On the Impact of Language Selection for Training and Evaluating Programming Language Models

Jonathan Katzy, Maliheh Izadi, Arie van Deursen

TU Delft

Intro

- Why do LLMs perform worse in some languages?

- Does language choice matter?

Background

- Large Language Models for Code tasks
- Multilingual models
- Fine-tuning
- Transfer Learning

Goals

- Map language similarities
- Identify distinct groupings of languages

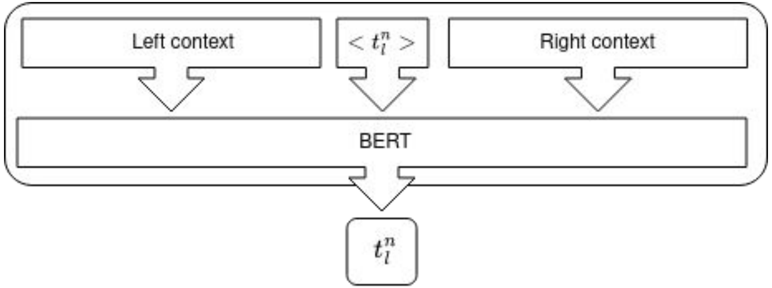
Approach - Overview

- Multilingual exploration of representations
- Comparison of token representations in a language
- Comparison of languages

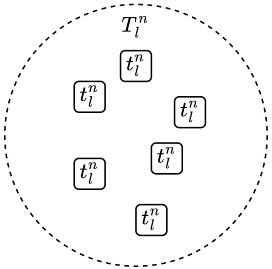
TU Delft

Approach - Representation

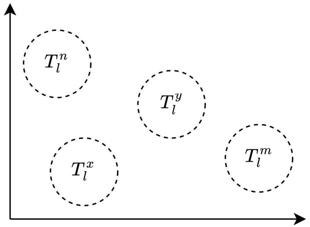
- BERT Representation



- "Token" set of representations



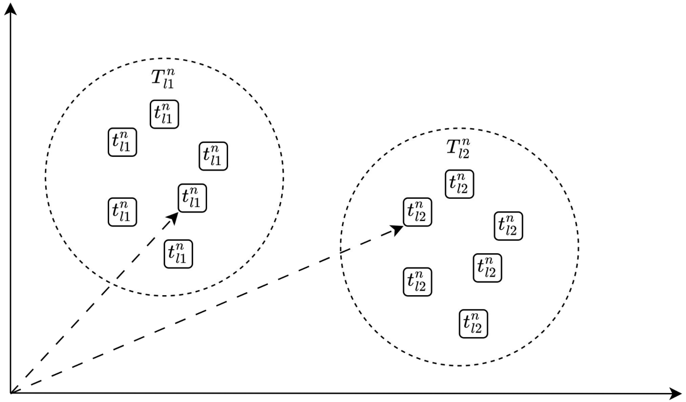
- "Language" Set of Tokens



TUDelft

Approach - Comparison

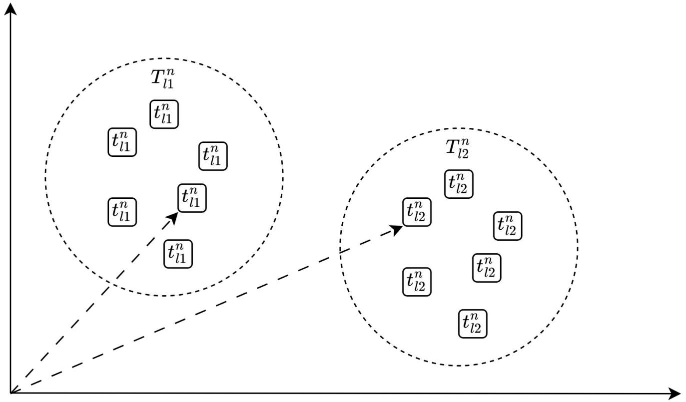
- Similarity between
  - Languages
  - Tokens
  - Representations



TU Delft

Approach - Comparison

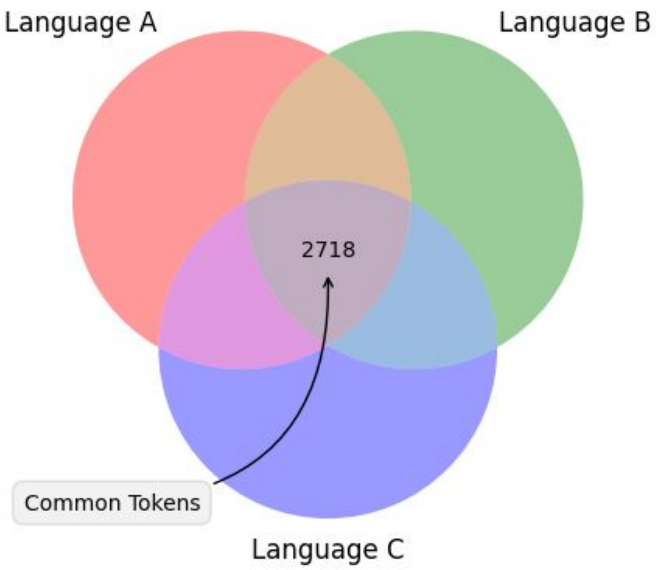
- Representation
  - Max Cosine similarity
- Token
  - Average
- Language
  - Average



