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

Jonathan Katzy, Maliheh Izadi, Arie van Deursen
Intro

- Why do LLMs perform worse in some languages?

- Does language choice matter?
Goals

• Map language similarities

• Identify distinct groupings of languages
Approach - Representation

- BERT Representation
- “Token” set of representations
- “Language” Set of Tokens
Approach - Comparison

• Similarity between
  – Languages
  – Tokens
  – Representations
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
Results

- Common languages are similar
Results

- Common languages are similar
- Others are not
Results

• Domain specific languages differ a little
Results

• Pretraining makes representation more consistent
Conclusion

• There are consistent differences

• Use-case more important than grammar

• Implications
  – Transfer learning
  – Fine-tuning
  – Low resource languages
Future work

• More architectures

• Correlation to performance

• Analyze downstream tasks
Questions?

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