

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

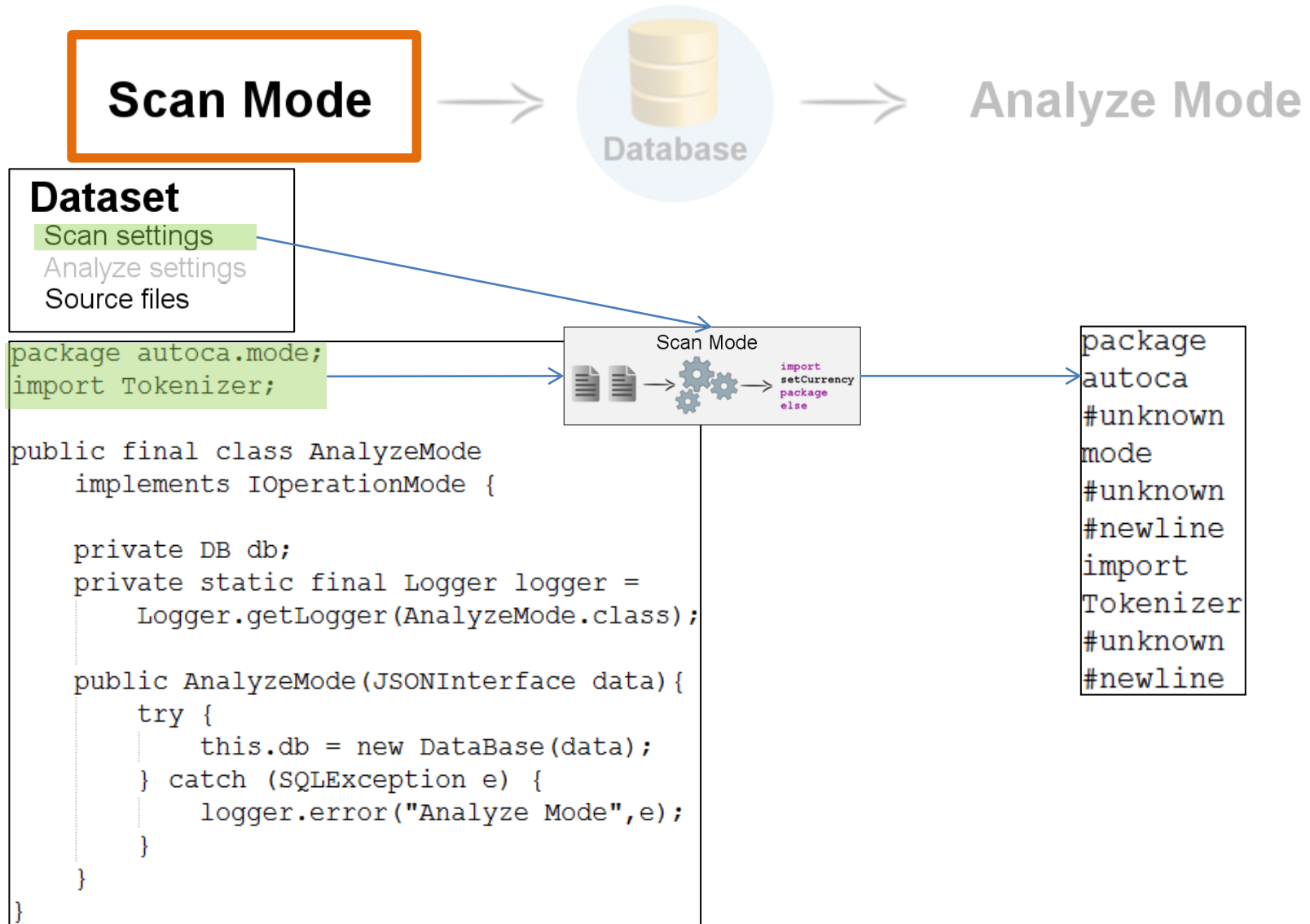
```
package autoca.mode;
import Tokenizer;

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



3 Database



	Occurences		
	Token	File	OrderId
package	package	AnalyzeMode.java	1
autoca	autoca	AnalyzeMode.java	2
#unknown	#unknown	AnalyzeMode.java	3
mode	mode	AnalyzeMode.java	4
#unknown	#unknown	AnalyzeMode.java	5
#newline	#newline	AnalyzeMode.java	6
import	import	AnalyzeMode.java	7
Tokenizer	Tokenizer	AnalyzeMode.java	8
#unknown	#unknown	AnalyzeMode.java	9
#newline	#newline	AnalyzeMode.java	10

4 Analyze methods

Scan Mode



Analyze Mode

Global
The keywords appear most commonly in the source code

Dataset
Scan settings
Analyze settings
Source files

Coverage

Occurrences

Token	File	OrderId
package	AnalyzeMode.java	1
autoca	AnalyzeMode.java	2
#unknown	AnalyzeMode.java	3