Approach - Data

- The stack
  - 20 languages
  - 100k files

- Variety of languages
  - Different grammars
  - Different use-cases

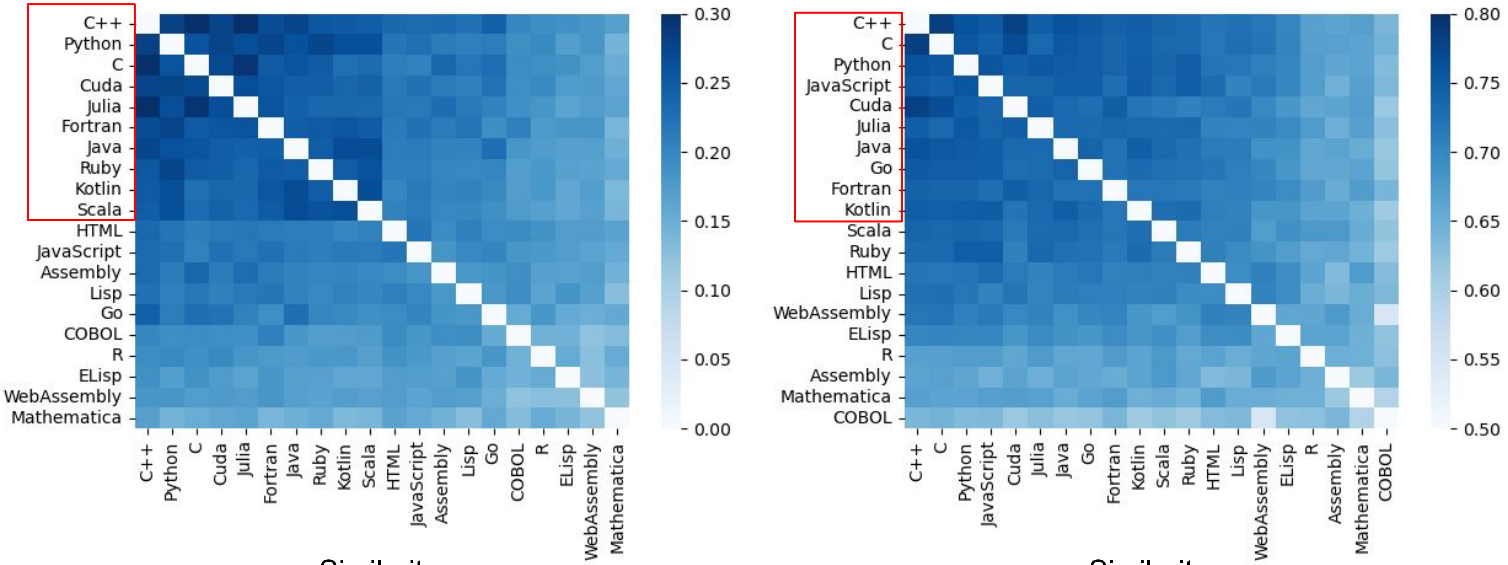


Approach

- Wide variety of languages
  - Different grammars
  - Different use-cases

Results

- Common languages are similar

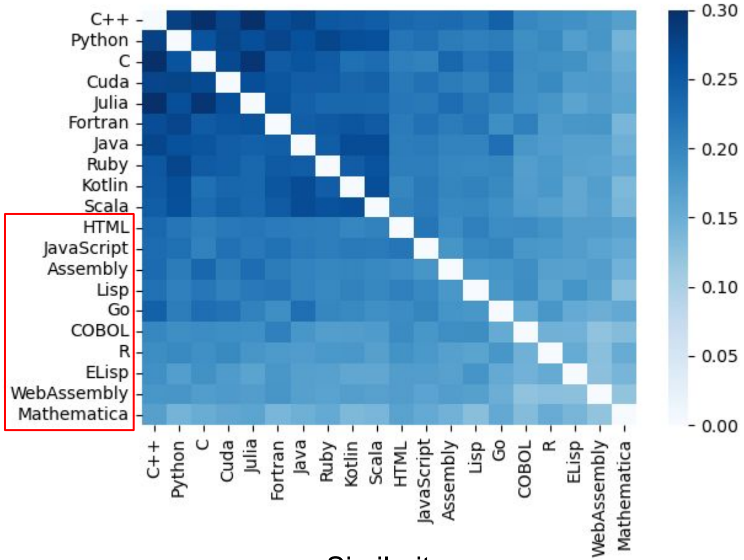


Similarity
No pre-training

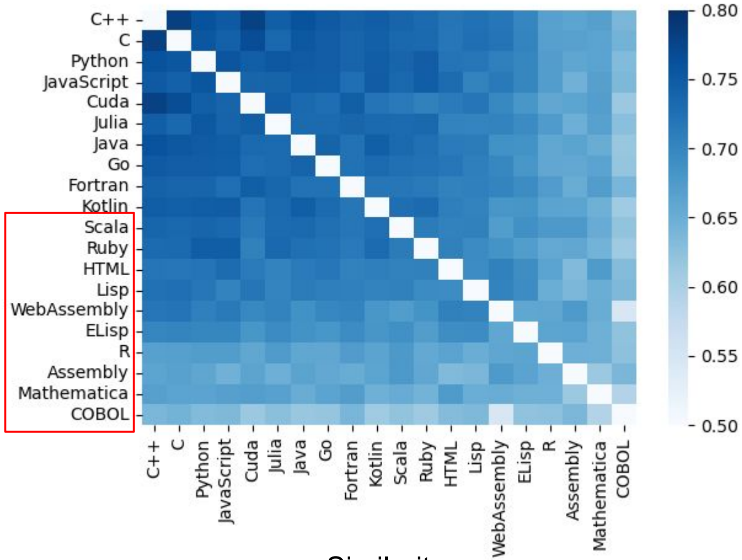
Similarity
pre-trained

