Guesslang

Release 2.2.2

Sep 22, 2021

Contents

Guesslang detects the programming language of a given source code. It supports more than **50** programming **languages** and detects the correct programming language with more than **90% accuracy**.

Guesslang is an open source deep learning software that have been trained with **over a million** source code files.

You can use Guesslang as a command line interface tool or as a Python module:

```
from guesslang import Guess

guess = Guess()

# Guess the language from code
language = guess.language_name("""
    % Quick sort

    -module (recursion).
    -export ([qsort/1]).

    qsort([]) -> [];
    qsort([Pivot|T]) ->
           qsort([X || X <- T, X < Pivot])
           ++ [Pivot] ++
           qsort([X || X <- T, X >= Pivot]).
    """)

print(language)  # --> Erlang
```

Guesslang supports **54** of the **world's most popular** programming languages:

Assembly	Batchfile	C	C#	C++
Clojure	CMake	COBOL	CoffeeScript	CSS
CSV	Dart	DM	Dockerfile	Elixir
Erlang	Fortran	Go	Groovy	Haskell
HTML	INI	Java	JavaScript	JSON
Julia	Kotlin	Lisp	Lua	Makefile
Markdown	Matlab	Objective-C	OCaml	Pascal
Perl	PHP	PowerShell	Prolog	Python
R	Ruby	Rust	Scala	Shell
SQL	Swift	TeX	TOML	TypeScript
Verilog	Visual Basic	XML	YAML	

Guesslang is used by Visual Studio Code to automatically detect the programming language of the source code that you paste into the editor:

Fig. 1: Visual Studio Code automatic language detection.

Guesslang is used by other projects including:

- Chameledit a web-editor that auto-highlights code,
- Pasta a Slack bot that pretty-pastes code,
- GG a guessing game.

CHAPTER 1

Install Guesslang

Guesslang requires **Python 3.7 or later**.

1.1 Install from Pypi

You can run the following command to install Guesslang on your system:

```
pip install guesslang
```

1.2 Install from source code

To install Guesslang from source code, just download the source code from https://github.com/yoeo/guesslang, then run this command:

```
pip install .
```

CHAPTER 2

Usage

2.1 Python package

Guesslang Python library helps you detect the programming language of a given text within your Python program. The Python classes are fully documentation here: *Guesslang package reference*.

2.2 Command line tool

On a terminal emulator, you can detect the programming language of a source code file by running `guesslang /path/to/file`.

As well, you can detect the programming language of a source code provided through the standard input using a pipeline like `some-command | guesslang`.

Examples:

- Detect the programming language of `/etc/bashrc` configuration file

```
guesslang /etc/bashrc

#  Programming language: Shell
```

- Detect the programming language of a source code stored in a file

```
echo "
  class Array
    def quick_sort
      return self if length <= 1
      pivot = self[0]
      less, greatereq = self[1..-1].partition { |x| x < pivot }
      less.quick_sort + [pivot] + greatereq.quick_sort
    end
  end
```

(continues on next page)

(continued from previous page)

```
" > /tmp/quicksort

guesslang /tmp/quicksort

#  Programming language: Ruby
```

- Execute a command that generates source code then detect the programming language on the fly:

```
echo '
  Array.prototype.quick_sort = function () {
     if (this.length < 2) { return this; }

     var pivot = this[Math.round(this.length / 2)];

     return this.filter(x => x <  pivot)
                .quick_sort()
                .concat(this.filter(x => x == pivot))
                .concat(this.filter(x => x >  pivot).quick_sort());
  };
' | guesslang

#  Programming language: JavaScript
```

- Show the programming language detection confidence score as probabilities:

```
echo "
def qsort(items):
    if not items:
        return []
    else:
        pivot = items[0]
        less = [x for x in items if x <  pivot]
        more = [x for x in items[1:] if x >= pivot]
        return qsort(less) + [pivot] + qsort(more)


if __name__ == '__main__':
    items = [1, 4, 2, 7, 9, 3]
    print(f'Sorted: {qsort(items)}')

" | guesslang --probabilities

# Language name       Probability
#  Python               74.80%
#  Haskell               6.73%
#  CoffeeScript          5.32%
#  Groovy                1.95%
#  Markdown              0.93%
#  ...
```

With Guesslang command line tool you can also show the detection **probabilities** for a given source code and even **train** your **custom** programming language detection model.

