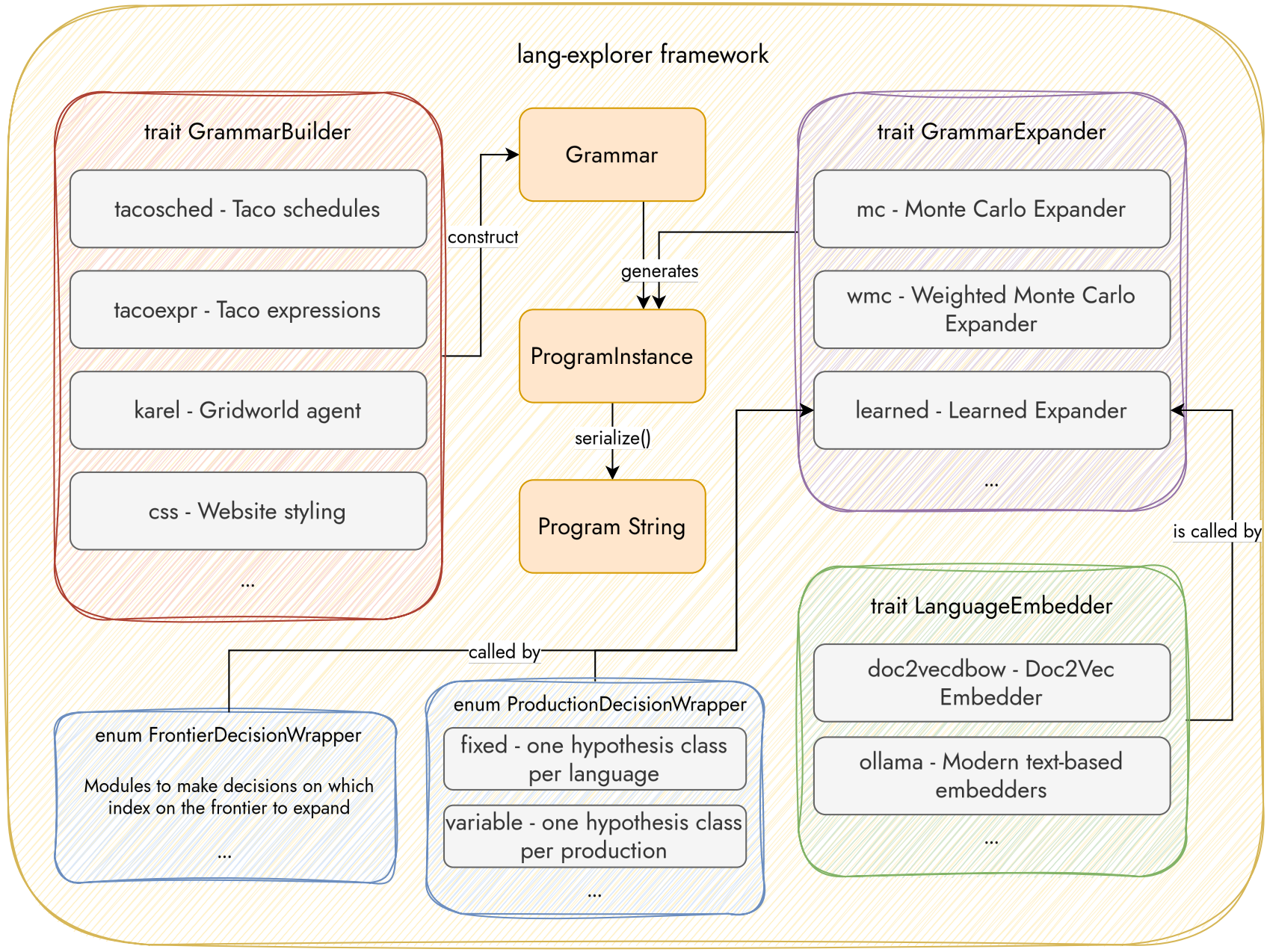
$$\mathcal{L} = -\log \sigma(w_p \cdot h_p) - \left(\log \sigma(-w_p \cdot h_{n1}) + \log \sigma(-w_p \cdot h_{n2}) + \log \sigma(-w_p \cdot h_{n3})\right)$$

WL-Kernel Example



My work: Lang-Explorer Framework

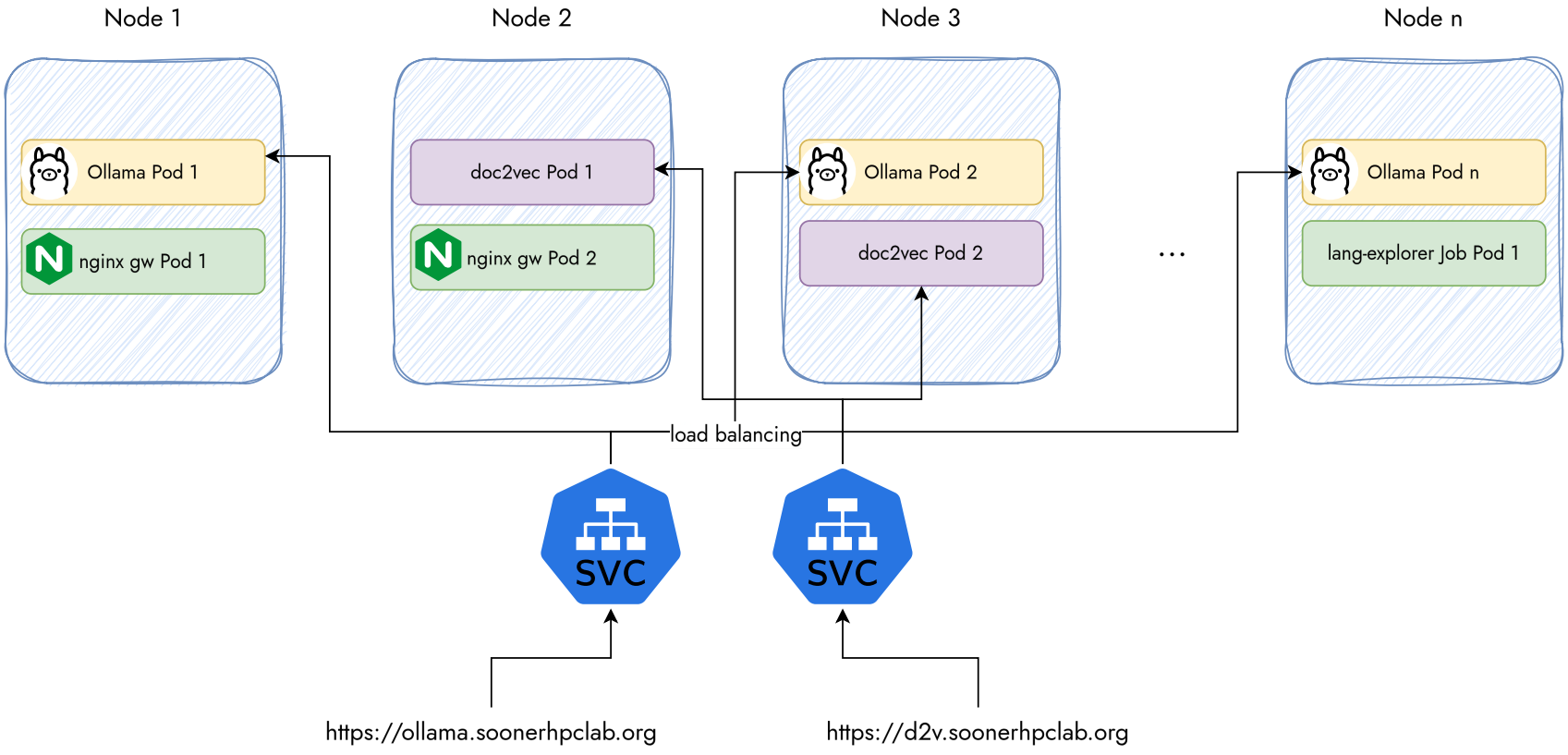
This research necessitated building a software system to build programs, embed them, and have as many interfaces as possible for extending the system further.
Chose to implement a system in Rust, with machine learning components using Burn (Simard et al. 2024).


lang-explorer framework
trait GrammarBuilder
tacosched - Taco schedules
tacoexpr - Taco expressions
karel - Gridworld agent
css - Website styling
...
construct
Grammar
generates
ProgramInstance
serialize()
Program String
trait GrammarExpander
mc - Monte Carlo Expander
wmc - Weighted Monte Carlo Expander
learned - Learned Expander
...
is called by
trait LanguageEmbedder
doc2vecdbow - Doc2Vec Embedder
ollama - Modern text-based embedders
...
called by
enum FrontierDecisionWrapper
Modules to make decisions on which index on the frontier to expand
...
enum ProductionDecisionWrapper
fixed - one hypothesis class per language
variable - one hypothesis class per production
...

Additional Infrastructure

Additional infrastructure was required to make these experiments work that took a good bit of time to setup. This included:

- A small Kubernetes cluster.
- Infrastructure on said cluster for traffic routing, TLS-termination, some basic monitoring.

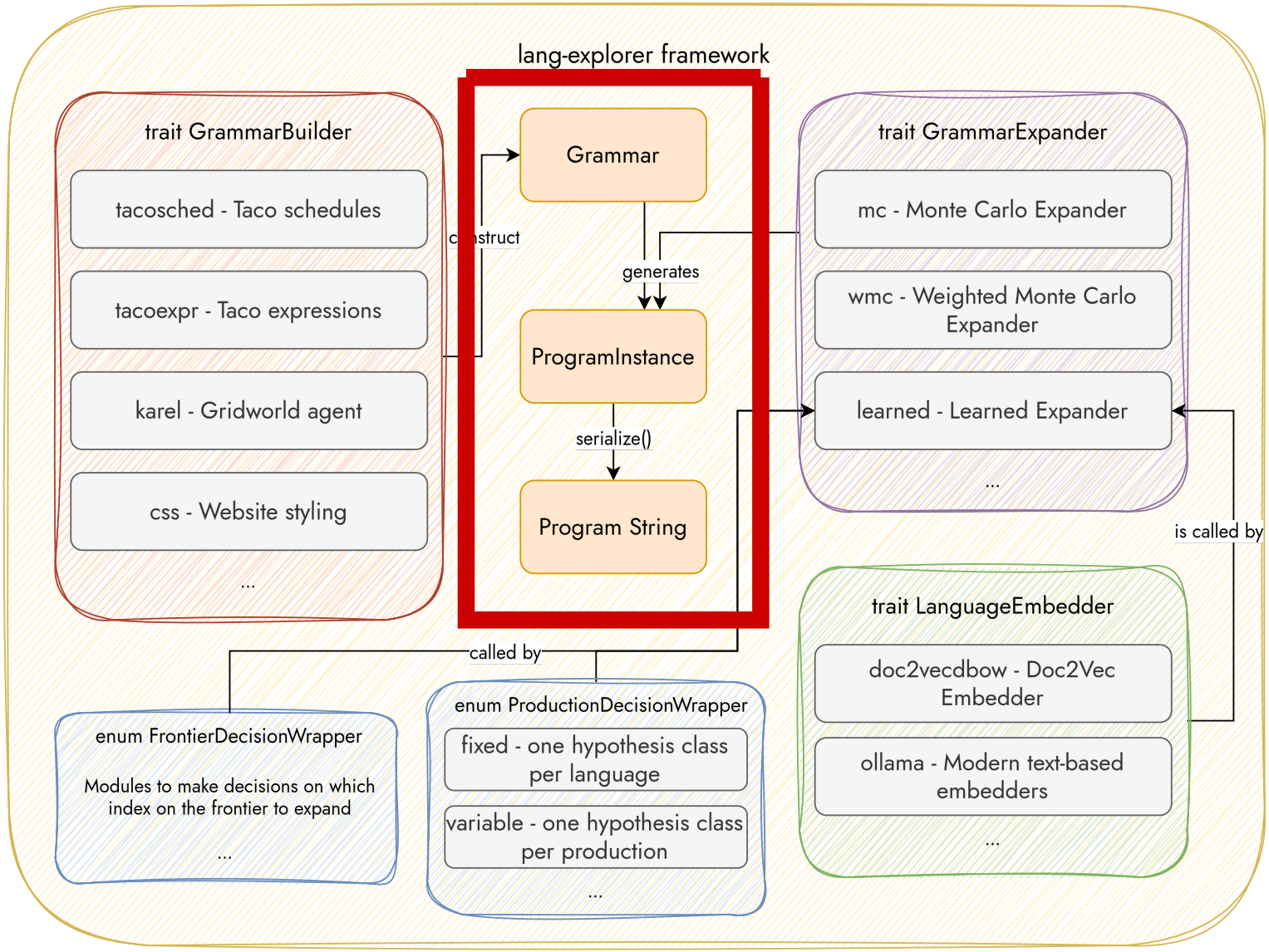

Node 1
Node 2
Node 3
Node n
Ollama Pod 1
nginx gw Pod 1
doc2vec Pod 1
nginx gw Pod 2
Ollama Pod 2
doc2vec Pod 2
...
Ollama Pod n
lang-explorer Job Pod 1
load balancing
SVC
SVC
https://ollama.soonerhpclab.org
https://d2v.soonerhpclab.org

Experiments

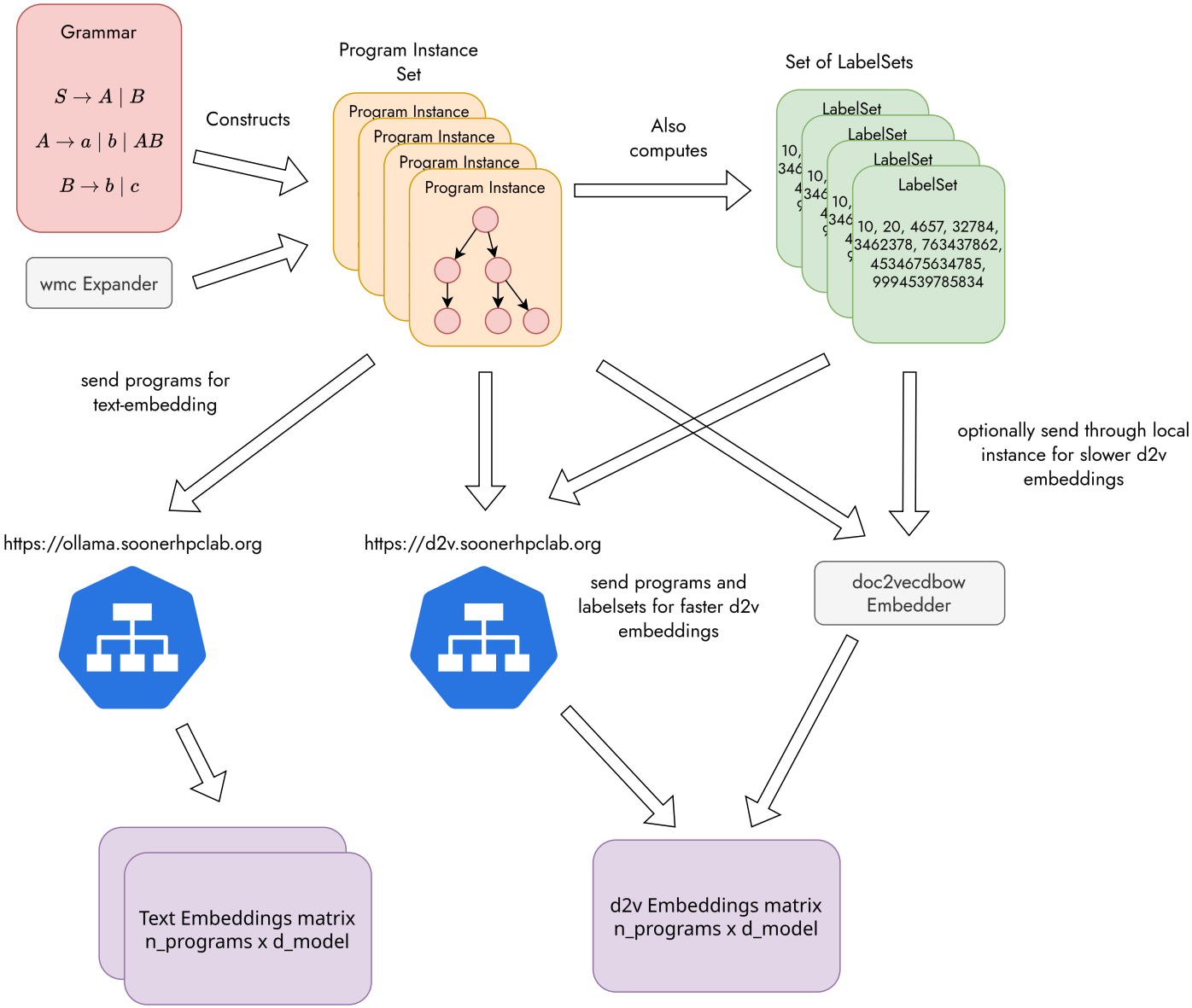
I performed two kinds of experiments with the generated datasets

- Qualitative analysis of the structure of the spaces with t-SNE and some nearest neighbor analysis.
- Quantitative analysis using a similarity metric to compare embedding similarity with graph similarity.