Results

- Common languages are similar
- Others are not



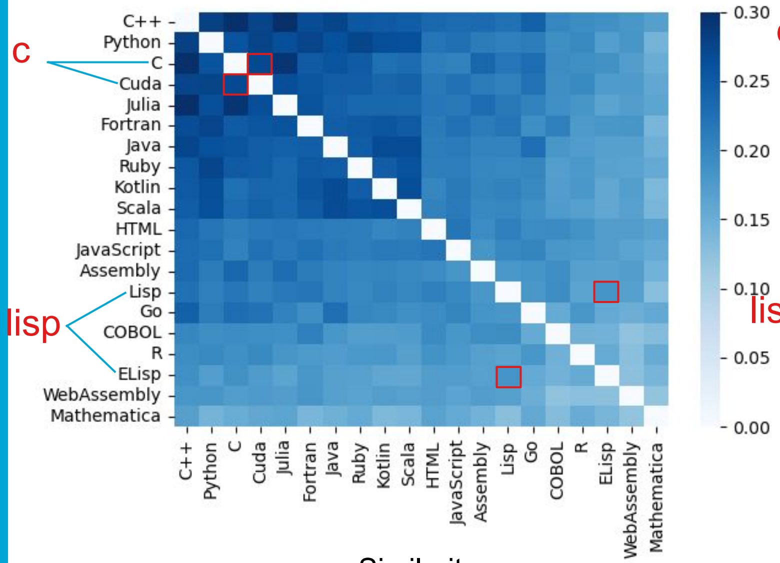
Similarity
No pre-training



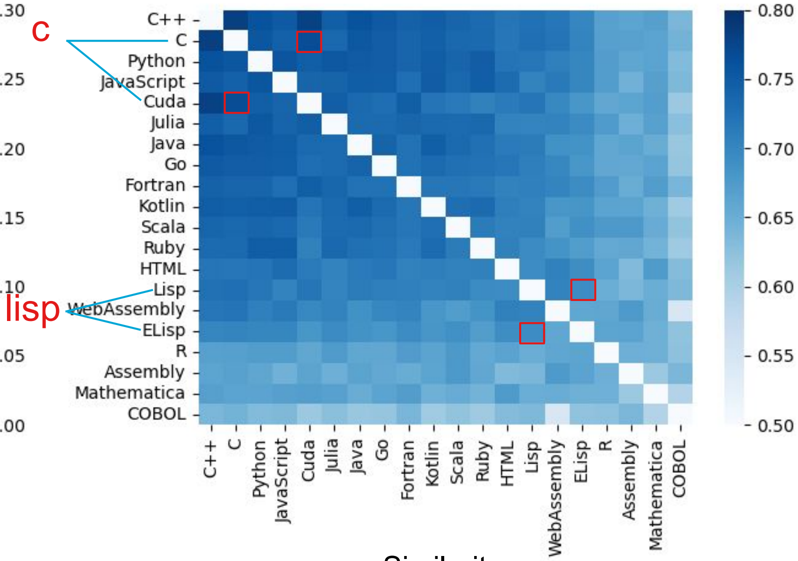
Similarity
pre-trained

Results

- Domain specific languages differ a little



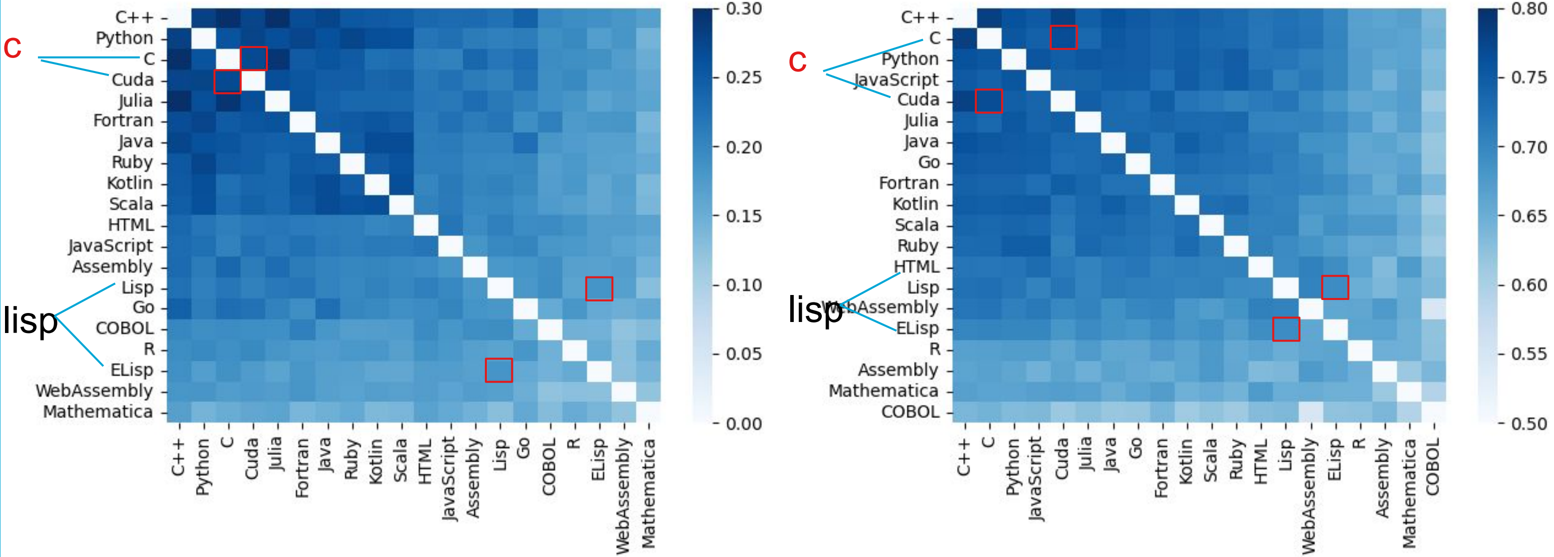
Similarity
No pre-training



Similarity
pre-trained

Results

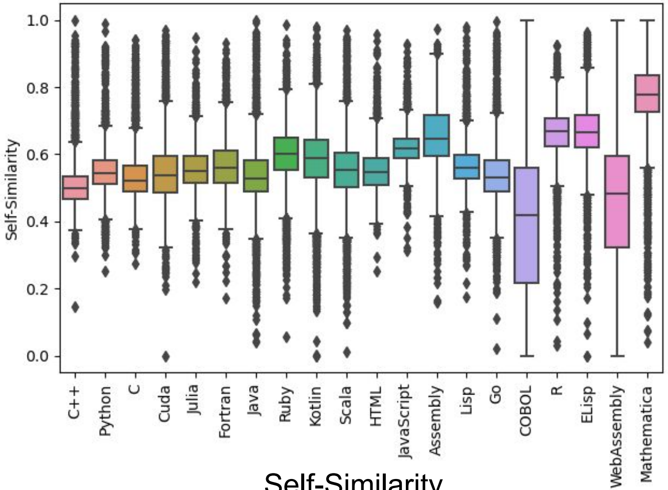
- Domain specific languages differ a little