mode	AnalyzeMode.java	4
#unknown	AnalyzeMode.java	5
#newline	AnalyzeMode.java	6
import	AnalyzeMode.java	7
Tokenizer	AnalyzeMode.java	8
#unknown	AnalyzeMode.java	9
#newline	AnalyzeMode.java	10
⋮	⋮	⋮

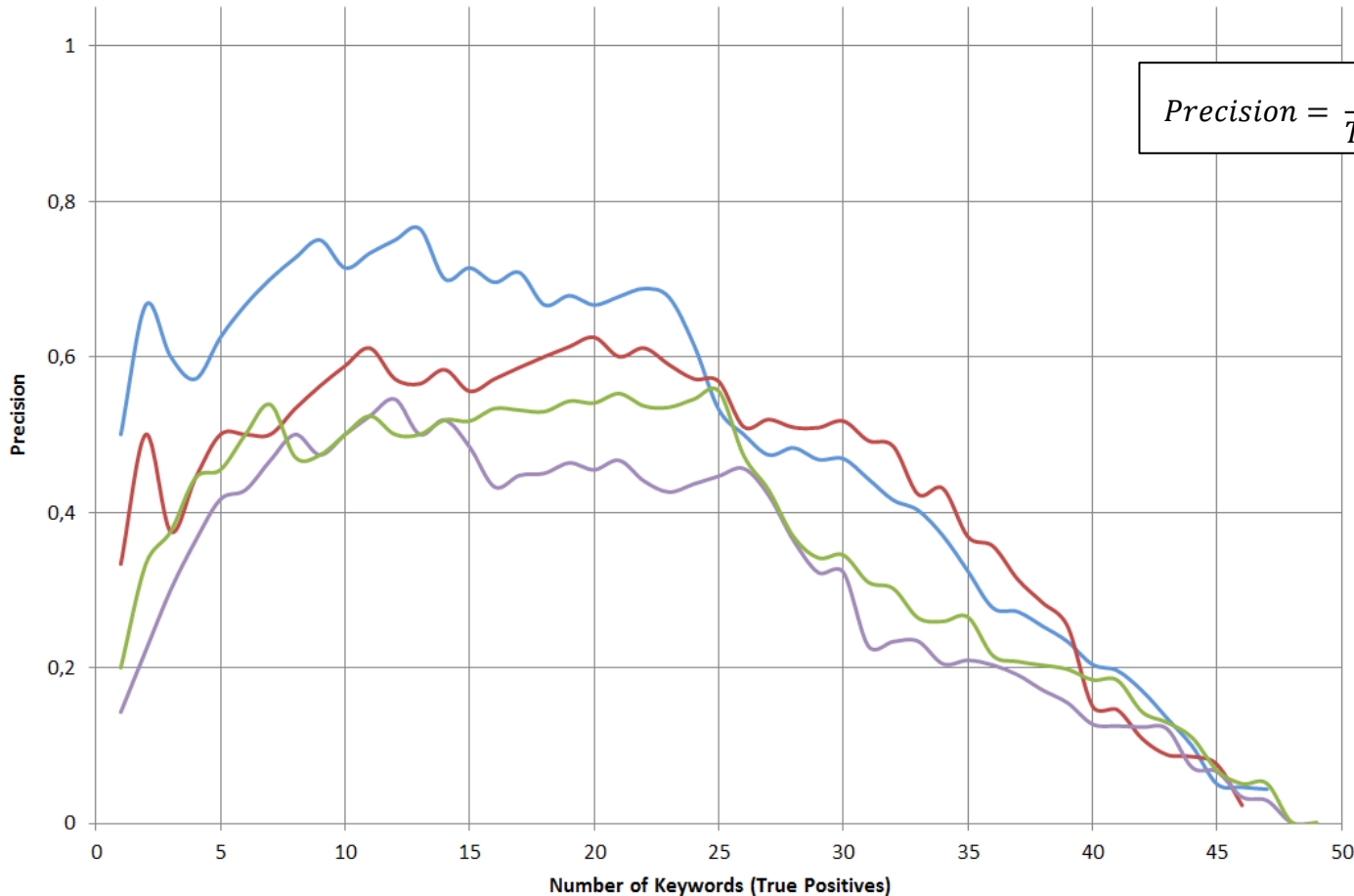
```

package
import
public
import
private
private
public
data
Logger
final
class
db
private
logger
IOperationMode
mode
import
    
```

TOKEN	COUNT
#unknown	34
#newline	19
#dedent	6
#indent	6
AnalyzeMode	3
public	2
data	2
Logger	2
final	2
class	2
db	2
private	2
logger	2
IOperationMode	1
mode	1
import	1

static	1
JSONInterface	1
implements	1
#string	1
Tokenizer	1
DataBase	1
catch	1
DB	1
SQLException	1
try	1
package	1
error	1
getLogger	1
autoca	1
this	1
new	1

5 Java result of the hypothesis



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

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

6 Filters

How can we improve those results?