Experimental Setup I



Experimental Setup II



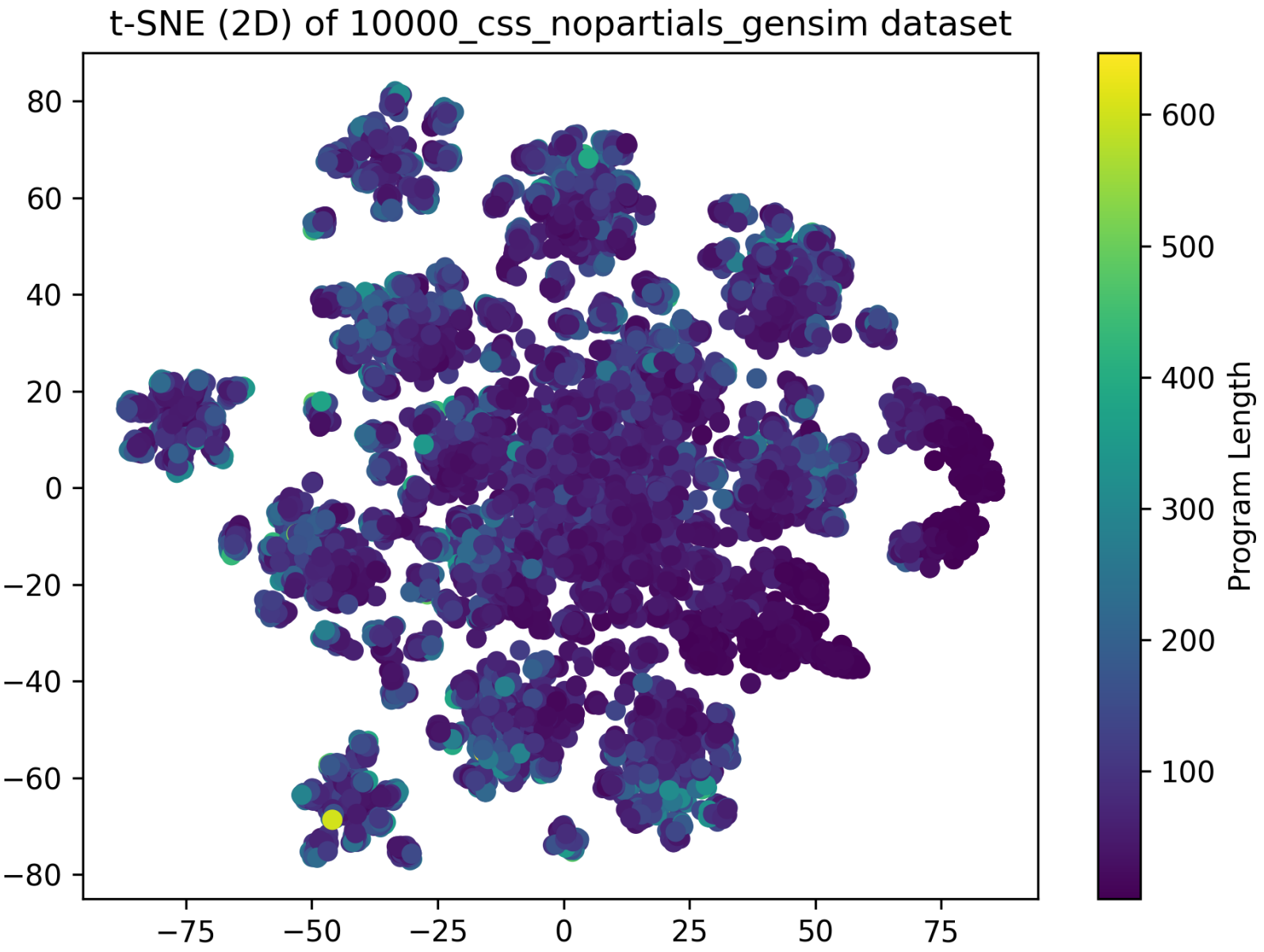
Program Embeddings t-SNE

If one embeds programs using doc2vec, what does the space of embeddings look like?

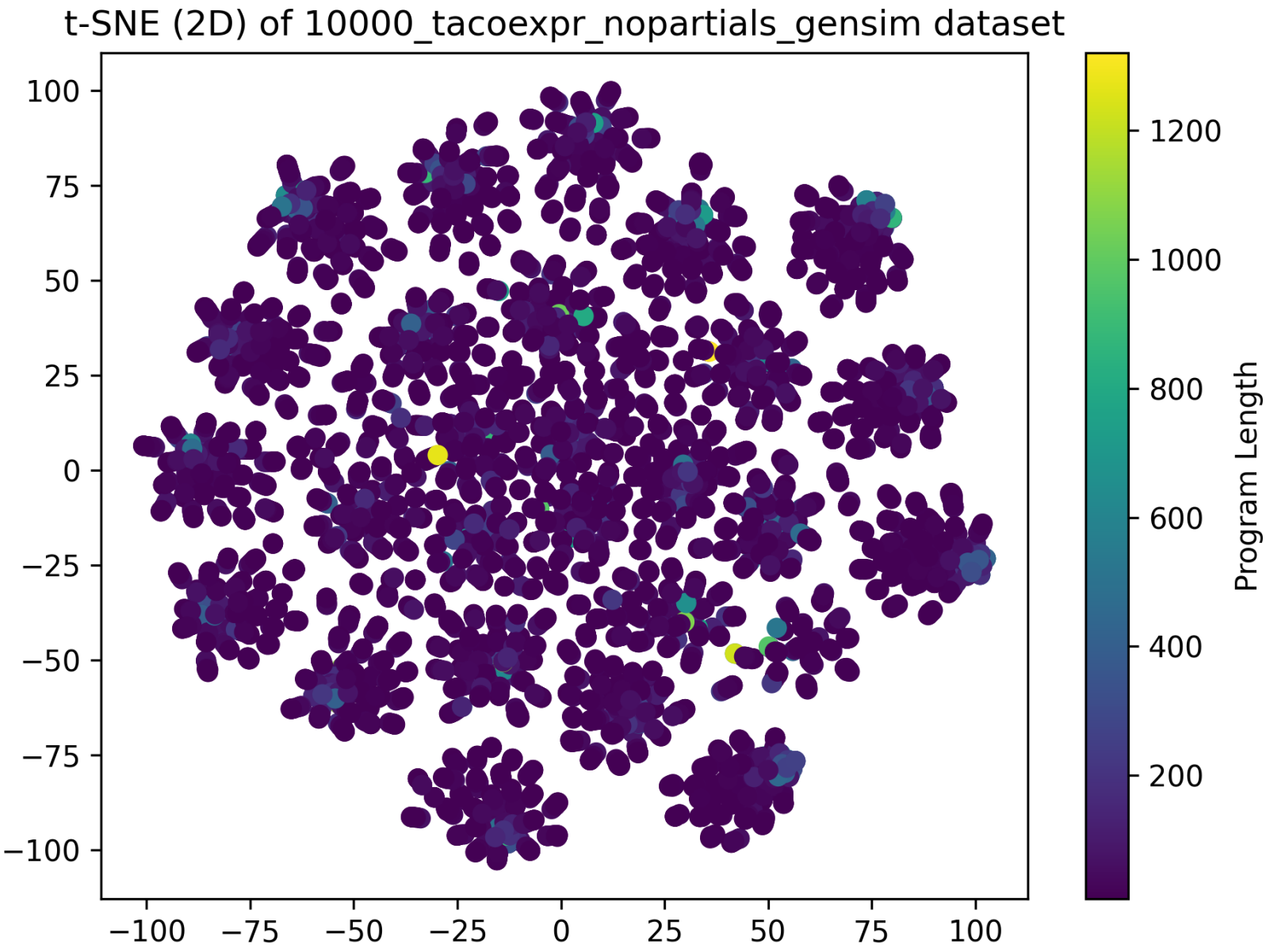
Program Embeddings t-SNE

If one embeds programs using doc2vec, what does the space of embeddings look like?
This can be approximated using T-Stochastic Neighbor Embeddings (Maaten and Hinton 2008).

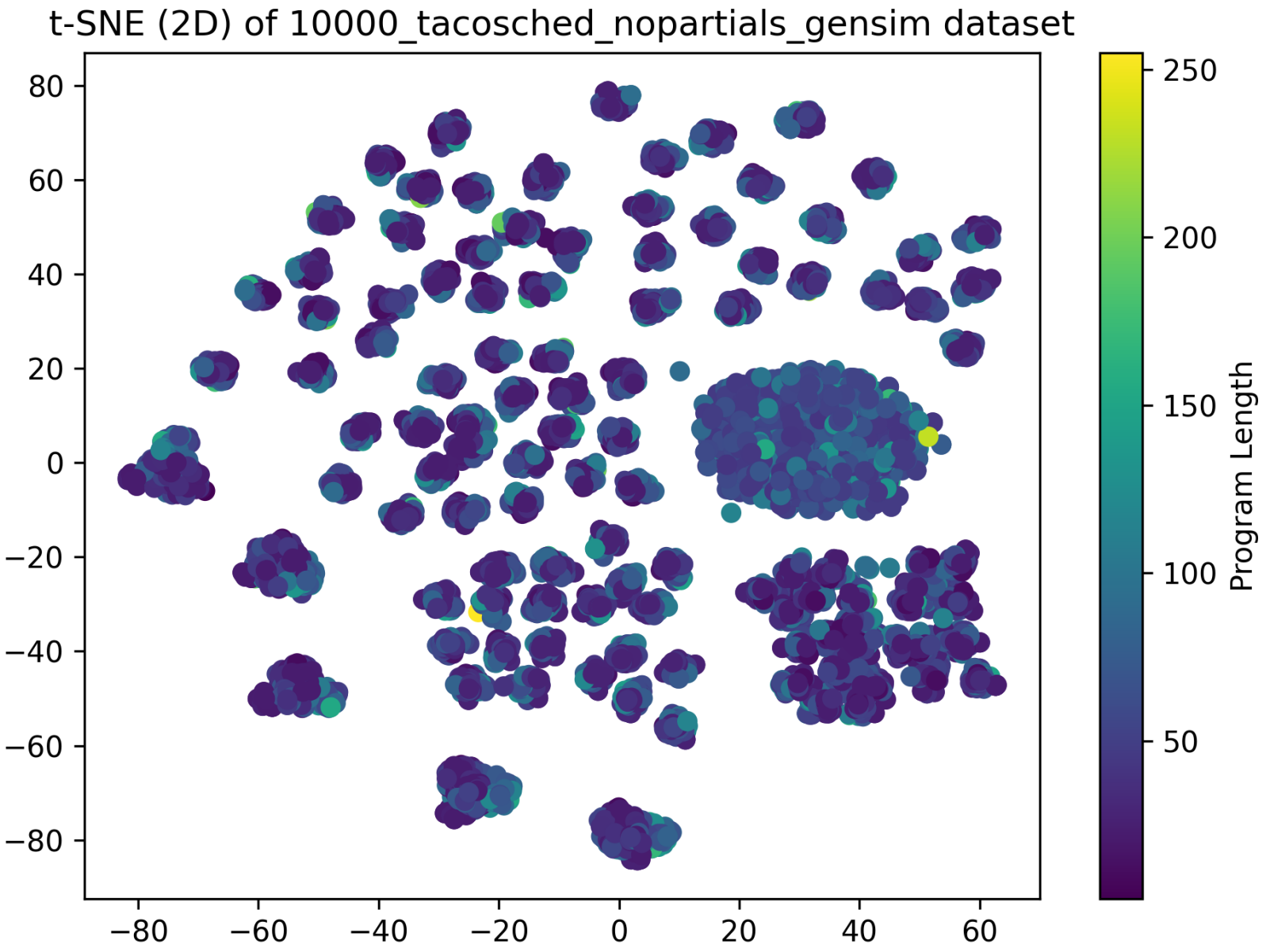
CSS t-SNE


t-SNE (2D) of 10000_css_nopartials_gensim dataset
80
60
40
20
0
−20
−40
−60
−80
−75
−50
−25
0
25
50
75
600
500
400
300
200
100
Program Length

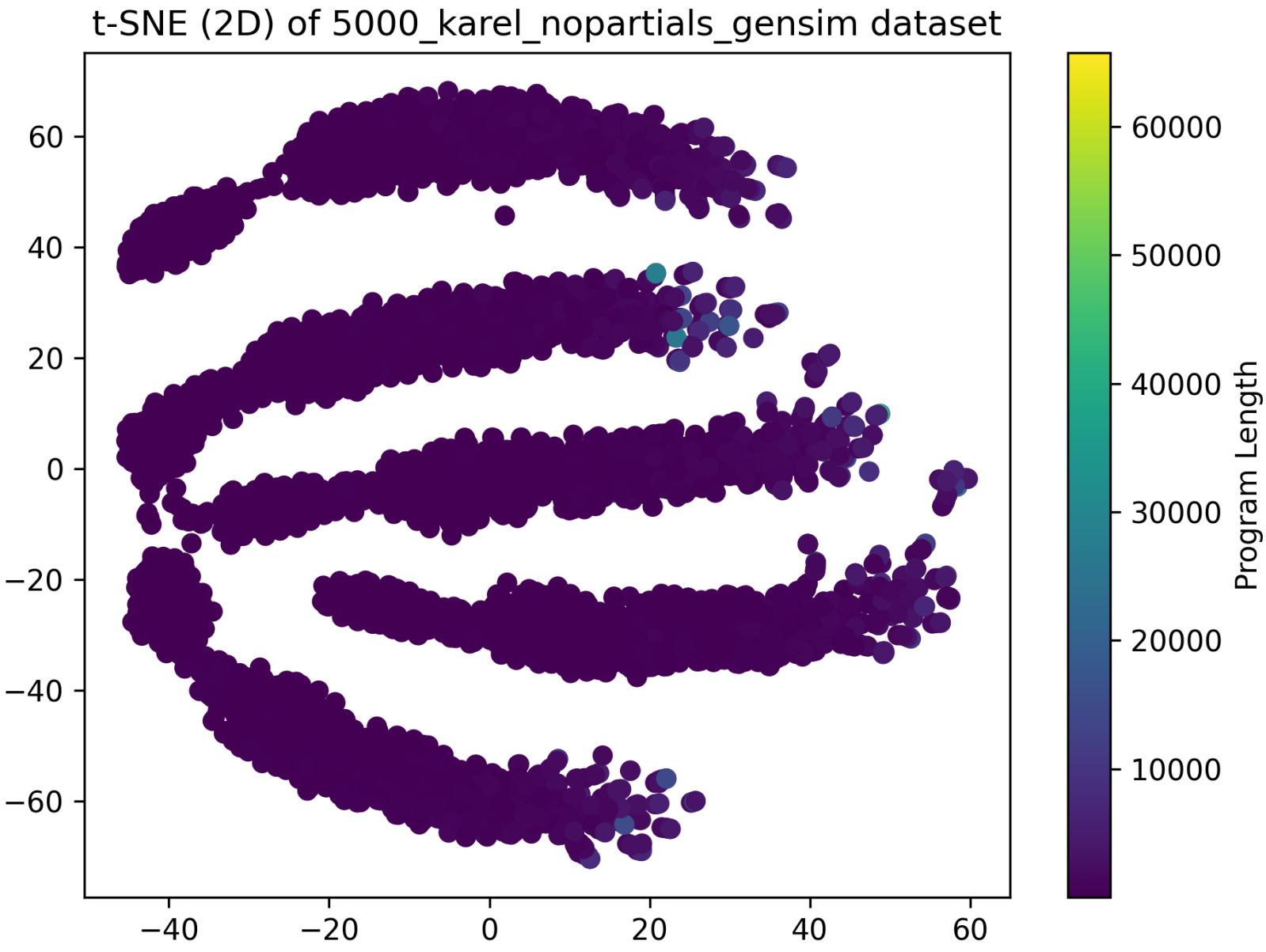
TACO Expression t-SNE



TACO Schedule t-SNE



Karel t-SNE

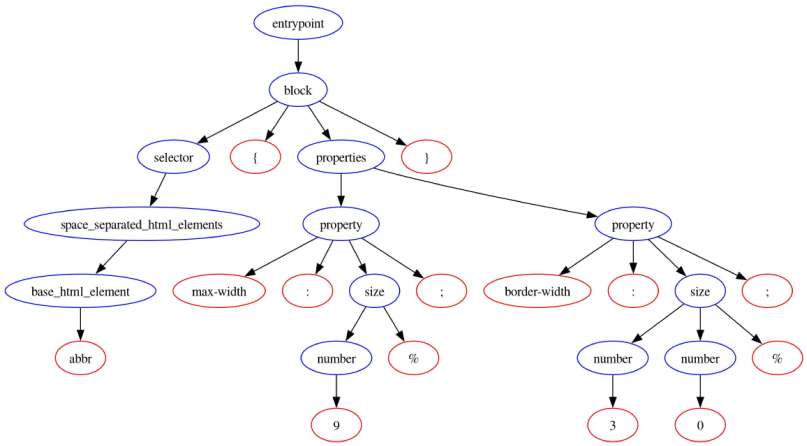

t-SNE (2D) of 5000_karel_nopartials_gensim dataset
60
40
20
0
−20
−40
−60
−40
−20
0
20
40
60
60000
50000
40000
30000
20000
10000
Program Length

Program Nearest Neighbors

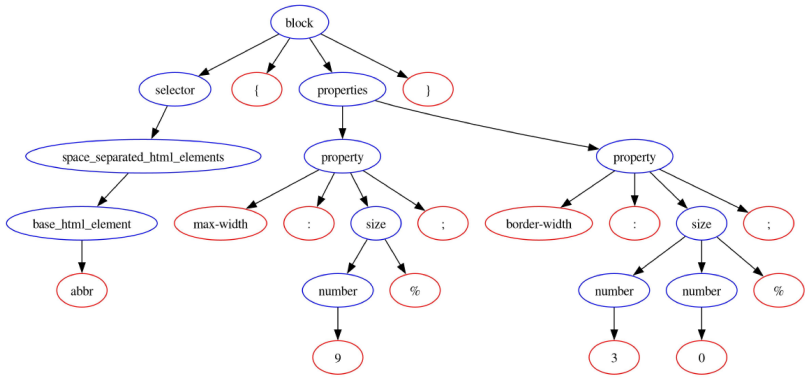
How similar do programs of nearby embeddings look to each other. Are they purely structurally similar? Is there semantic similarity?