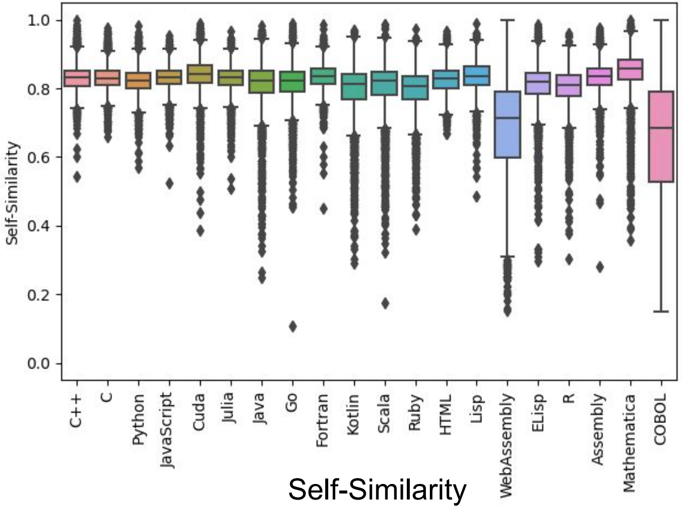
TU Delft

Results

- Pretraining makes representation more consistent



Self-Similarity
No pre-training



Self-Similarity
pre-trained

Why

- Difference in language performance
- Implications for applications
  - Transfer learning
  - Fine-tuning
  - Low resource languages

Conclusion

- There are consistent differences
- Use-case more important than grammar
- Implications
  - Transfer learning
  - Fine-tuning
  - Low resource languages

TUDelft

Future work

- More architectures
- Correlation to performance
- Analyze downstream tasks

TU Delft

Questions?

