Bachelor Thesis:
Automatic Token Classification for Unknown Languages

Jan Kurš, Joel Guggisberg
1 Introduction

• Given code of an unknown programming language, attempt to automatically recognize which are the keywords of the language.

• To find said keywords assume that many programming languages have common constructs

```java
package autoca.mode;
import Tokenizer;

public final class AnalyzeMode
    implements IOperationMode {

    private DB db;
    private static final Logger logger =
        Logger.getLogger(AnalyzeMode.class);

    public AnalyzeMode(JSONInterface data) {
        try {
            this.db = new Database(data);
        }
        catch (SQLException e) {
            logger.error("Analyze Mode", e);
        }
    }
```
2 Architecture

Scan Mode

Dataset
- Scan settings
- Analyze settings
- Source files

```java
package autoca.mode;
import Tokenizer;

public final class AnalyzeMode implements IOperationMode {
    private DB db;
    private static final Logger logger =
        Logger.getLogger(AnalyzeMode.class);

    public AnalyzeMode(JSONInterface data)
    {
        try {
            this.db = new Database(data);
        } catch (SQLException e) {
            logger.error("Analyze Mode", e);
        }
    }
}
```
3 Database

Scan Mode  →  Database  →  Analyze Mode

<table>
<thead>
<tr>
<th>Token</th>
<th>File</th>
<th>OrderId</th>
</tr>
</thead>
<tbody>
<tr>
<td>package</td>
<td>AnalyzeMode.java</td>
<td>1</td>
</tr>
<tr>
<td>autoca</td>
<td>AnalyzeMode.java</td>
<td>2</td>
</tr>
<tr>
<td>#unknown</td>
<td>AnalyzeMode.java</td>
<td>3</td>
</tr>
<tr>
<td>mode</td>
<td>AnalyzeMode.java</td>
<td>4</td>
</tr>
<tr>
<td>#unknown</td>
<td>AnalyzeMode.java</td>
<td>5</td>
</tr>
<tr>
<td>#newline</td>
<td>AnalyzeMode.java</td>
<td>6</td>
</tr>
<tr>
<td>import</td>
<td>AnalyzeMode.java</td>
<td>7</td>
</tr>
<tr>
<td>Tokenizer</td>
<td>AnalyzeMode.java</td>
<td>8</td>
</tr>
<tr>
<td>#unknown</td>
<td>AnalyzeMode.java</td>
<td>9</td>
</tr>
<tr>
<td>#newline</td>
<td>AnalyzeMode.java</td>
<td>10</td>
</tr>
</tbody>
</table>
4 Analyze methods

Global
The keywords appear most commonly in the source code

Coverage
The token that appear most commonly in different files are keywords

Indent
The token that appear most commonly at the beginning of a line before an indent are keywords

Newline
The token that appear most commonly at the first position of a new line are keywords

Scan Mode → Database → Analyze Mode

Dataset
Scan settings
Analyze settings
Source files
Keywords in Java: 50
Projects: 179
Files: 100'764
Distinct tokens: 414’334
Occurences of tokens: 92’036’362

Global
The keywords appear most commonly over all source code

Newline
The token that appear in most files are keywords

Coverage
The token that appear in most files are keywords

Indent
The token at the beginning of a line before an indent are keywords

Precision = \frac{True Positive}{True Positive + False Positive}
6 Filters

How can we improve those results?

<table>
<thead>
<tr>
<th>TOKEN</th>
<th>COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>2</td>
</tr>
<tr>
<td>final</td>
<td>2</td>
</tr>
<tr>
<td>class</td>
<td>2</td>
</tr>
<tr>
<td>private</td>
<td>2</td>
</tr>
<tr>
<td>import</td>
<td>1</td>
</tr>
</tbody>
</table>

**Scan mode filter**: Removes all tokens marked by the scan mode.

**Intersection filter**: Counts in how many projects a token occurs and removes the tokens that don’t occur in enough projects. Used to remove project specific pollution.

**Upper case filter**: Removes all tokens containing capital letters. Since in Java and many other languages keywords are written in lower-case letters.
Keywords in Java: 50
Projects: 179
Files: 100’764
Distinct tokens: 414’334
Occurrences of tokens: 92’036’362

Global
The keywords appear most commonly over all source code

Coverage
The token that appear in most files are keywords

Newline
The token that appear in most files are keywords

Indent
The token at the beginning of a line before an indent are keywords

\[ \text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \]
Keywords in Java: 50
Projects: 179
Files: 100,764
Distinct tokens: 414,334
Occurrences of tokens: 92,036,362

Coverage
The token that appear in most files are keywords

Intersection filter: Counts in how many projects a token occurs and removes the tokens that don’t occur in enough projects. Used to remove project specific pollution.

Precision = \frac{True Positive}{True Positive + False Positive}
8 Summary