CSS Nearest Neighbors Example I

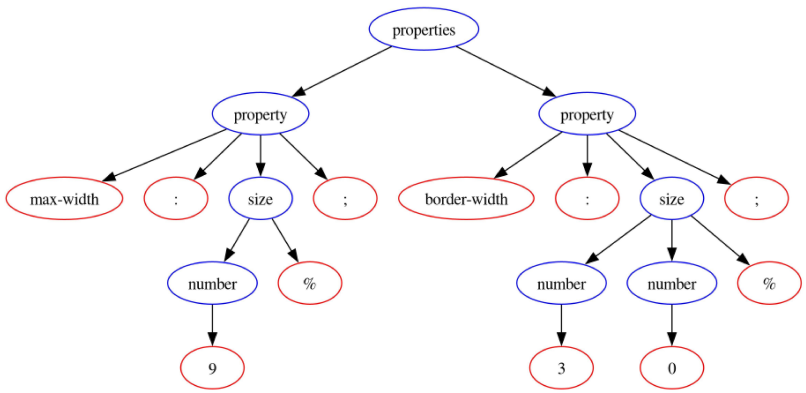
Original Program



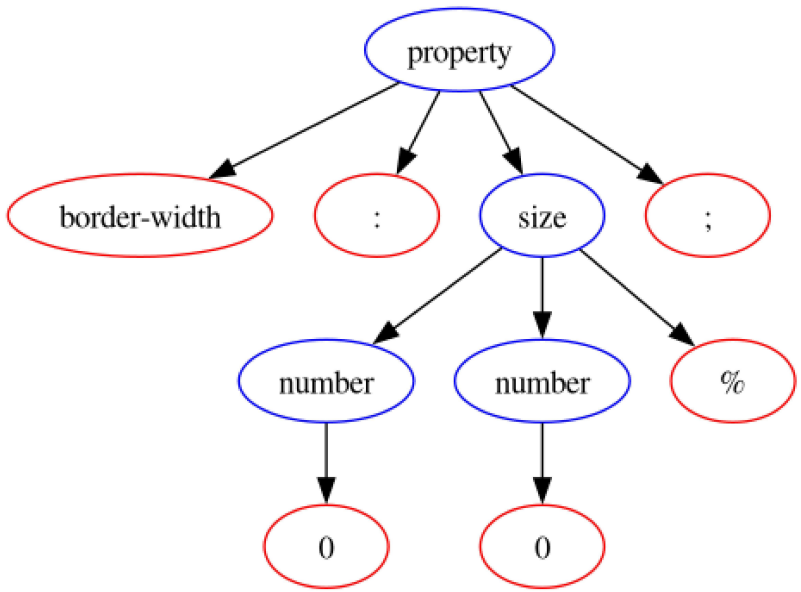
1st NN



2nd NN

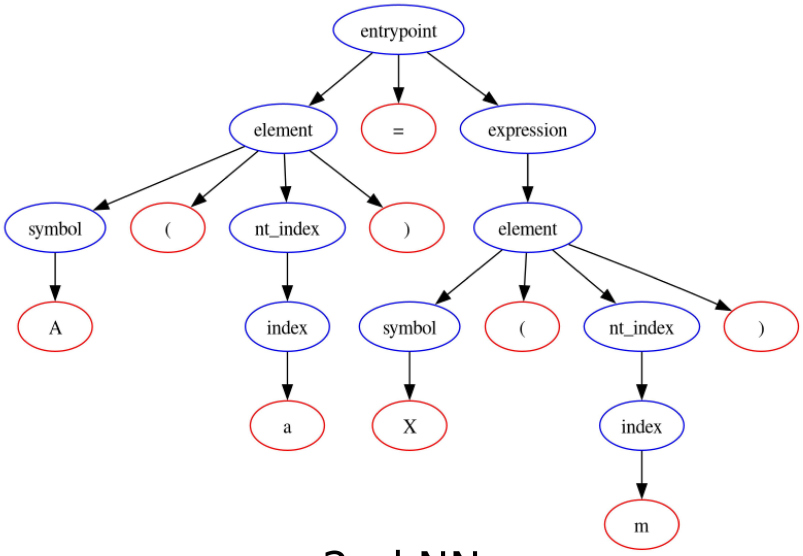


3rd NN

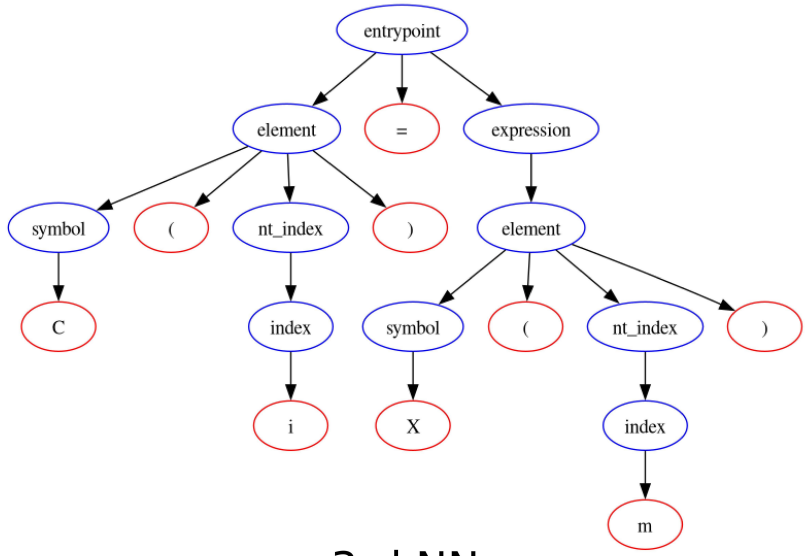


TACO Expression Nearest Neighbor Example I

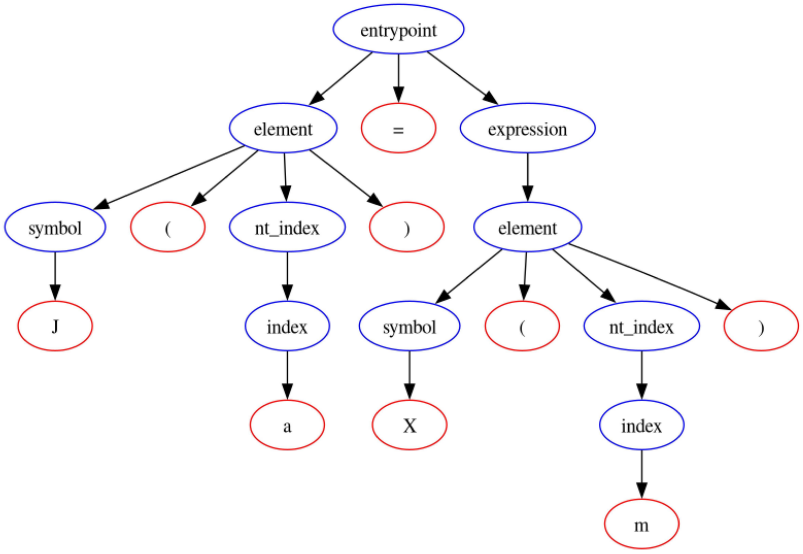
Original Program



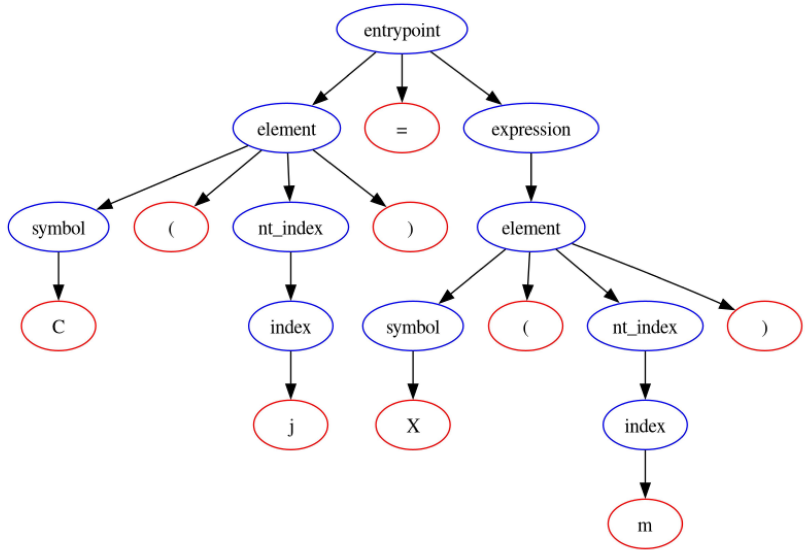
1st NN

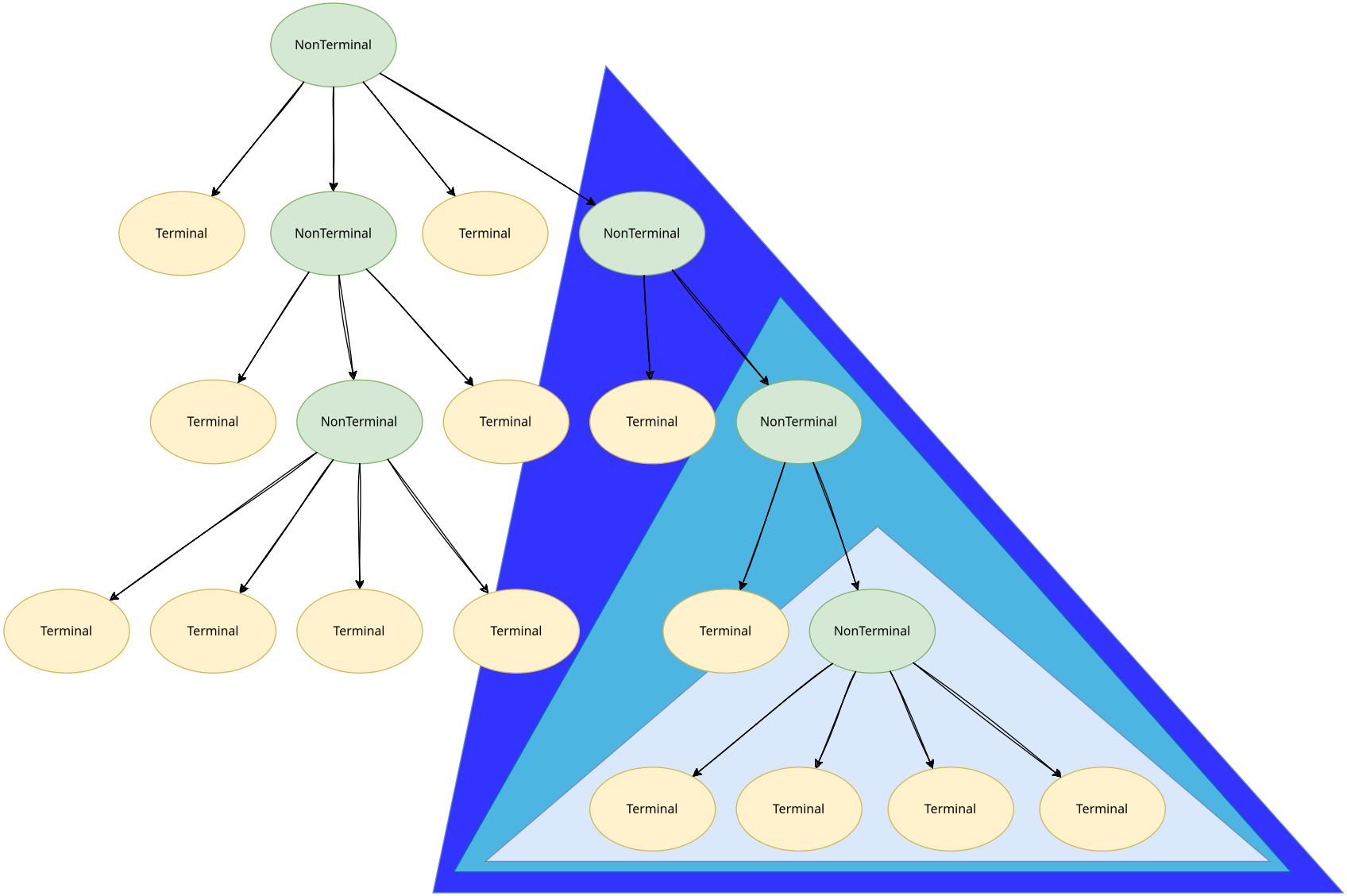


2nd NN



3rd NN




NonTerminal
Terminal
NonTerminal
Terminal
NonTerminal
Terminal
NonTerminal
Terminal
Terminal
NonTerminal
Terminal
Terminal
Terminal
Terminal
Terminal
Terminal
NonTerminal
Terminal
Terminal
Terminal
Terminal

Quantitative Analysis

Qualitative analysis is interesting, but I want something concrete.

Quantitative Analysis

Qualitative analysis is interesting, but I want something concrete. Can we compute some similarity between ASTs themselves, and then compare that similarity to embedding similarity for the same programs?

Feature Labels:

$$a_1 = \{1, 1, 1, 2, 2, 3, 4, 5, 6, 7, 8, 9, 9\}$$

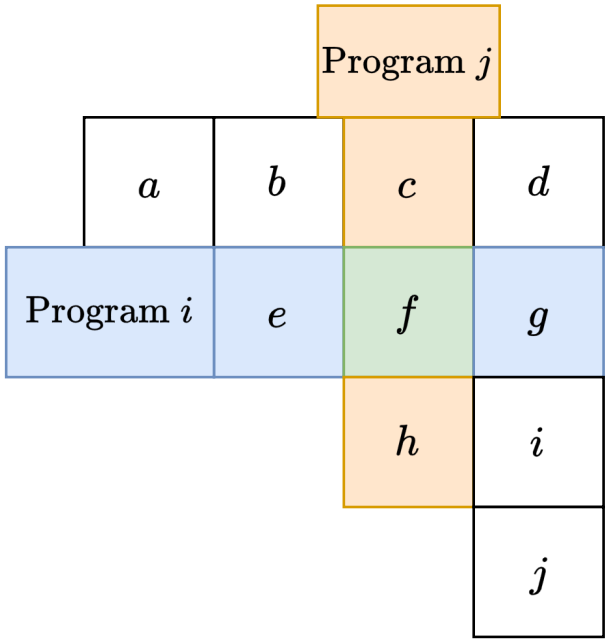
$$a_2 = \{1, 2, 3, 5, 6, 6, 6, 7, 8, 8, 9, 10, 11\}$$

	1	2	3	4	5	6	7	8	9	10	11
v_1	3	2	1	1	1	1	1	1	2	0	0
v_2	1	1	1	0	1	3	1	2	1	1	1

$$||v_1 - v_2||_2 = \sqrt{14}$$

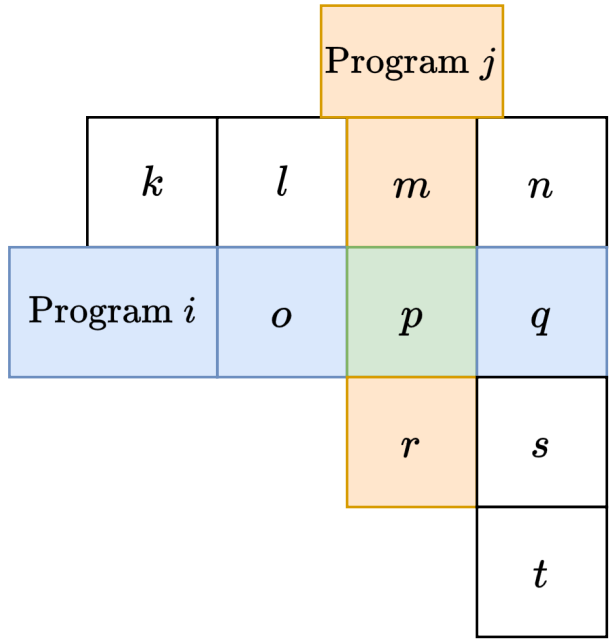
AST Similarities

$\mathcal{A}$



Embedding Similarities

$\mathcal{E}$



Average Similarity:

$$\frac{1}{|\mathcal{A}|} \sum_{a_i \in \mathcal{A}, e_i \in \mathcal{E}} |a_i - e_i| \rightarrow \frac{1}{10} \left(|a-k| + |b-l| + \ldots + |i-s| + |j-t| \right)$$

The text-embedding models I chose to compare against were:

- Nomic-AI's Text Embedding model (Nussbaum et al. 2024)
- Mixed-Bread AI's Text Embedding Model (Lee et al. 2024)
- Snowflake's Arctic Embedding model V1 (Merrick et al. 2024)

As will be shown in histograms of these similarity scores, there can be a large difference simply because the different distributions are not normalized.

As will be shown in histograms of these similarity scores, there can be a large difference simply because the different distributions are not normalized.
This can be mitigated with Min-Max normalization:

$$x_{\text{new}} = \frac{x_{\text{old}} - \min(x)}{\max(x) - \min(x)}$$

Pairwise Similarity Scores I

Distribution	Simple Average	Normalized Simple Average
doc2vecgensim	**11.787715**	**0.127106**
nomic-embed-text	4.991453	0.167829
mxbai-embed-large	3.461594	0.149734
snowflake-arctic-embed:137m	4.204036	0.136181

Table: TACO Schedule pairwise similarity scores, lower is better.

Distribution	Simple Average	Normalized Simple Average
doc2vecgensim	**6.520979**	**0.090355**
nomic-embed-text	3.453399	0.128993
mxbai-embed-large	2.641928	0.125813
snowflake-arctic-embed:137m	3.706221	0.135822

Table: TACO Expression pairwise similarity scores, lower is better.

Pairwise Similarity Scores II

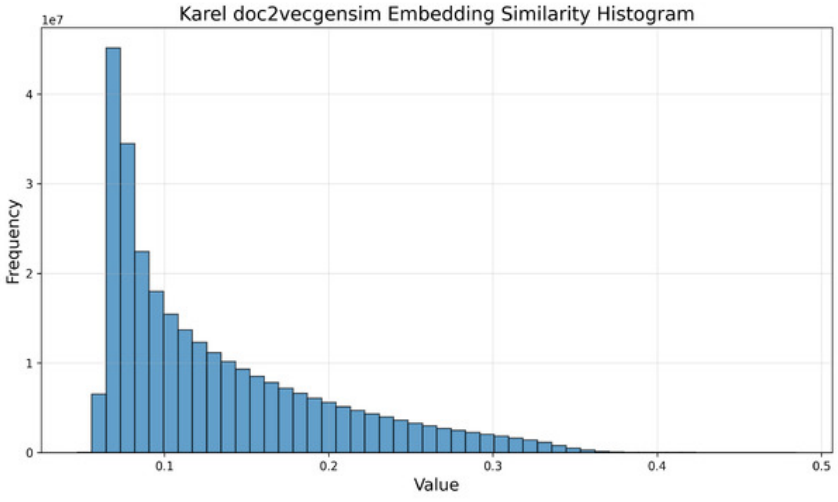
Distribution	Simple Average	Normalized Simple Average
doc2vecgensim	**12.883659**	**0.388166**
nomic-embed-text	3.568656	0.296640
mxbai-embed-large	4.214493	0.276651
snowflake-arctic-embed:137m	6.096455	0.280733

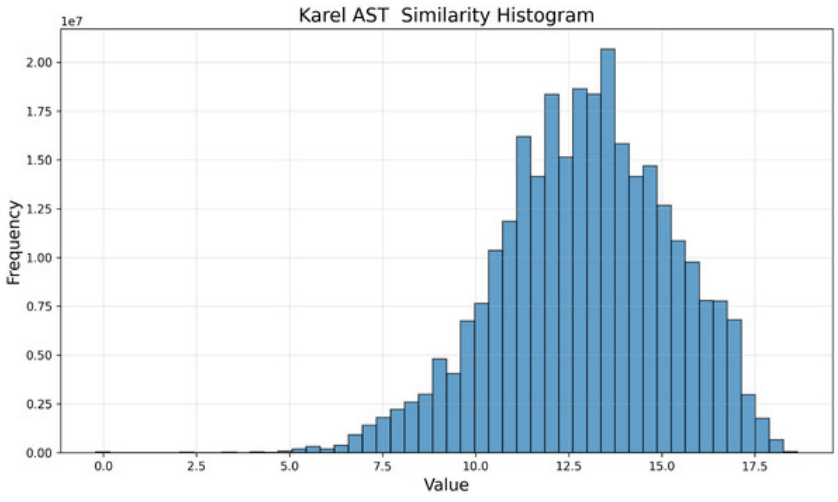
Table: Karel pairwise similarity scores, lower is better.