JKatzy.nl

J.B.Katzy@TUDelft.nl

@katzy_jonathan

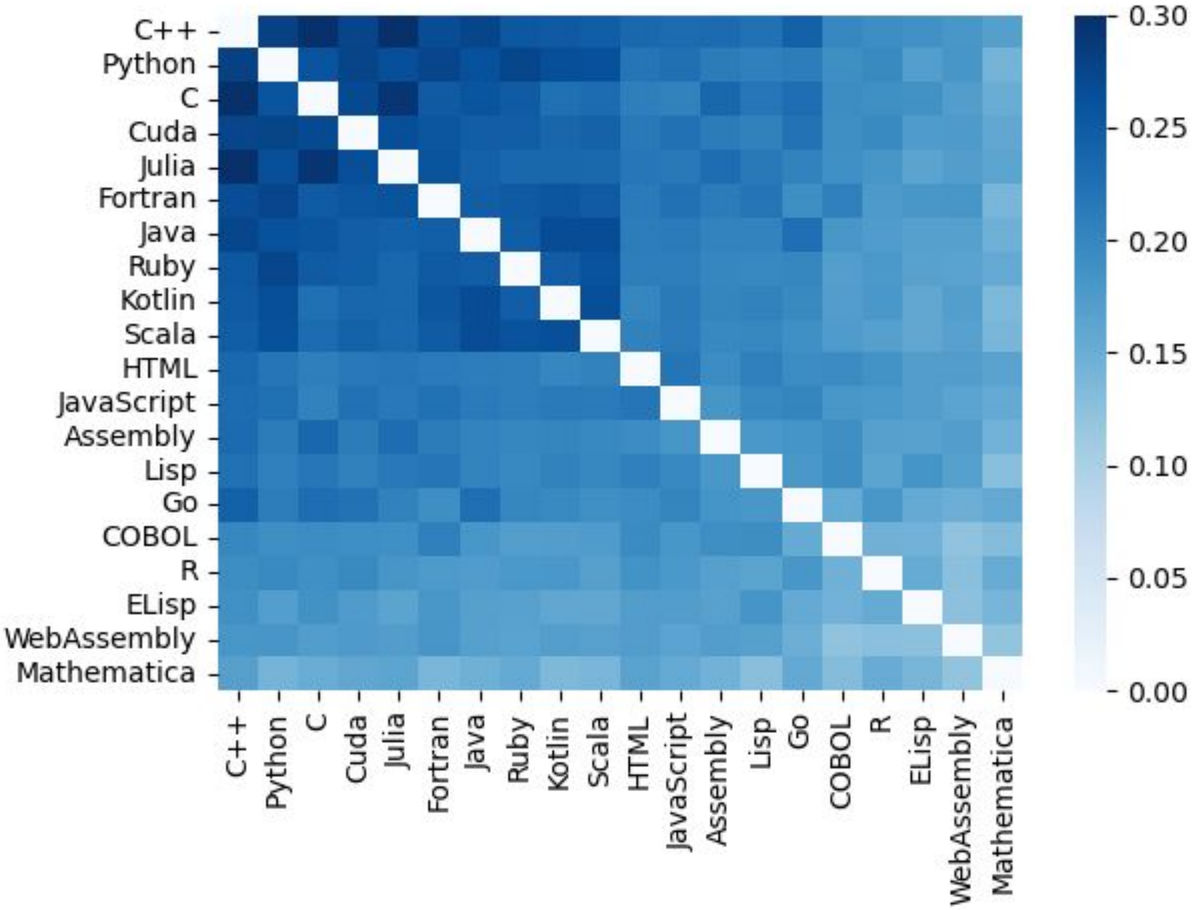
jkatzy

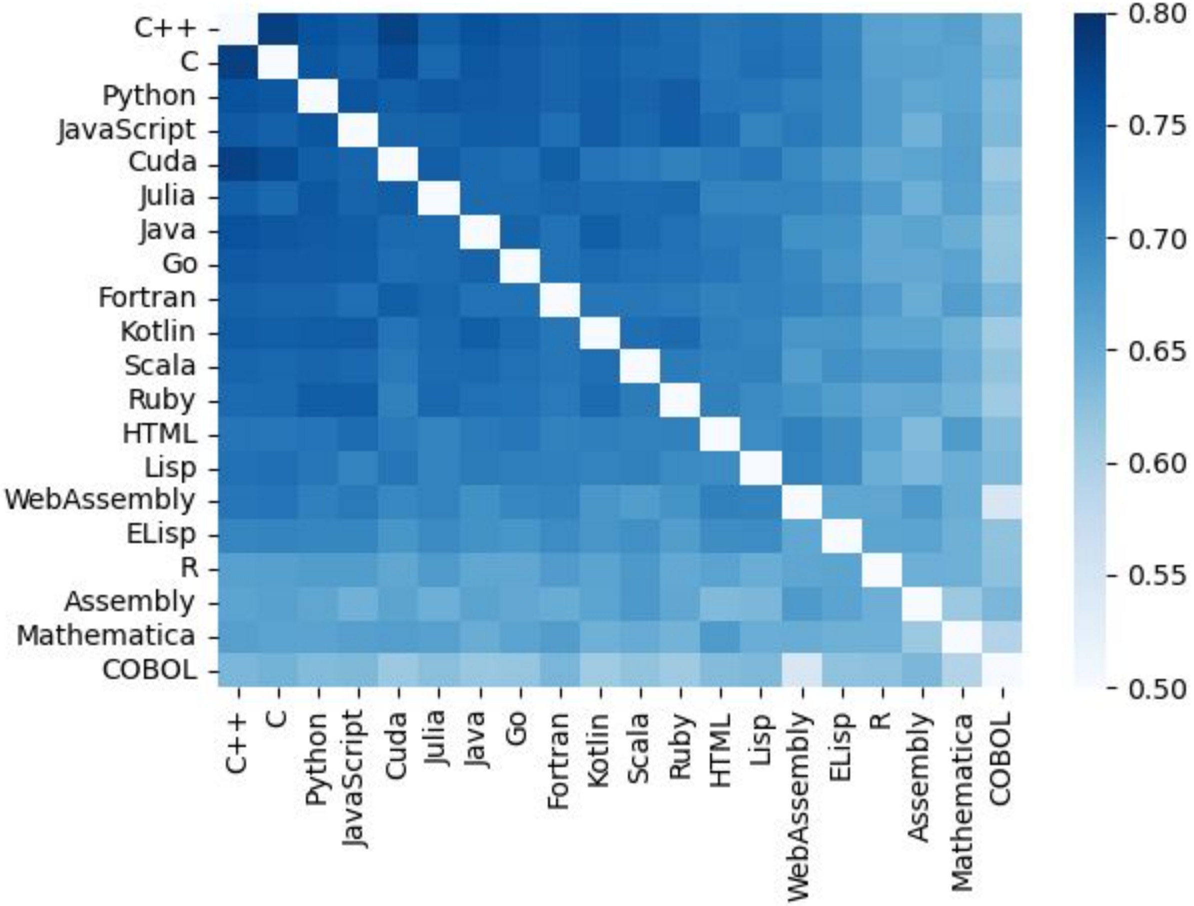
Language representation

Representation tasks

TU Delft

Language	Inclusion criteria	Files	Total Tokens
Assembly	Unique syntax with a limited vocabulary	100,000	364,776,405
C	Widely used general-purpose programming language	100,000	326,871,237
COBOL	The language often present in legacy systems, with a very unique syntax	2,978	10,613,233
C++	Widely used general-purpose programming language, close to Java and C	100,000	368,090,173
Cuda	Domain specific application of C++	58,355	283,624,967
Emacs Lisp	Domain-specific application of Lisp	54,768	188,661,262
Fortran	Scientific computing language, with similar syntax to Julia and Ruby	100,000	607,478,891
Go	Domain-specific language with elements from C, C++, Python, and Ruby	100,000	232,054,204
HTML	Domain-specific language, with unique syntax	100,000	723,969,345
Java	Widely used general-purpose programming language	100,000	183,040,204
JavaScript	Widely used domain-specific programming language	100,000	325,109,387
Julia	New emerging scientific computing language	100,000	242,836,338
Kotlin	Mixture of Java and JS elements but less verbose	100,000	111,578,961
Lisp	General purpose list-based programming language	100,000	832,184,093
Mathematica	Mathematical computing language with unique features	26,895	1,035,010,885
Python	General purpose programming language, with semantic whitespace	100,000	237,414,388
R	Scientific computing language	39,194	154,180,798
Ruby	General purpose language with syntax similar to Python and Julia	100,000	93,200,451
Scala	JVM-based language with syntactic elements from JavaScript and C++	100,000	141,672,916
WebAssembly	Domain-specific emerging list-based language	5,359	59,809,452