Global result

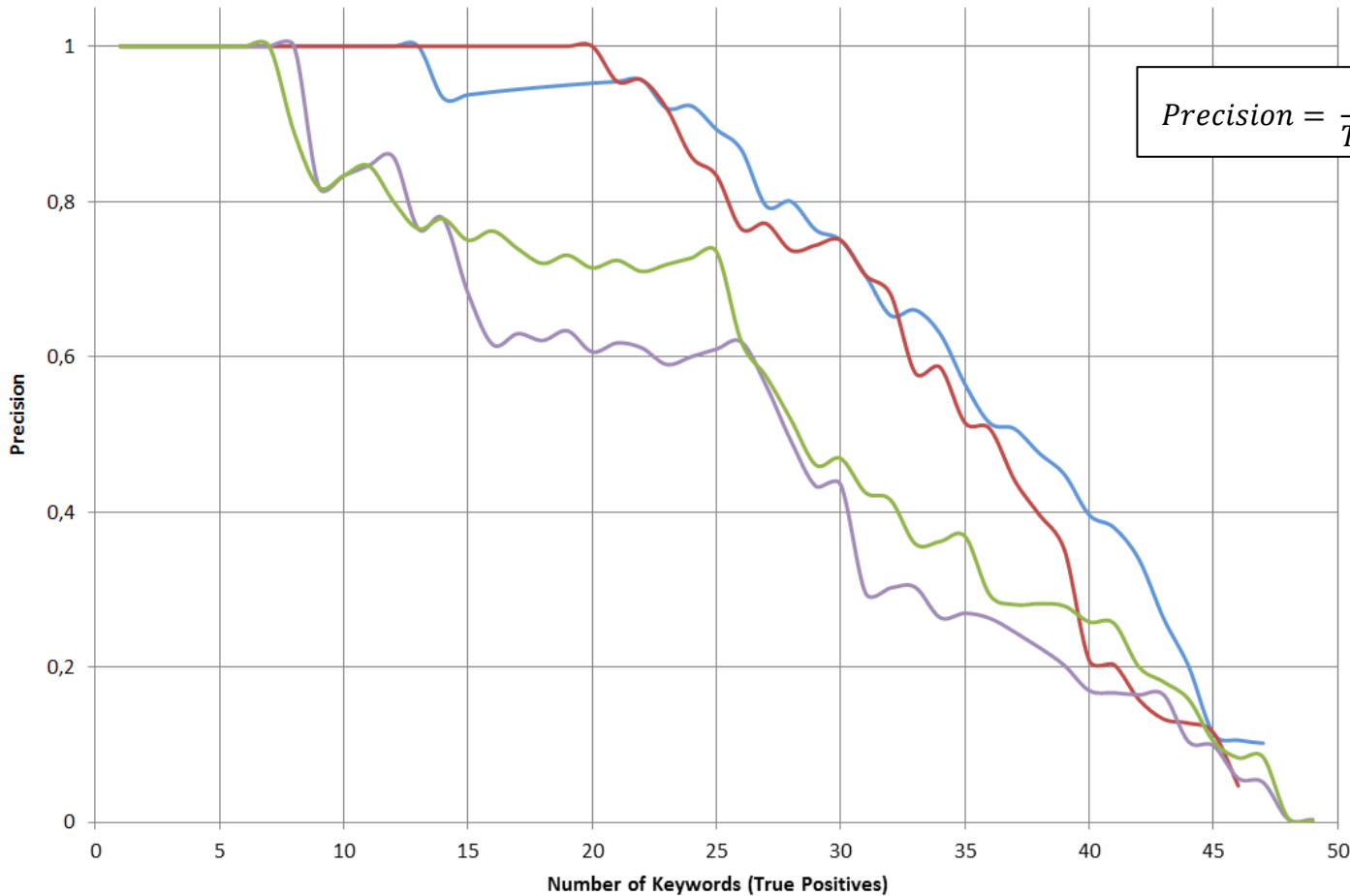
TOKEN	COUNT
static	1
implements	1
public	2
catch	1
final	2
class	2
private	2
try	1
package	1
this	1
import	1
new	1

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.