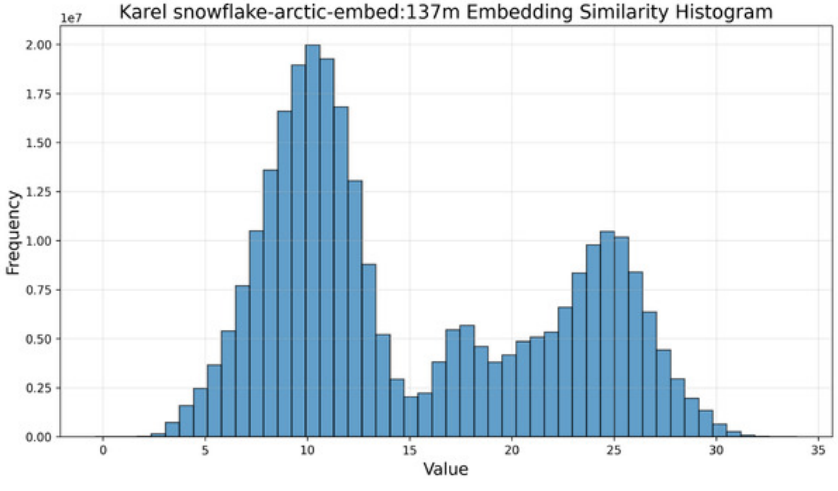
Distribution	Simple Average	Normalized Simple Average
doc2vecgensim	**5.943372**	**0.167782**
nomic-embed-text	3.857277	0.121263
mxbai-embed-large	2.344701	0.120693
snowflake-arctic-embed:137m	3.856680	0.117318

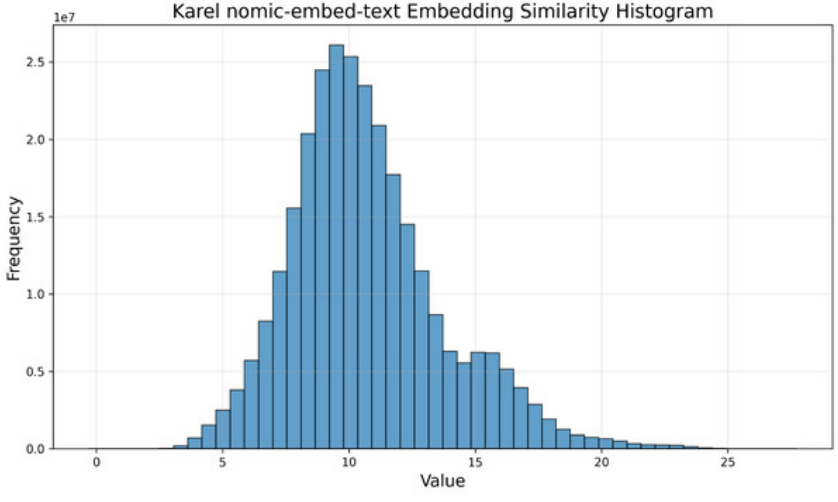
Table: CSS pairwise similarity scores, lower is better.

Pairwise Similarity Distributions I

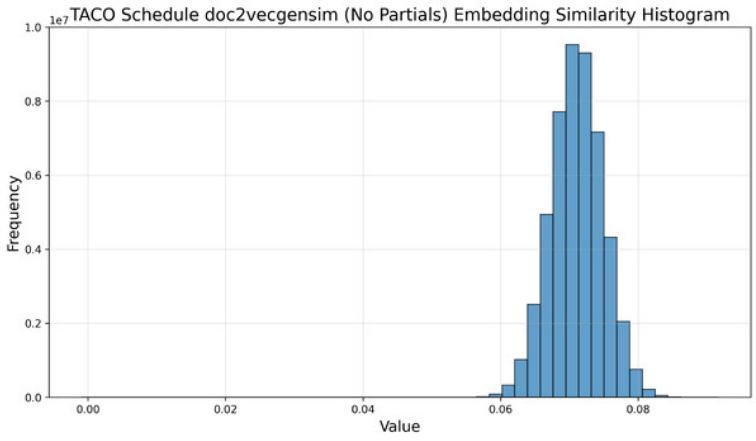


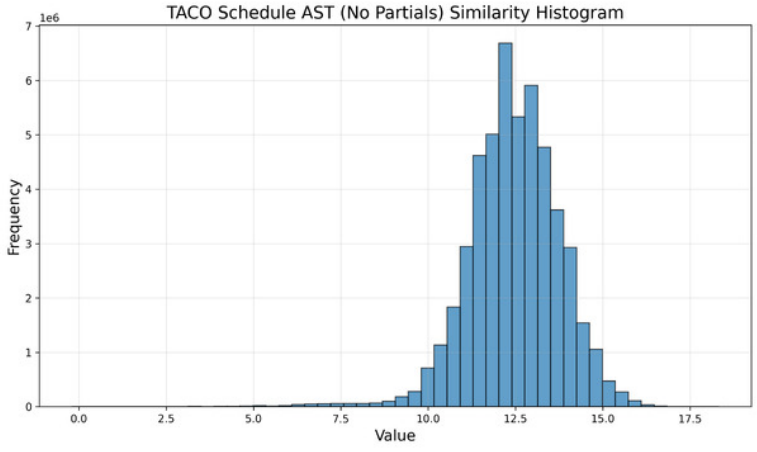


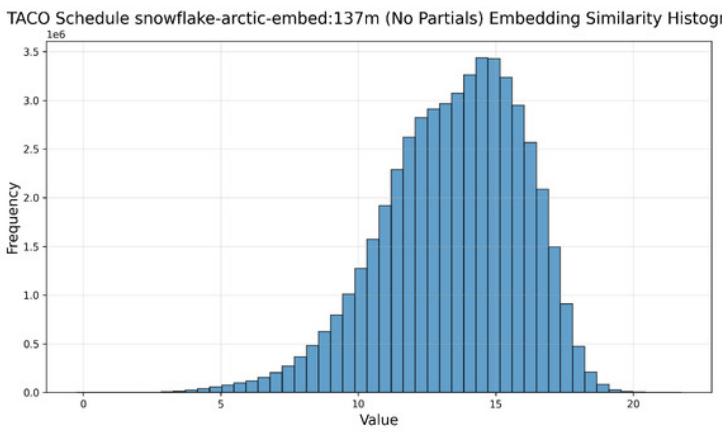


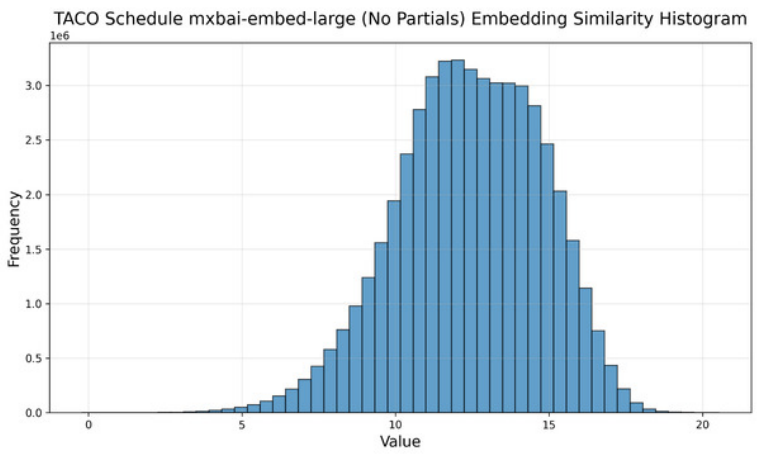


Pairwise Similarity Distributions II

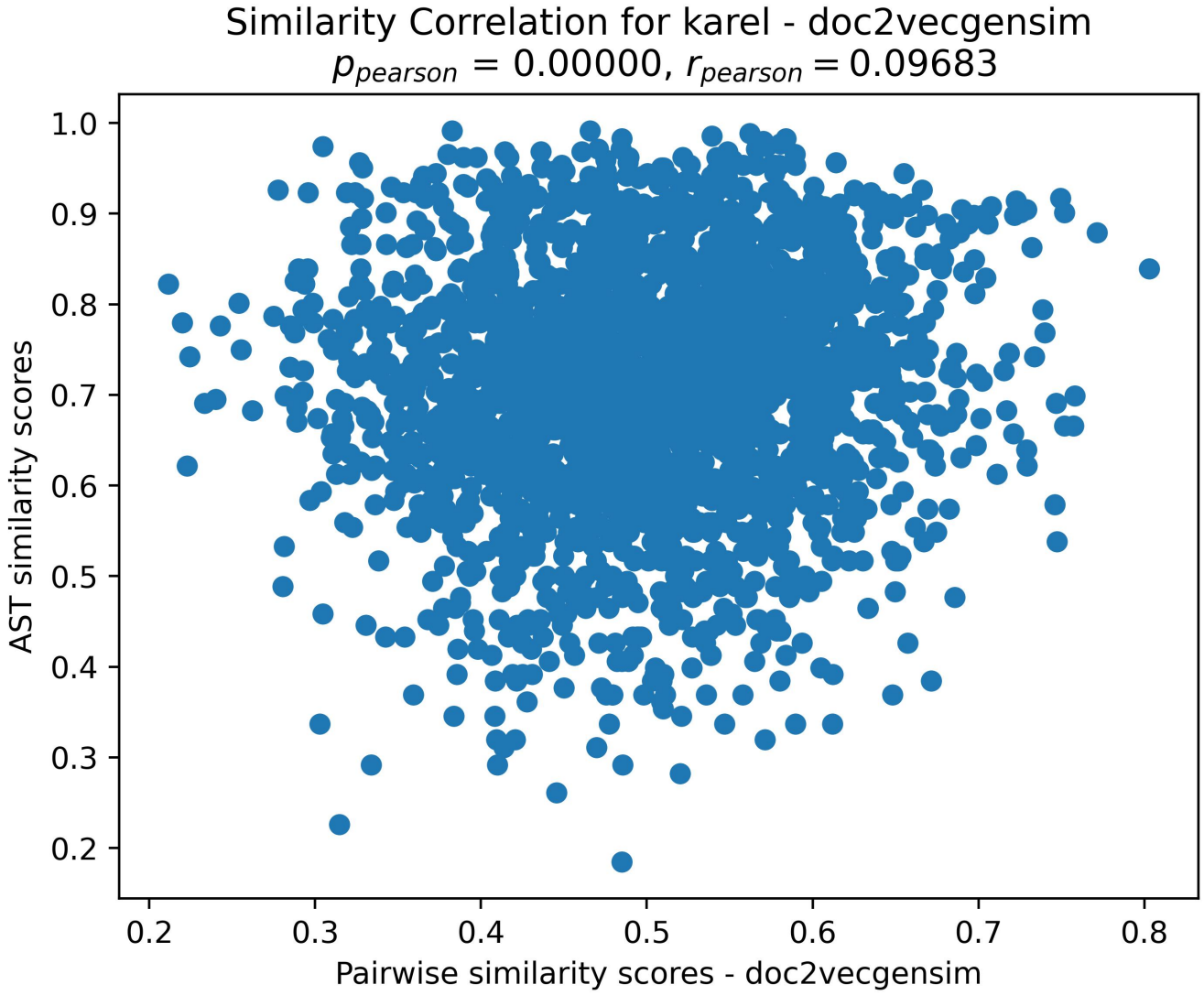




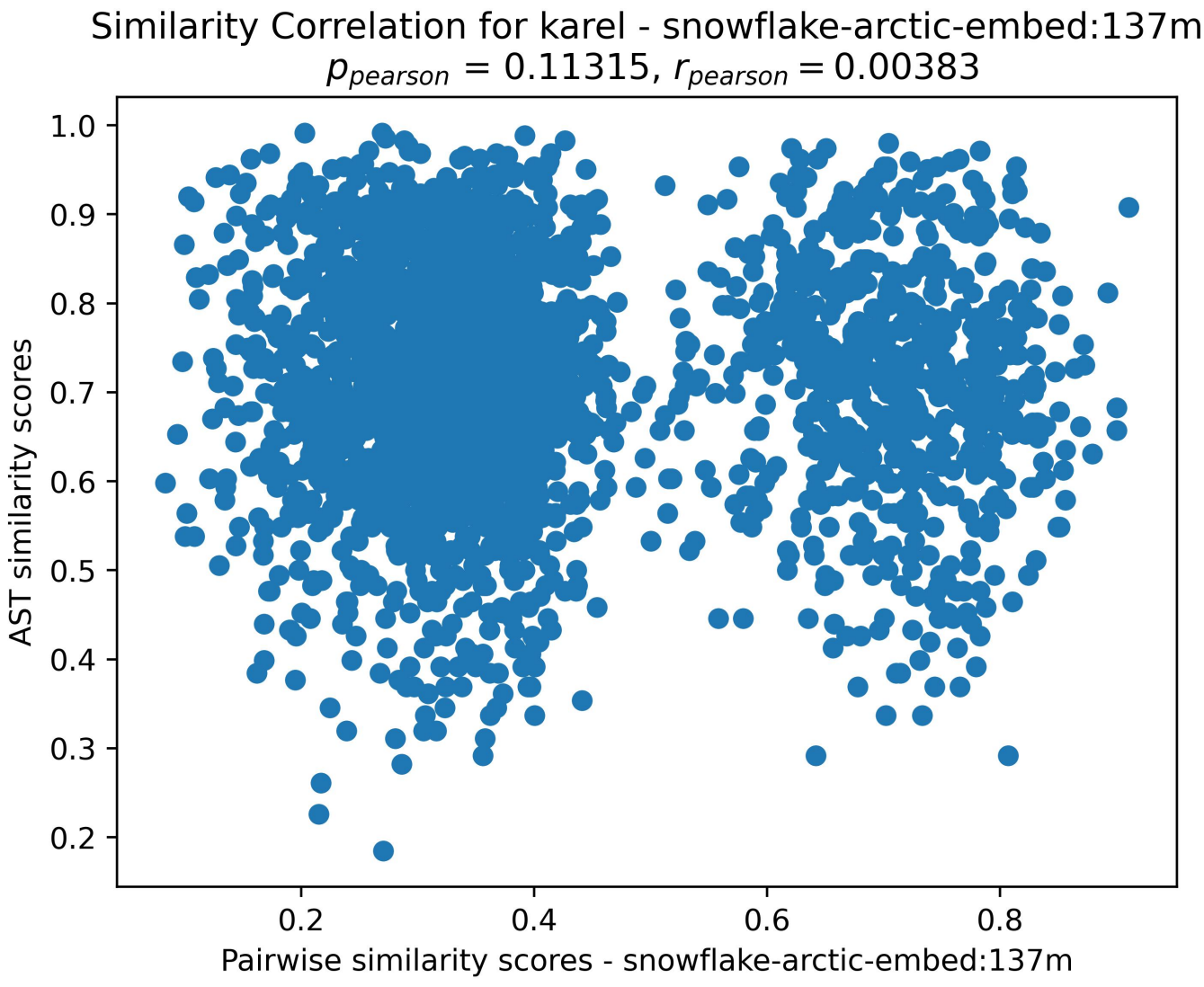




Pairwise Similarity Correlation I



Pairwise Similarity Correlation II



Discussion

Main experimental results:

- t-SNE plots for embedding spaces

Discussion

Main experimental results:

- t-SNE plots for embedding spaces
- Nearest neighbor plots between programs

Discussion

Main experimental results:

- t-SNE plots for embedding spaces
- Nearest neighbor plots between programs
- Quantitative pairwise similarity measures for embeddings and ASTs

Discussion

Main experimental results:

- t-SNE plots for embedding spaces
- Nearest neighbor plots between programs
- Quantitative pairwise similarity measures for embeddings and ASTs
- Distributions of pairwise similarities

Discussion

Main experimental results:

- t-SNE plots for embedding spaces
- Nearest neighbor plots between programs
- Quantitative pairwise similarity measures for embeddings and ASTs
- Distributions of pairwise similarities
- Correlation analysis of similarity measures

Limitations

- Synthetic datasets were used, which are small samples of an infinite space

Limitations

- Synthetic datasets were used, which are small samples of an infinite space
- t-SNE is an approximate visualization method, lots of information is ultimately lost

Limitations

- Synthetic datasets were used, which are small samples of an infinite space
- t-SNE is an approximate visualization method, lots of information is ultimately lost
- AST structure is not inextricably linked to semantics; comparing based on this can be useful in some languages, but not in others.

Future Work

- Rework Doc2vec for better performance

Future Work

- Rework Doc2vec for better performance
- GraphMAE (Hou, He, et al. 2023; Hou, Liu, et al. 2022) implementation

Future Work

- Rework Doc2vec for better performance
- GraphMAE (Hou, He, et al. 2023; Hou, Liu, et al. 2022) implementation
- Scale up experiment sizes on more nodes.

Future Work

- Rework Doc2vec for better performance
- GraphMAE (Hou, He, et al. 2023; Hou, Liu, et al. 2022) implementation
- Scale up experiment sizes on more nodes.
- Automate hyper-parameter and neural architecture search (White et al. 2023)

Future Work

- Rework Doc2vec for better performance
- GraphMAE (Hou, He, et al. 2023; Hou, Liu, et al. 2022) implementation
- Scale up experiment sizes on more nodes.
- Automate hyper-parameter and neural architecture search (White et al. 2023)
- Add support for more languages, including other general design languages like TLA+ (Lamport 1994)

Future Work

- Rework Doc2vec for better performance
- GraphMAE (Hou, He, et al. 2023; Hou, Liu, et al. 2022) implementation
- Scale up experiment sizes on more nodes.
- Automate hyper-parameter and neural architecture search (White et al. 2023)
- Add support for more languages, including other general design languages like TLA+ (Lamport 1994)
- RL implementations within Learned Expander, evaluation services

Bibliography I


Bunel, Rudy et al. (2018). "Leveraging Grammar and Reinforcement Learning for Neural Program Synthesis". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=H1Xw62kRZ.


Hou, Zhenyu, Yufei He, et al. (2023). "GraphMAE2: A Decoding-Enhanced Masked Self-Supervised Graph Learner". In: *Proceedings of the ACM Web Conference 2023*. WWW '23. Austin, TX, USA: Association for Computing Machinery, pp. 737–746. ISBN: 9781450394161. DOI: 10.1145/3543507.3583379. URL: https://doi.org/10.1145/3543507.3583379.


Hou, Zhenyu, Xiao Liu, et al. (2022). "GraphMAE: Self-Supervised Masked Graph Autoencoders". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. KDD '22. Washington DC, USA: Association for Computing Machinery, pp. 594–604. ISBN: 9781450393850. DOI: 10.1145/3534678.3539321. URL: https://doi.org/10.1145/3534678.3539321.


Kjolstad, Fredrik et al. (Oct. 2017). "The tensor algebra compiler". In: *Proc. ACM Program. Lang.* 1.OOPSLA. DOI: 10.1145/3133901. URL: https://doi.org/10.1145/3133901.