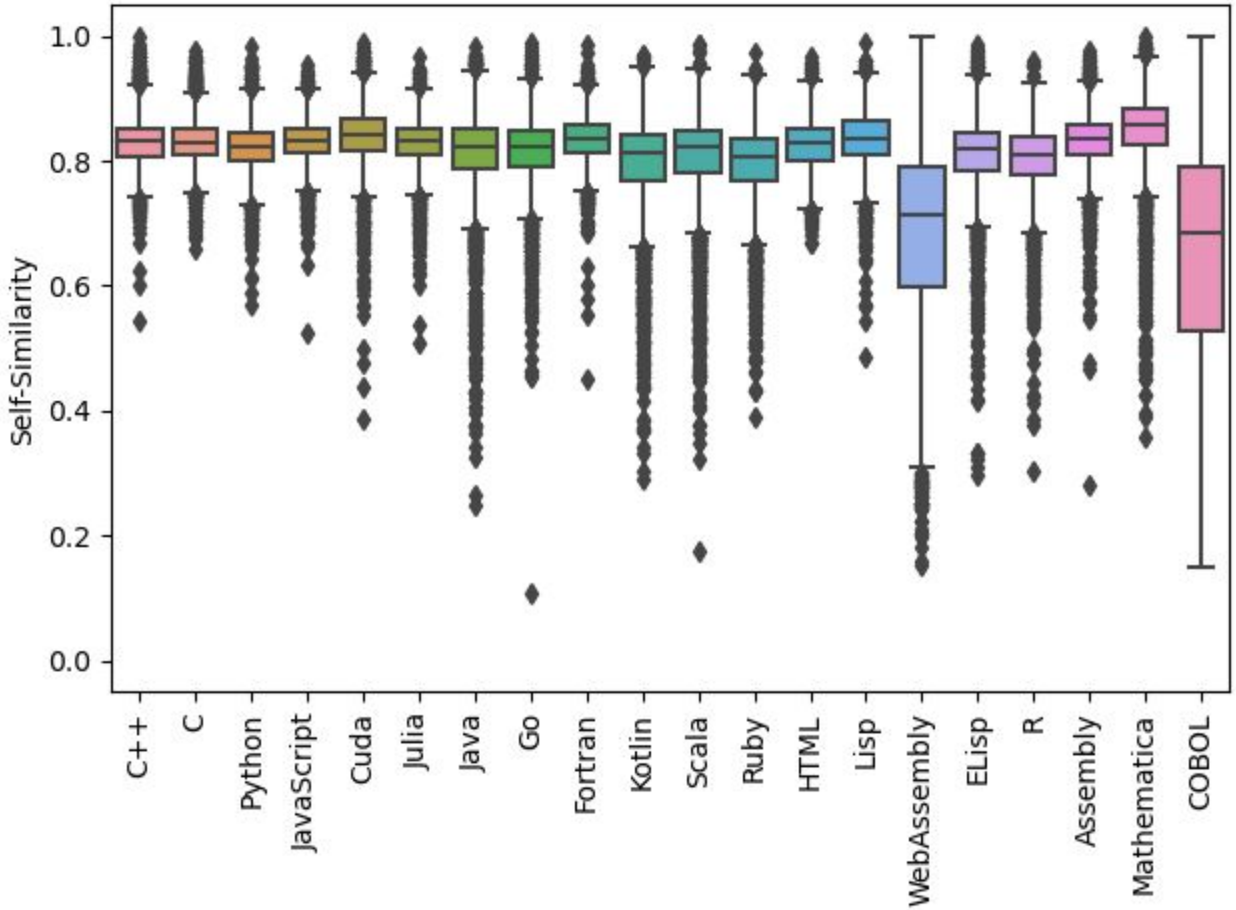

C++
Python
C
Cuda
Julia
Fortran
Java
Ruby
Kotlin
Scala
HTML
JavaScript
Assembly
Lisp
Go
COBOL
R
ELisp
WebAssembly
Mathematica
0.30
0.25
0.20
0.15
0.10
0.05
0.00