7 Java results filtered



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

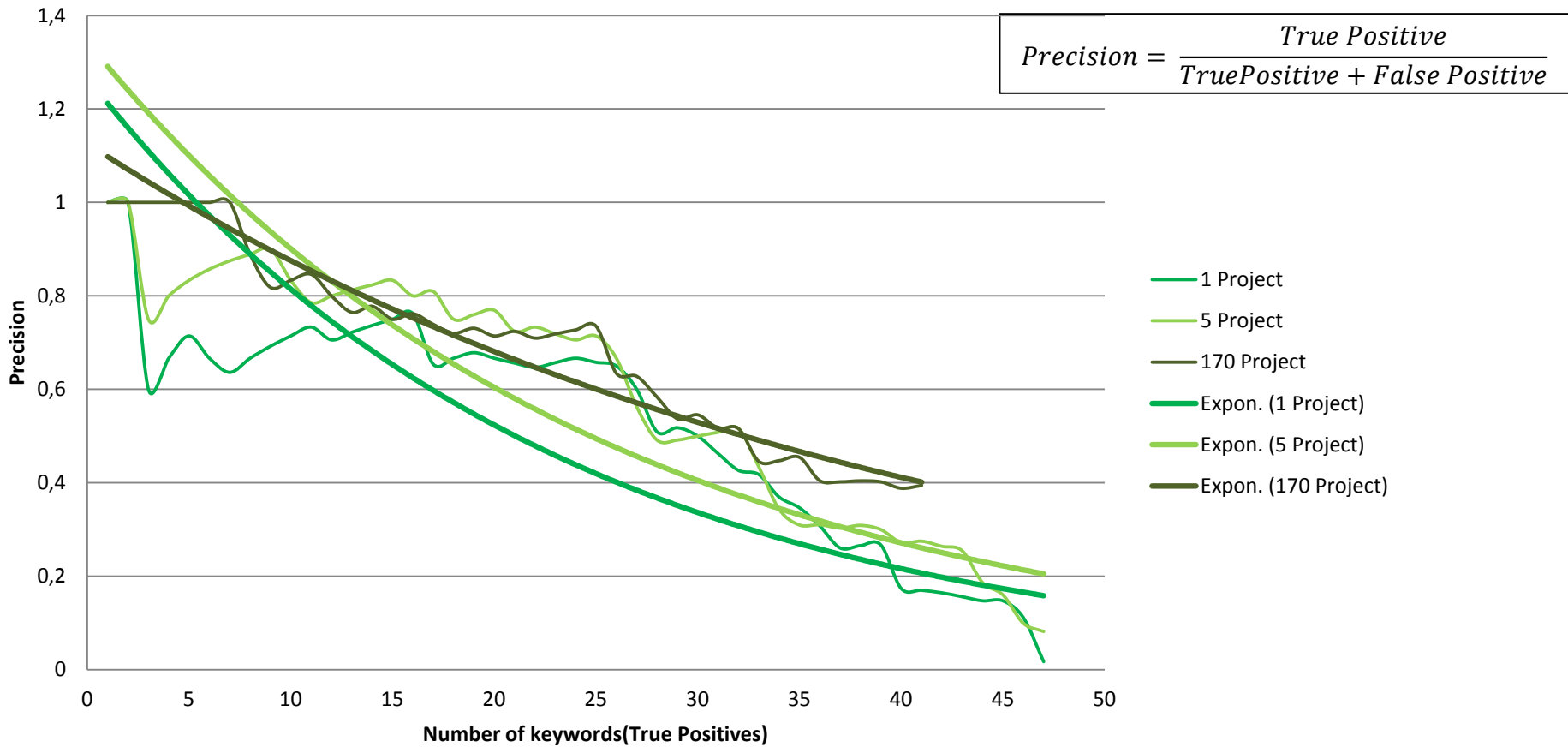
Newline

The token that appear in most files are keywords

Indent

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

7 More data better Results?



Keywords in Java: 50
Projects: 179
Files: 100'764
Distinct tokens: 414'334
Occurences 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.

8 Summary