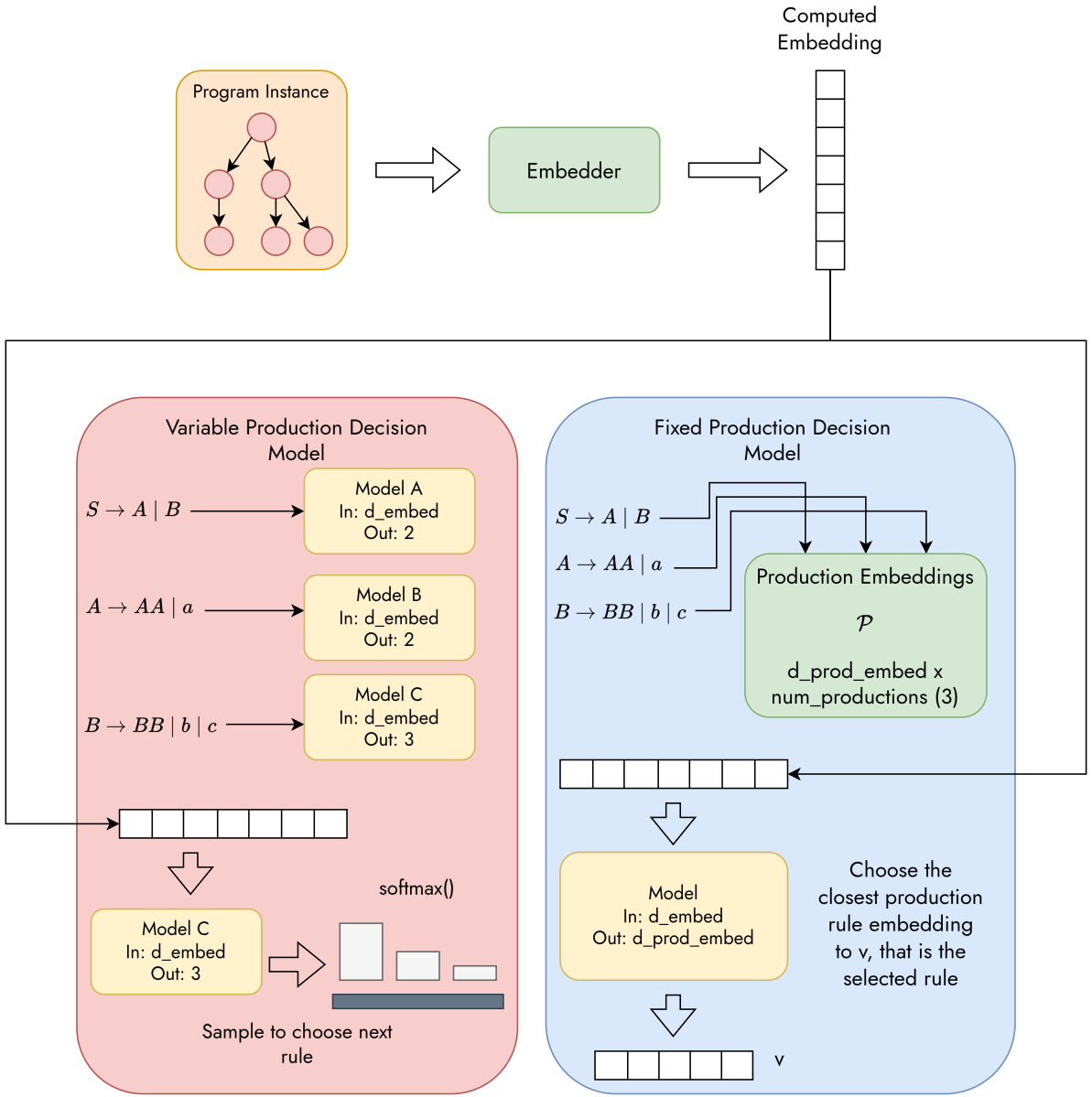

Lamport, Leslie (May 1994). *The Temporal Logic of Actions*. Tech. rep. 79. ACM Transactions on Programming Languages and Systems 16. Microsoft. URL: https://www.microsoft.com/en-us/research/publication/the-temporal-logic-of-actions/.


Le, Quoc and Tomas Mikolov (22–24 Jun 2014). "Distributed Representations of Sentences and Documents". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, pp. 1188–1196. URL: https://proceedings.mlr.press/v32/le14.html.

- Lee, Sean et al. (2024). *Open Source Strikes Bread - New Fluffy Embeddings Model*. Accessed October 10th, 2025. URL: https://www.mixedbread.ai/blog/mxbai-embed-large-v1.
- Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86, pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.
- Merrick, Luke et al. (2024). *Arctic-Embed: Scalable, Efficient, and Accurate Text Embedding Models*. arXiv: 2405.05374 [cs.CL]. URL: https://arxiv.org/abs/2405.05374.
- Narayanan, Annamalai et al. (July 2017). "graph2vec: Learning Distributed Representations of Graphs". In: ACM. DOI: 10.1145/1235.

- Nussbaum, Zach et al. (2024). *Nomic Embed: Training a Reproducible Long Context Text Embedder*. arXiv: 2402.01613 [cs.CL].
- Simard, Nathaniel et al. (Aug. 2024). *Burn*. Version 0.14.0. URL: https://github.com/tracel-ai/burn.
- Simmons-Edler, Riley, Anders Miltner, and Sebastian Seung (June 2018). "Program Synthesis Through Reinforcement Learning Guided Tree Search". In: abs/1806.02932. URL: https://api.semanticscholar.org/CorpusID:47013320.
- White, Colin et al. (Jan. 2023). *Neural Architecture Search: Insights from 1000 Papers*. DOI: 10.48550/arXiv.2301.08727.