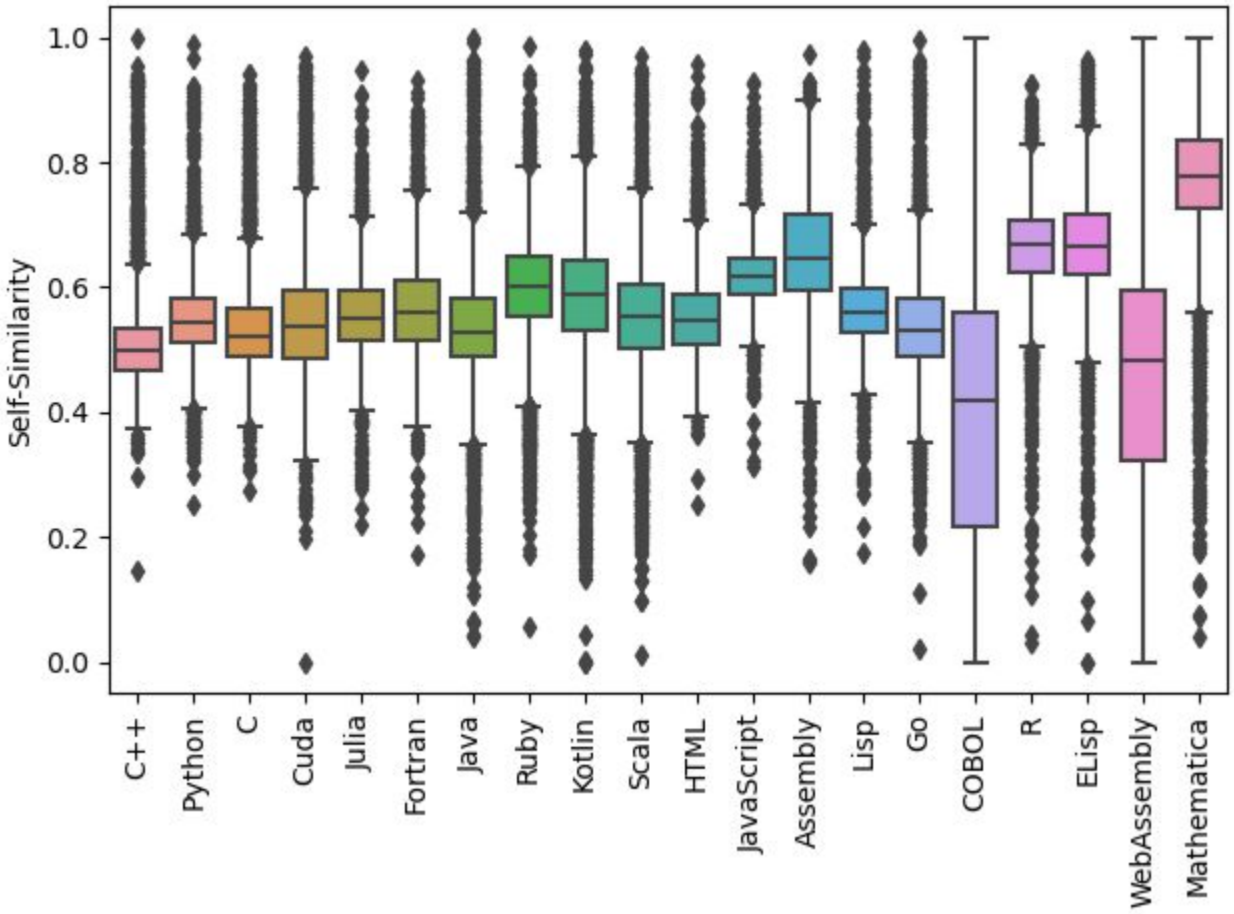
C++
C
Python
JavaScript
Cuda
Julia
Java
Go
Fortran
Kotlin
Scala
Ruby
HTML
Lisp
WebAssembly
ELisp
R
Assembly
Mathematica
COBOL
0.80
0.75
0.70
0.65
0.60
0.55
0.50

TU Delft

Ir


Self-Similarity
1.0
0.8
0.6
0.4
0.2
0.0
C++
C
Python
JavaScript
Cuda
Julia
Java
Go
Fortran
Kotlin
Scala
Ruby
HTML
Lisp
WebAssembly
ELisp
R
Assembly
Mathematica
COBOL

Im



Self-Similarity
1.0
0.8
0.6
0.4
0.2
0.0
C++
Python
C
Cuda
Julia
Fortran
Java
Ruby
Kotlin
Scala
HTML
JavaScript
Assembly
Lisp
Go
COBOL
R
ELisp
WebAssembly
Mathematica

TU Delft