Appendices I



Appendices II

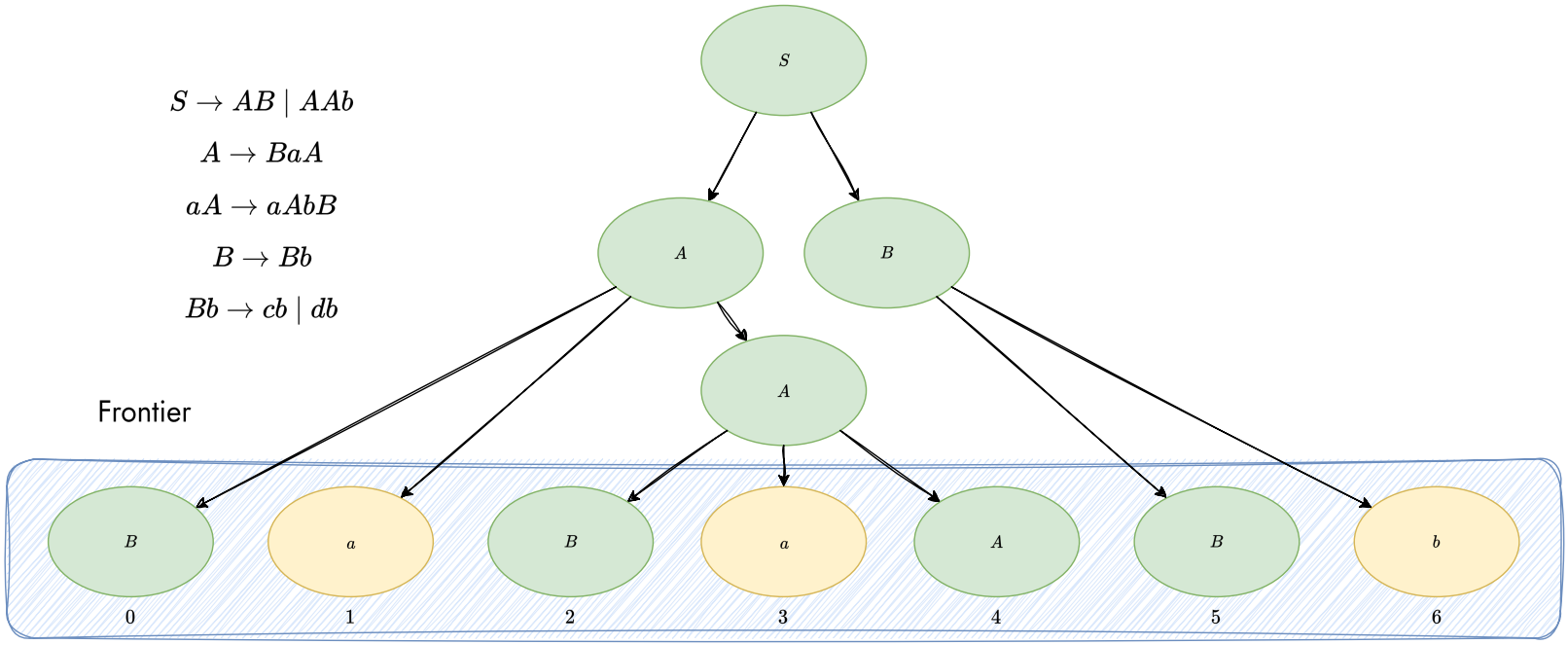
$S \to AB \mid AAb$

$A \to BaA$

$aA \to aAbB$

$B \to Bb$

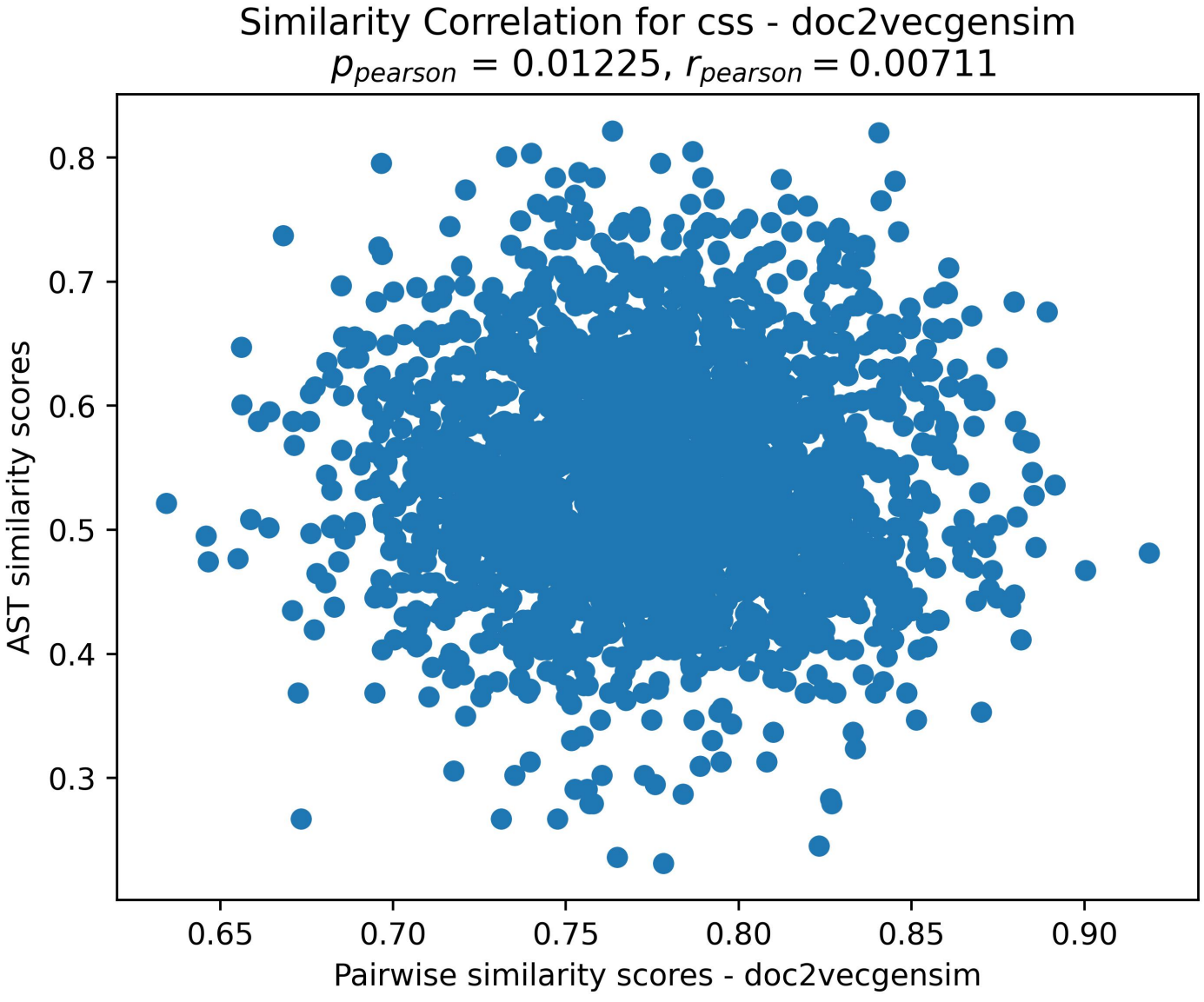
$Bb \to cb \mid db$

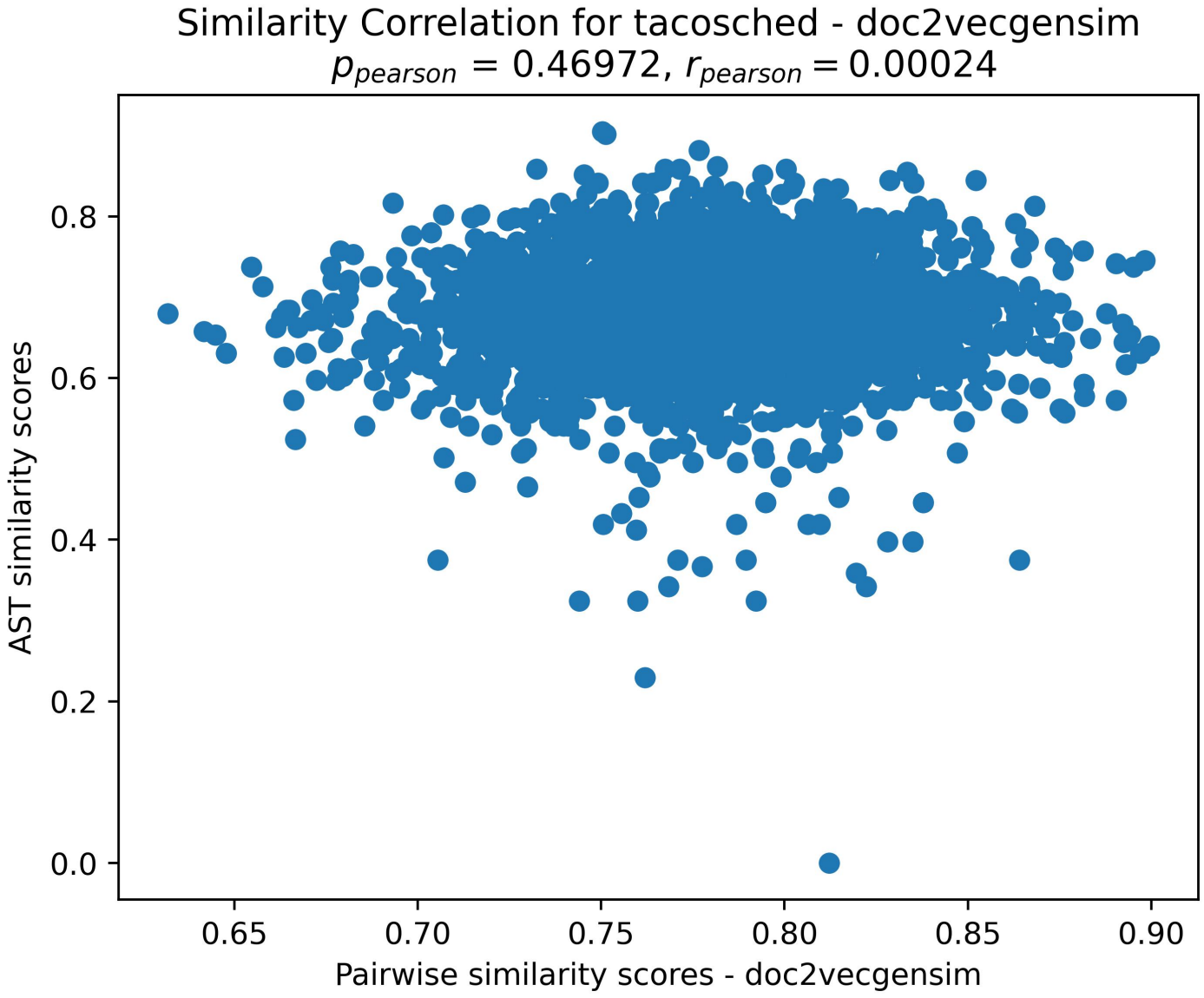


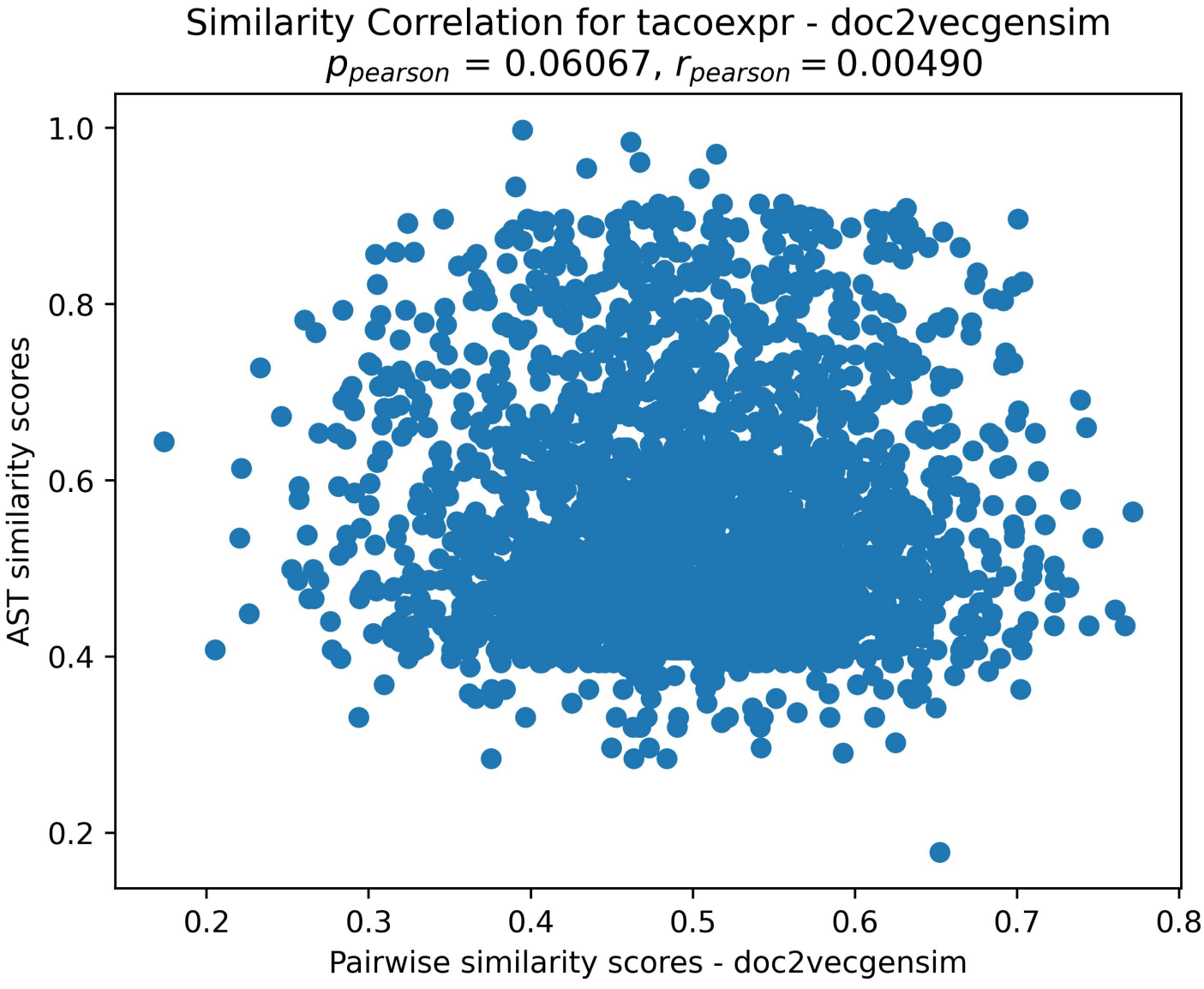
Frontier Decision Table

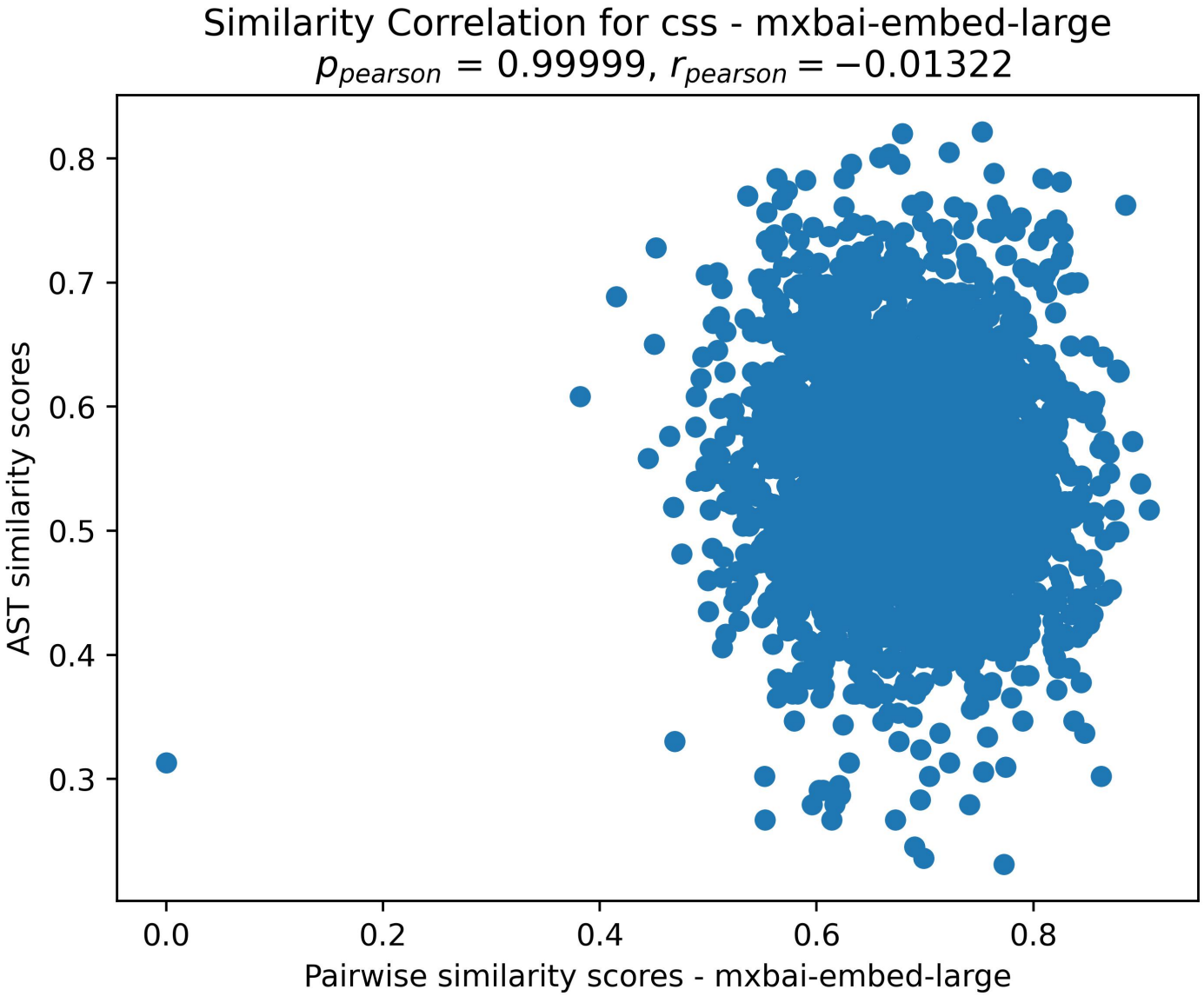
	0	1	2	3	4	5	6
$S \to AB \mid AAb$	0	0	0	0	0	0	0
$A \to BaA$	0	0	0	0	1	0	0
$aA \to aAbB$	0	0	0	0	1	0	0
$B \to Bb$	1	0	1	0	0	1	0
$Bb \to cb \mid db$	0	0	0	0	0	1	0

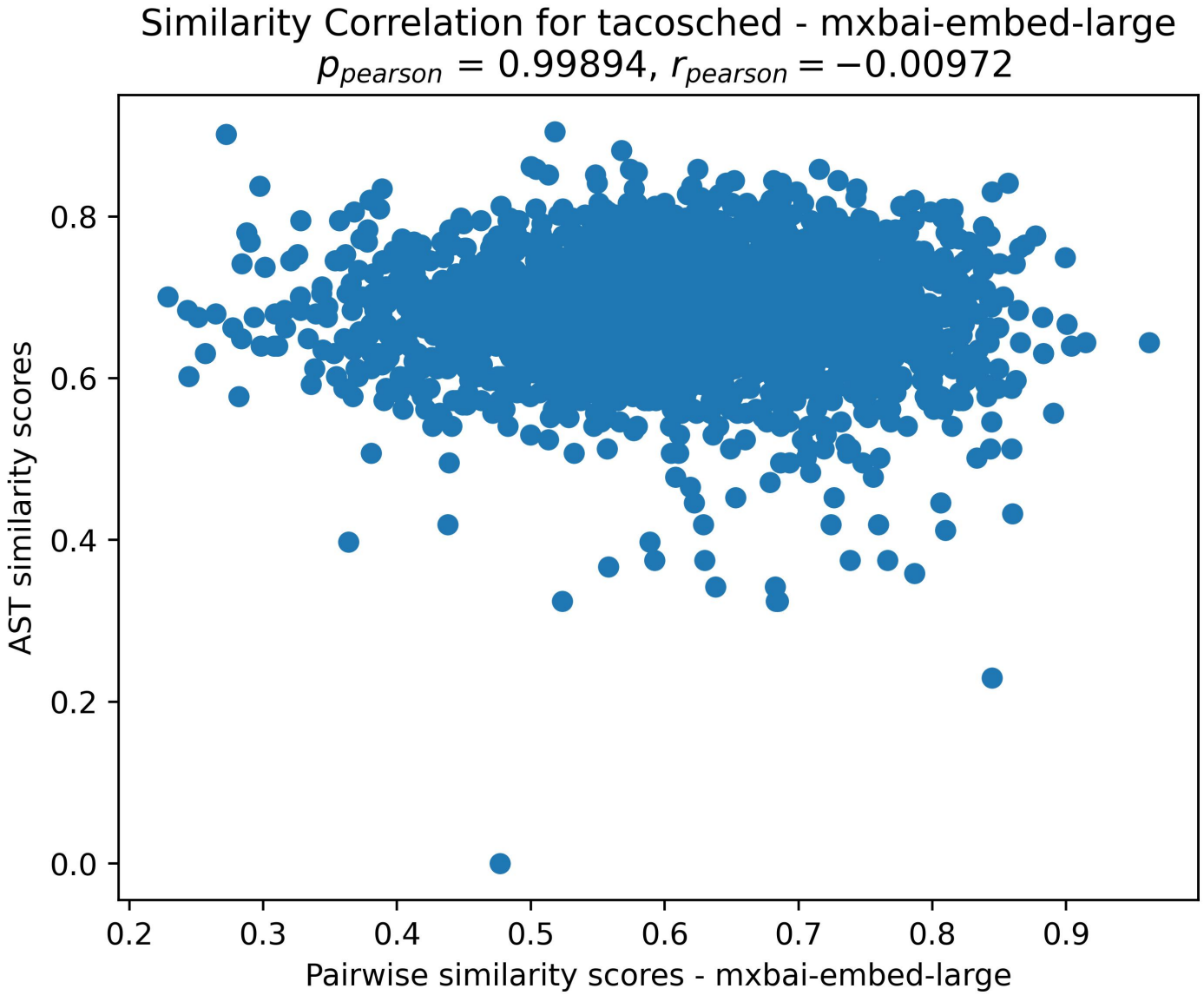
Appendices III

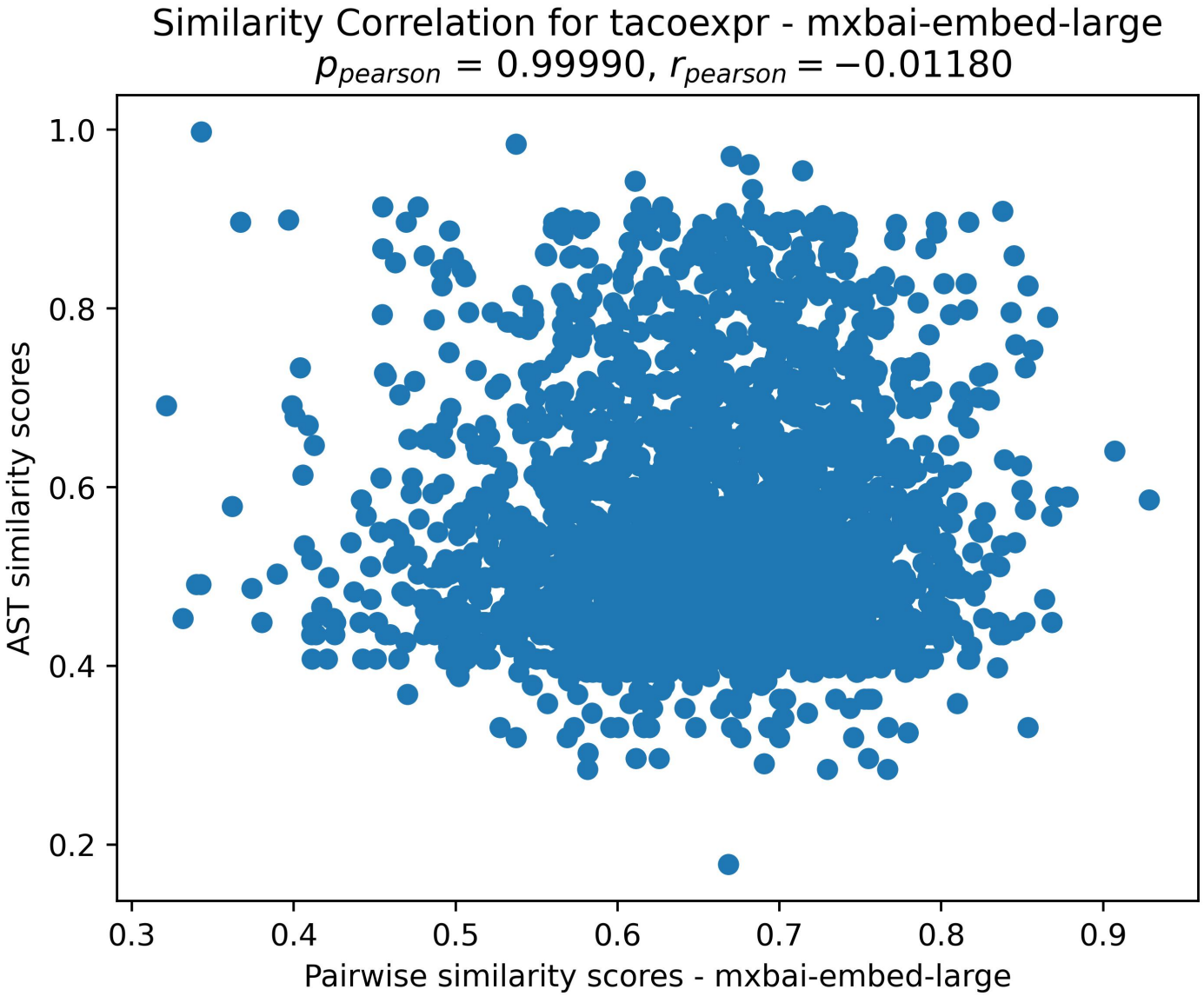


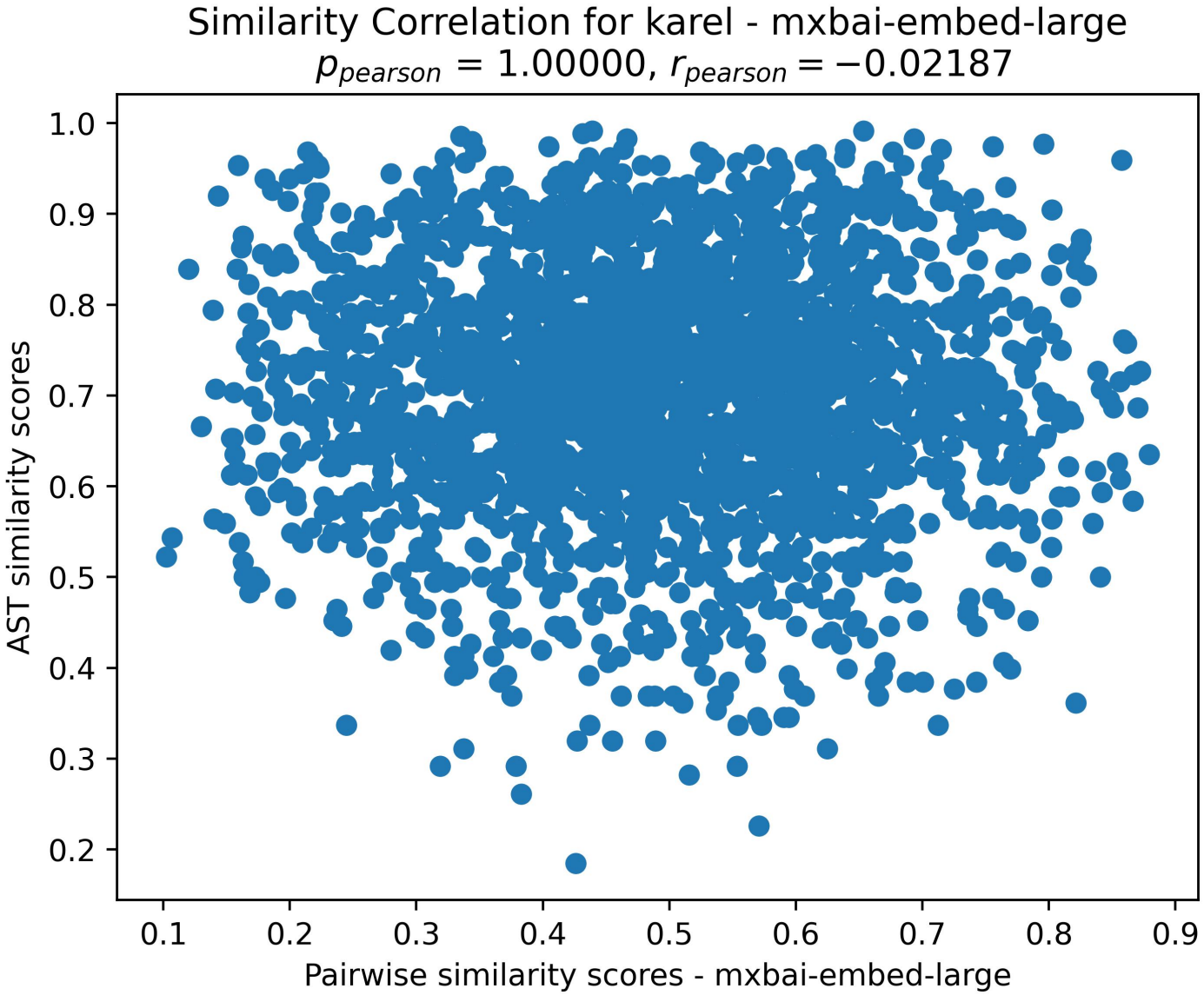

Similarity Correlation for tacosched - doc2vecgensim
$p_{pearson}$ = 0.46972, $r_{pearson}$ = 0.00024
AST similarity scores
0.0
0.2
0.4
0.6
0.8
0.65
0.70
0.75
0.80
0.85
0.90
Pairwise similarity scores - doc2vecgensim

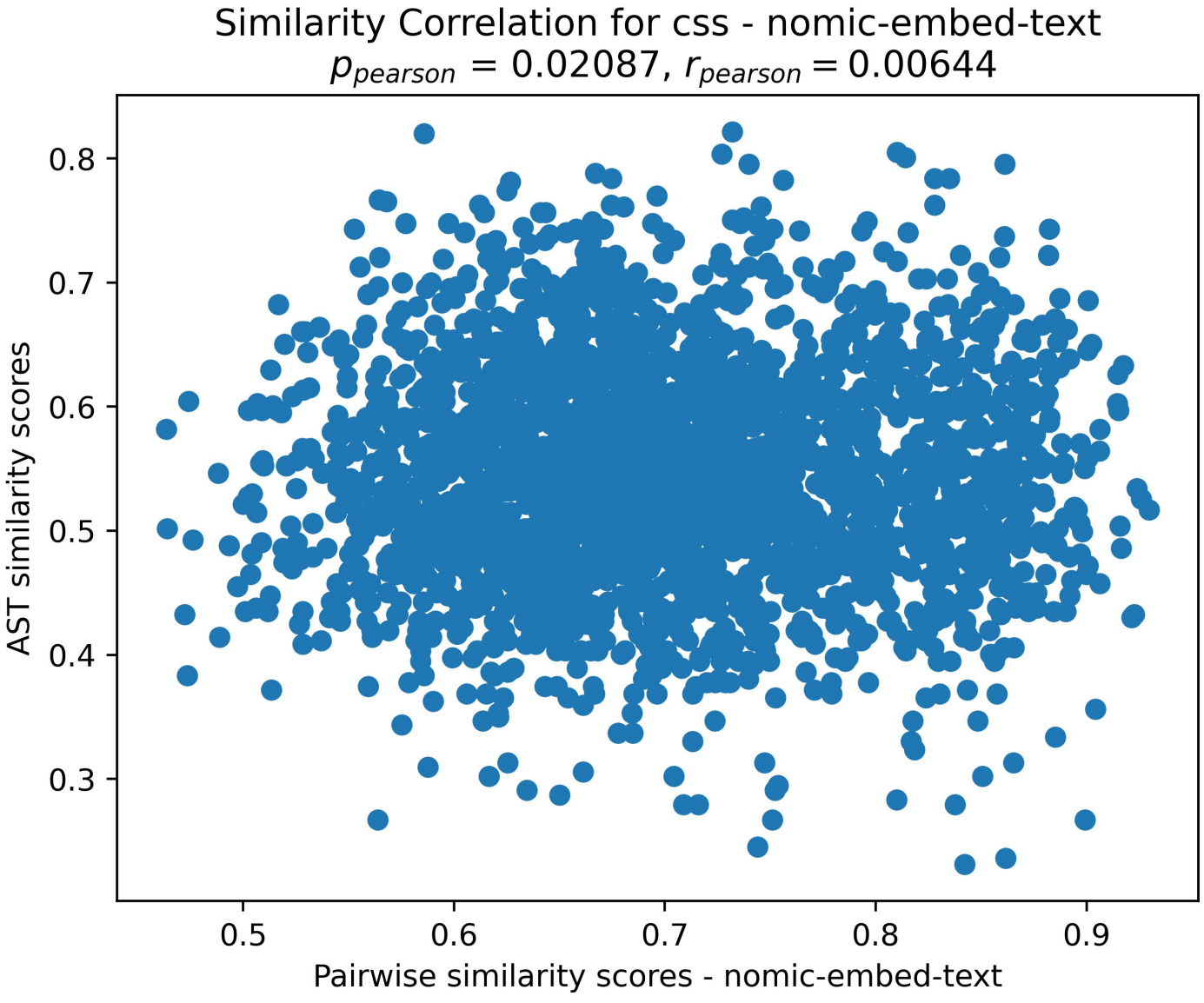

Similarity Correlation for tacoexpr - doc2vecgensim
$p_{pearson} = 0.06067$, $r_{pearson} = 0.00490$
AST similarity scores
1.0
0.8
0.6
0.4
0.2
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Pairwise similarity scores - doc2vecgensim

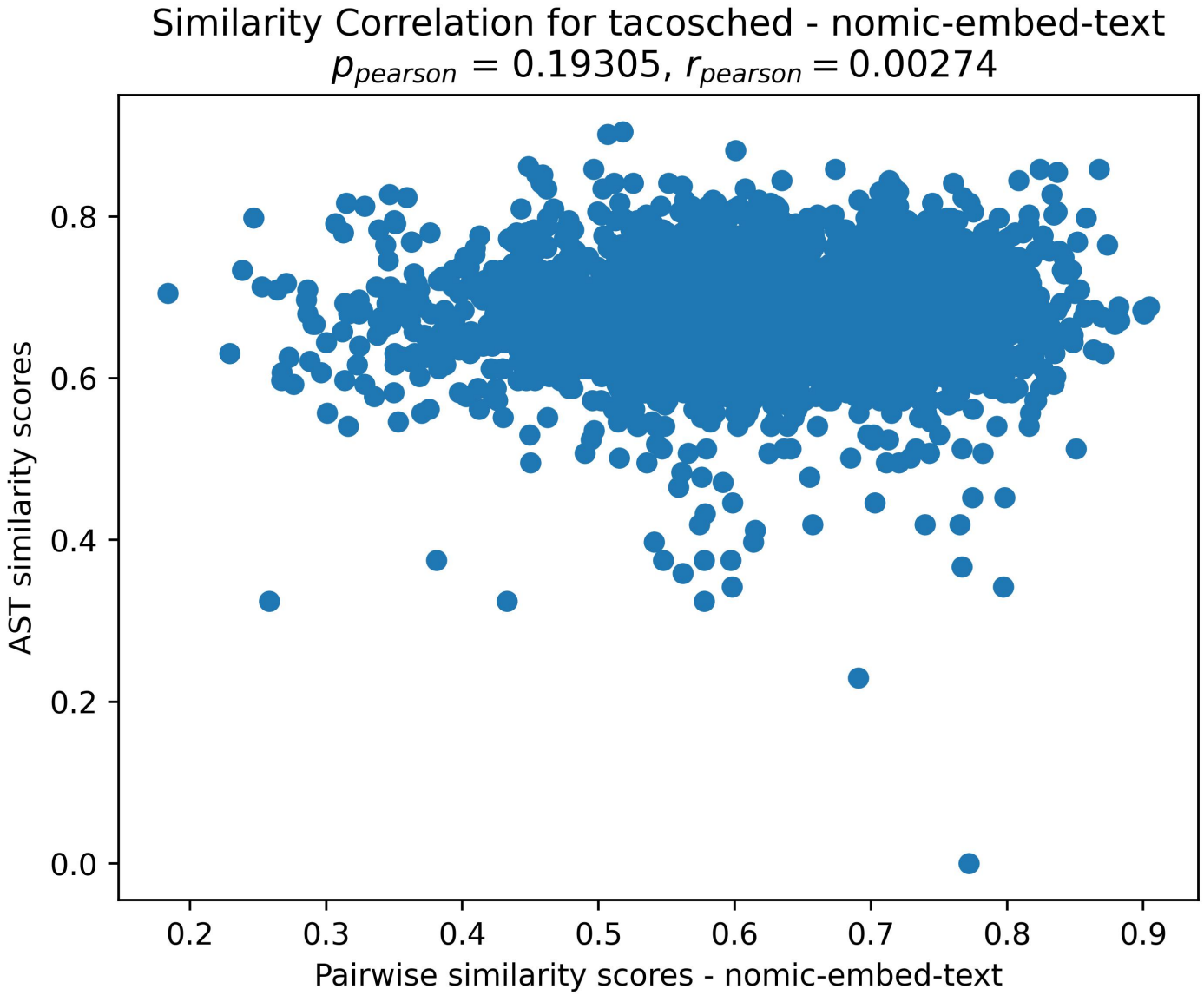

Similarity Correlation for css - mxbai-embed-large
$p_{pearson} = 0.99999$, $r_{pearson} = -0.01322$
AST similarity scores
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.2
0.4
0.6
0.8
Pairwise similarity scores - mxbai-embed-large

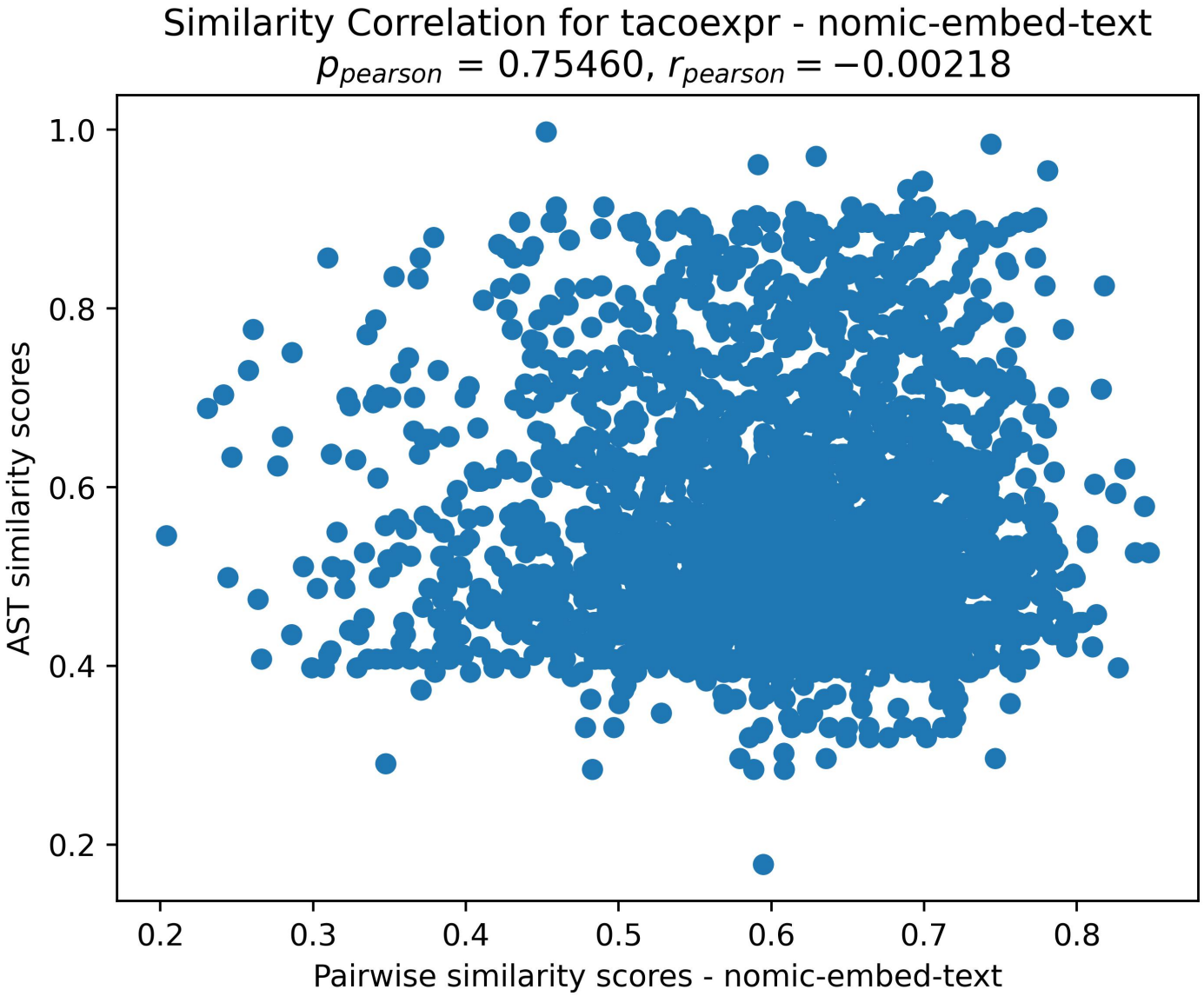

Similarity Correlation for tacosched - mxbai-embed-large
$p_{pearson} = 0.99894$, $r_{pearson} = -0.00972$
AST similarity scores
0.0
0.2
0.4
0.6
0.8
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Pairwise similarity scores - mxbai-embed-large

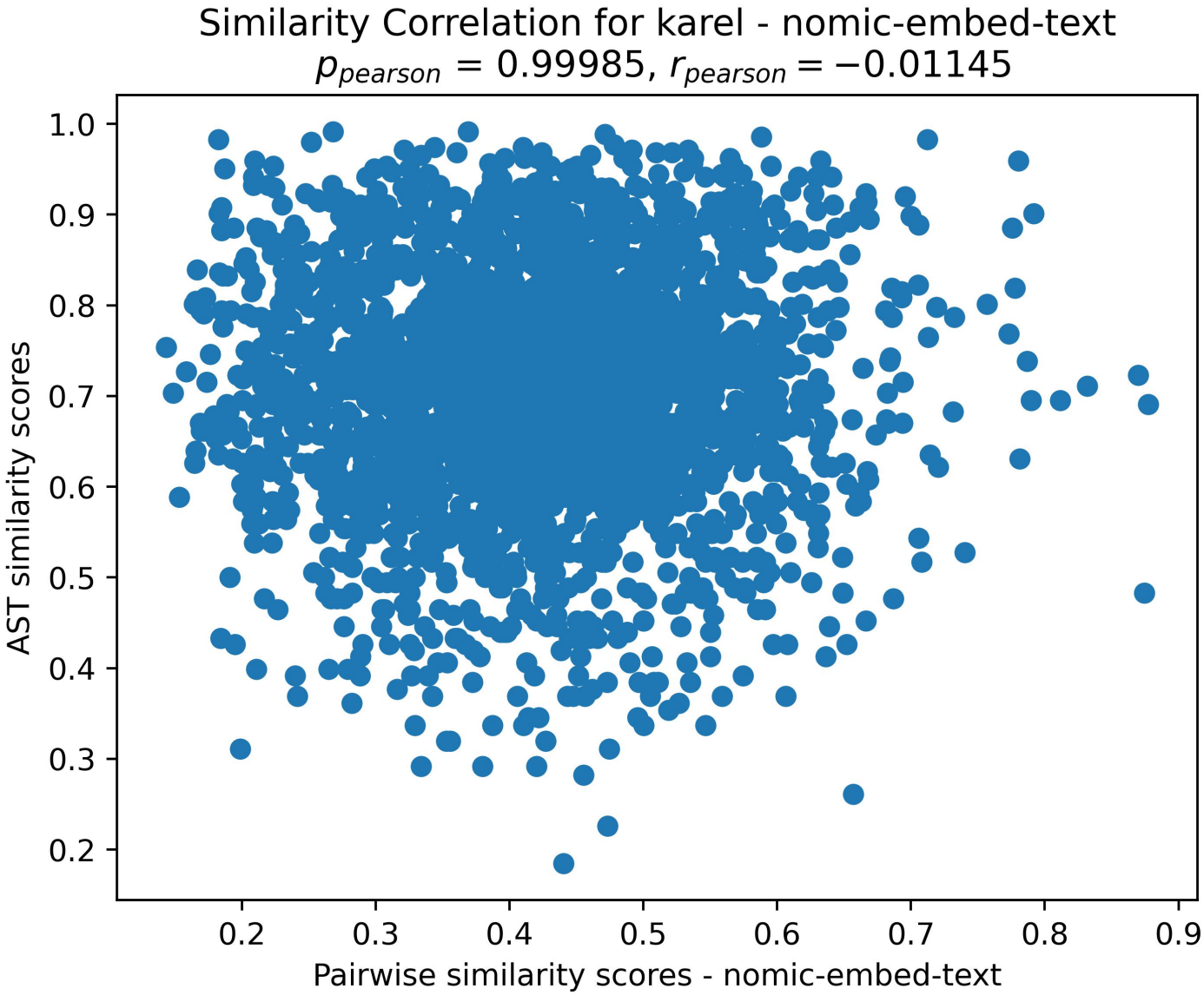

Similarity Correlation for tacoexpr - mxbai-embed-large
$p_{pearson} = 0.99990$, $r_{pearson} = -0.01180$
AST similarity scores
1.0
0.8
0.6
0.4
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Pairwise similarity scores - mxbai-embed-large


Similarity Correlation for karel - mxbai-embed-large
$p_{pearson} = 1.00000$, $r_{pearson} = -0.02187$
AST similarity scores
Pairwise similarity scores - mxbai-embed-large
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

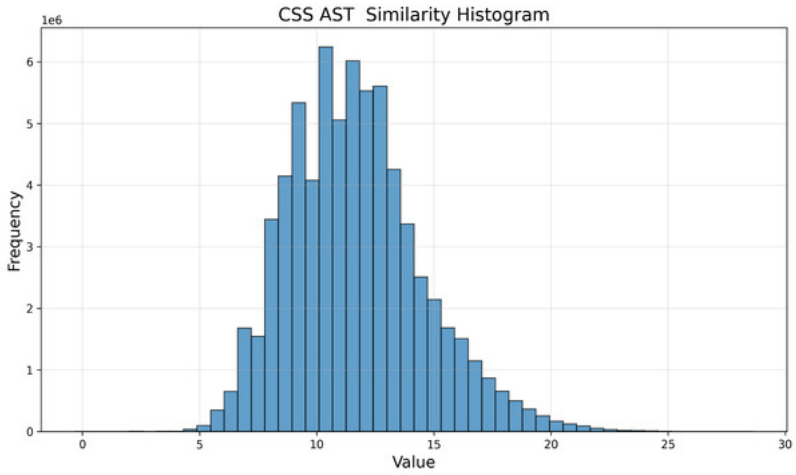

Similarity Correlation for css - nomic-embed-text
$p_{pearson} = 0.02087$, $r_{pearson} = 0.00644$
AST similarity scores
0.8
0.7
0.6
0.5
0.4
0.3
0.5
0.6
0.7
0.8
0.9
Pairwise similarity scores - nomic-embed-text

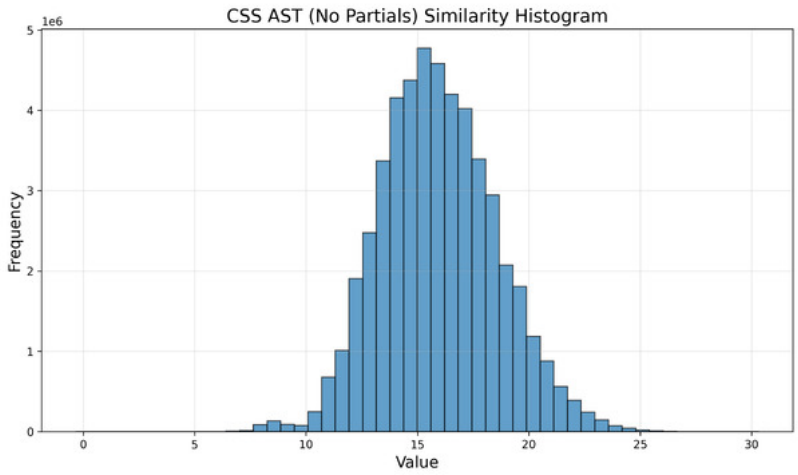

Similarity Correlation for tacosched - nomic-embed-text
$p_{pearson}$ = 0.19305, $r_{pearson}$ = 0.00274
AST similarity scores
0.0
0.2
0.4
0.6
0.8
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Pairwise similarity scores - nomic-embed-text

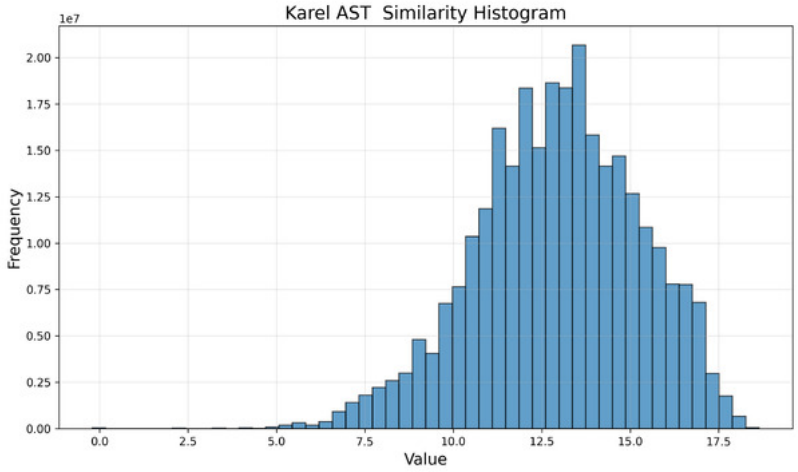

Similarity Correlation for tacoexpr - nomic-embed-text
$p_{pearson} = 0.75460$, $r_{pearson} = -0.00218$
AST similarity scores
1.0
0.8
0.6
0.4
0.2
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Pairwise similarity scores - nomic-embed-text

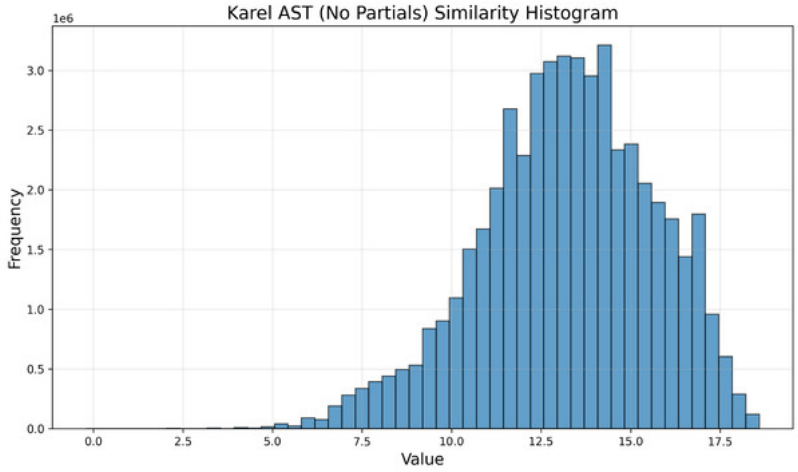

Similarity Correlation for karel - nomic-embed-text
$p_{pearson} = 0.99985$, $r_{pearson} = -0.01145$
AST similarity scores
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Pairwise similarity scores - nomic-embed-text

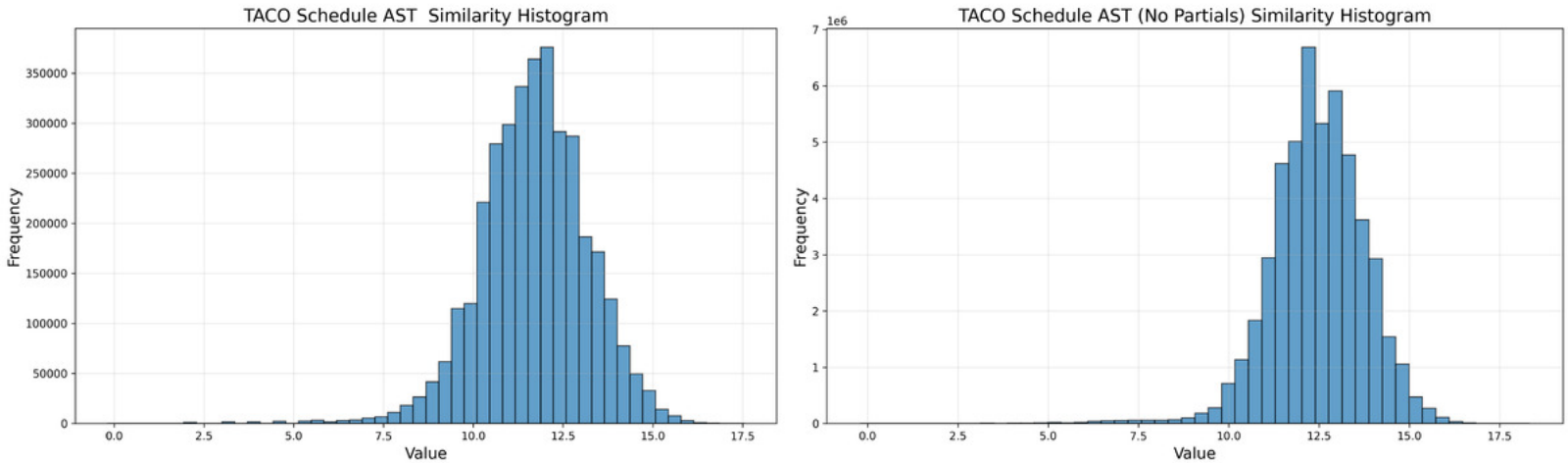
Appendices XIV



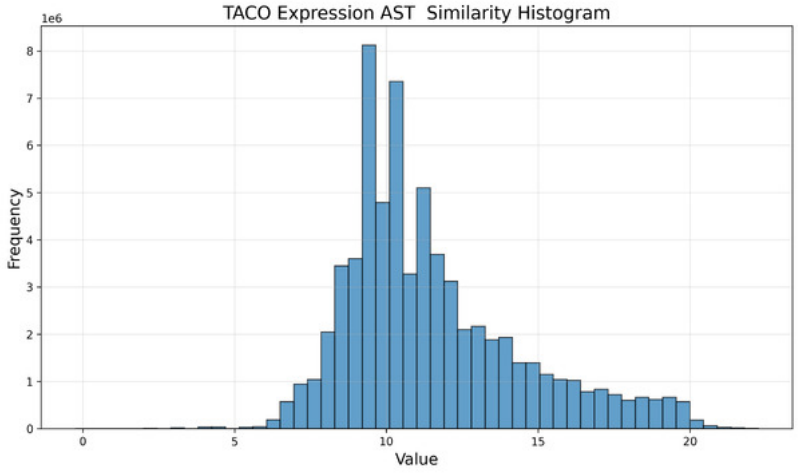


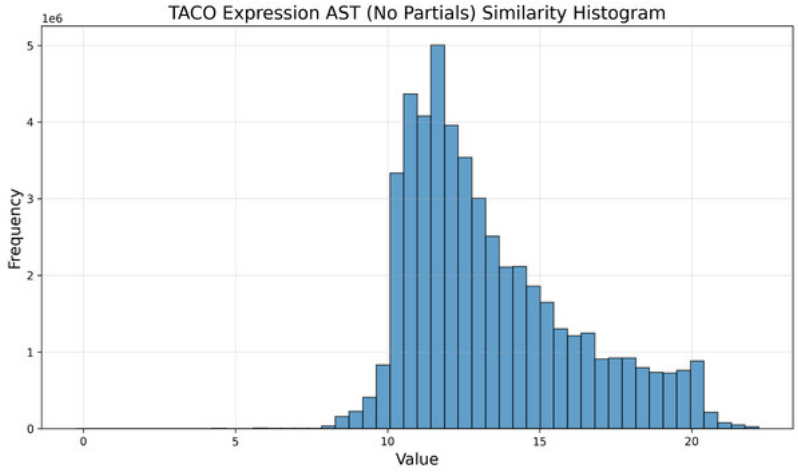

Karel AST Similarity Histogram
1e7
Frequency
Value
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Karel AST (No Partials) Similarity Histogram
1e6
Frequency
Value
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

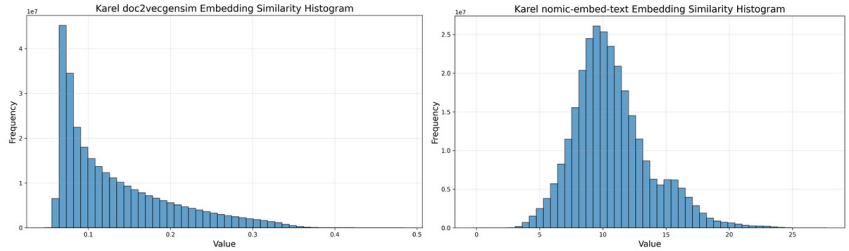

TACO Schedule AST Similarity Histogram
Frequency
Value
0
50000
100000
150000
200000
250000
300000
350000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
TACO Schedule AST (No Partials) Similarity Histogram
1e6
Frequency
Value
0
1
2
3
4
5
6
7
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

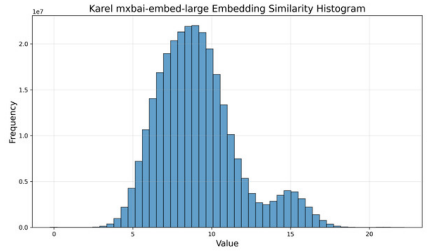
Appendices XVII

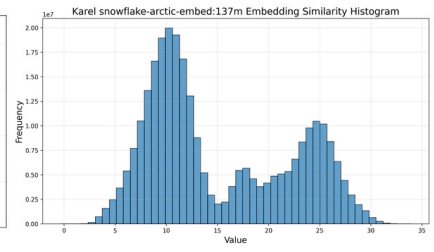


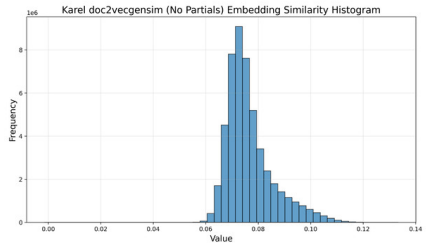


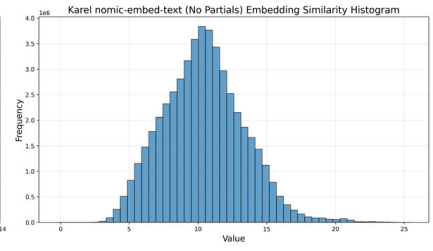
Appendices XVIII

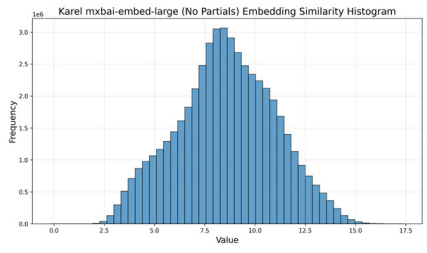


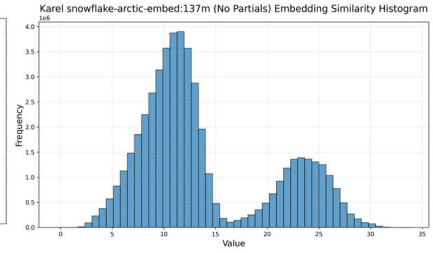


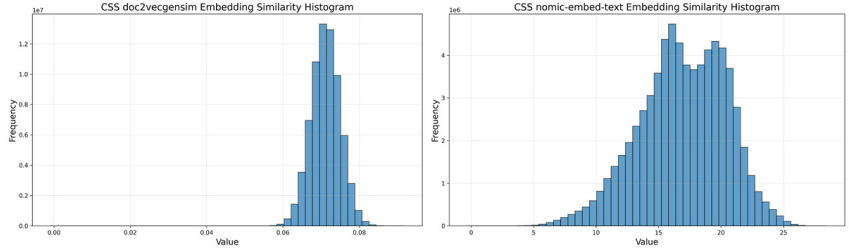




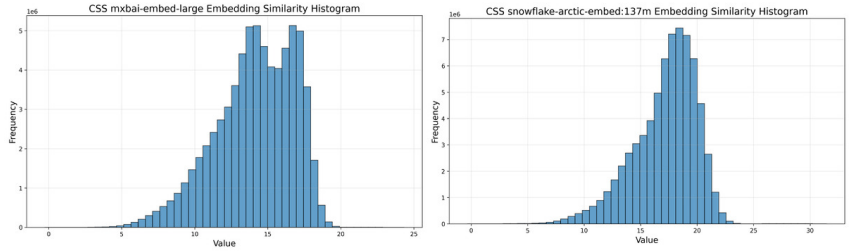




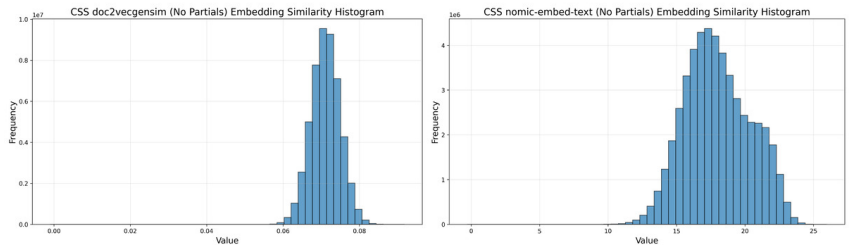



CSS doc2vecgensim Embedding Similarity Histogram
1e7
Frequency
Value

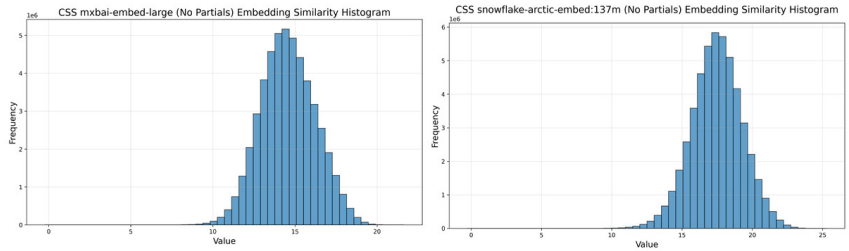
CSS nomic-embed-text Embedding Similarity Histogram
1e6
Frequency
Value


CSS mxbai-embed-large Embedding Similarity Histogram
1e6
Frequency
Value

CSS snowflake-arctic-embed:137m Embedding Similarity Histogram
1e6
Frequency
Value

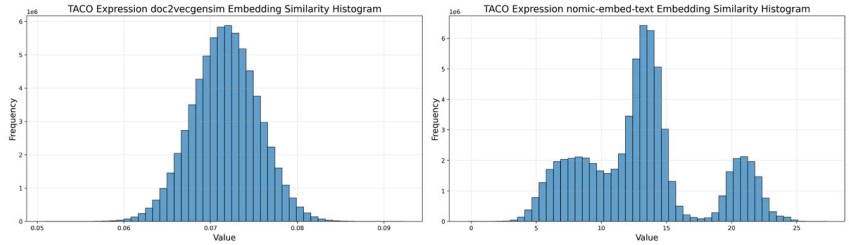

CSS doc2vecgensim (No Partials) Embedding Similarity Histogram
1e7
Frequency
Value

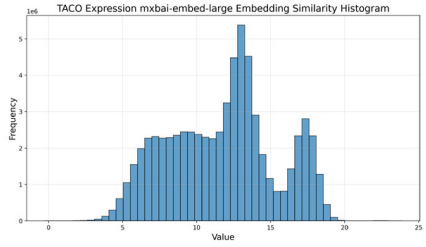
CSS nomic-embed-text (No Partials) Embedding Similarity Histogram
1e6
Frequency
Value

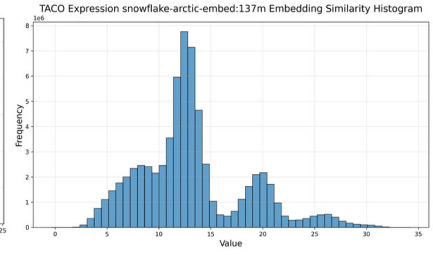

CSS mxbai-embed-large (No Partials) Embedding Similarity Histogram
1e6
Frequency
Value

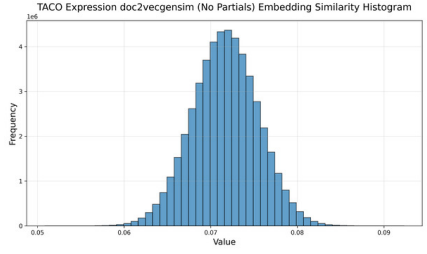
CSS snowflake-arctic-embed:137m (No Partials) Embedding Similarity Histogram
1e6
Frequency
Value

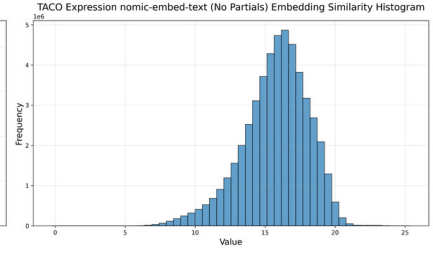
Appendices XX

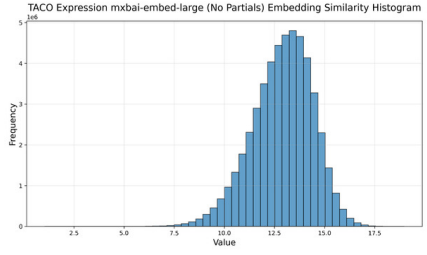


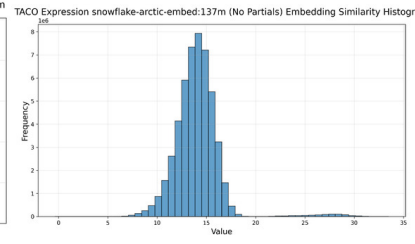


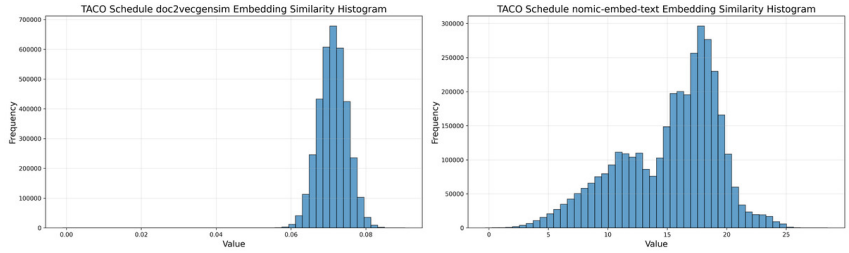




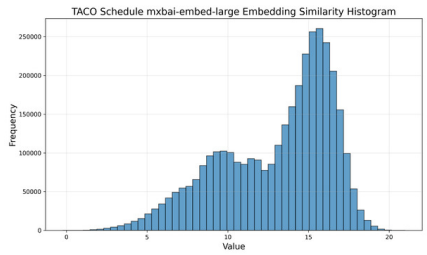


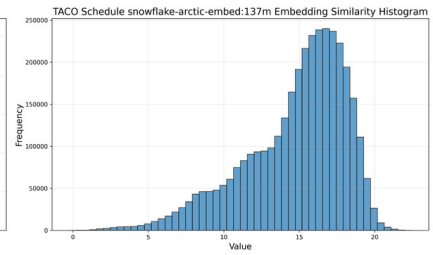


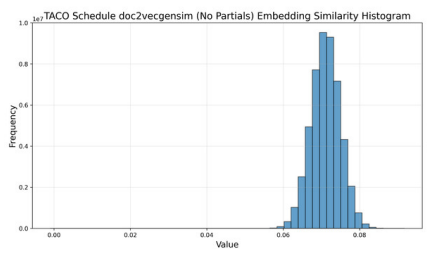


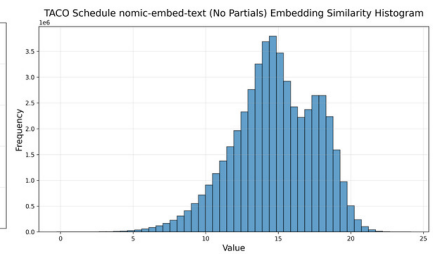

TACO Schedule doc2vecgensim Embedding Similarity Histogram
Frequency
Value

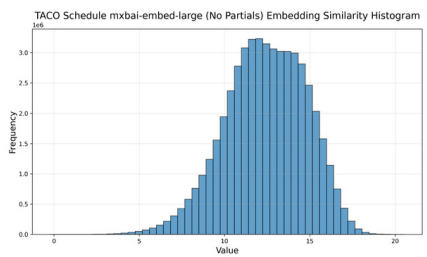
TACO Schedule nomic-embed-text Embedding Similarity Histogram
Frequency
Value

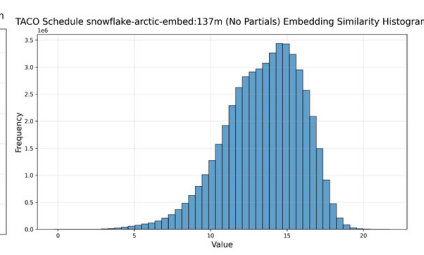

TACO Schedule mxbai-embed-large Embedding Similarity Histogram
Frequency
Value


TACO Schedule snowflake-arctic-embed:137m Embedding Similarity Histogram
Frequency
Value


TACO Schedule doc2vecgensim (No Partials) Embedding Similarity Histogram
Frequency
Value


TACO Schedule nomic-embed-text (No Partials) Embedding Similarity Histogram
Frequency
Value


TACO Schedule mxbai-embed-large (No Partials) Embedding Similarity Histogram
Frequency
Value


TACO Schedule snowflake-arctic-embed:137m (No Partials) Embedding Similarity Histogram
Frequency
Value