Run `guesslang --help` to see all the available options.

CHAPTER 3

How does Guesslang guess?

3.1 Deep learning Model

Guesslang uses a deep learning Tensorflow model built with **1,900,000** unique source code files, randomly picked from **170,000** public Github projects.

Guesslang model is a Deep Neural Network classifier combined with Linear classifier. The model's hyperparameters have been fine tuned to have both the best **performances** and the best **generalization**.

3.2 Training

Having a data set with a **very large** number of **diverse** examples is essential to correctly train a model. This large dataset is built with GuesslangTools. It is used to train, evaluate and test Guesslang's deep learning model.

To avoid overfitting, each repositories is **strictly** associated with only one of the 3 aforementioned tasks. Therefore files from a repository assigned to the training task can only be used to train the model and cannot be used to evaluate nor test it.

The training and evaluation steps are done in a loop, as shown by the following loss curve.

The test in the other hand is done after the last training and evaluation steps to ensure that the final model performs well.

3.3 Accuracy

Guesslang deep learning model performs very well, with **93.45% accuracy**. This accuracy was calculated by testing Guesslang with 230,000 distinct source files.

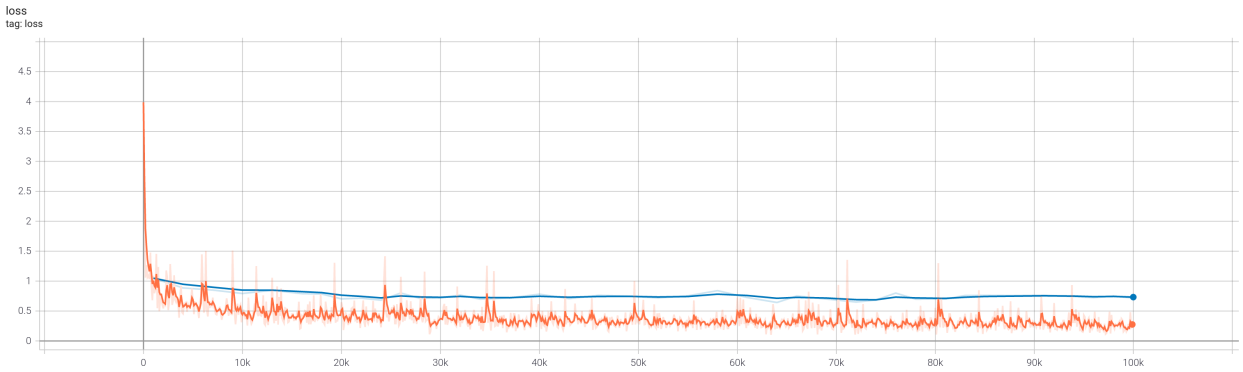

Fig. 1: — *Loss curve, less is better.*
training, evaluation.

3.4 Limitations

Guesslang accuracy is very high but it is not perfect.

Some challenging source codes that are at the border between two languages can fool Guesslang. In fact, a valid C source code is almost always a valid C++ code, and a valid JavaScript source code is always a valid TypeScript code.

This phenomenon is shown by Guesslang's confusion matrix:

In addition to that, Guesslang may not guess the correct programming languages of **very small** code snippets. Small snippets don't always provide enough insights to accurately guess the programming language.

For example, `print("Hello world")` is a valid code snippet in several programming languages including Python, Scala, Ruby, Lua, Perl, etc…

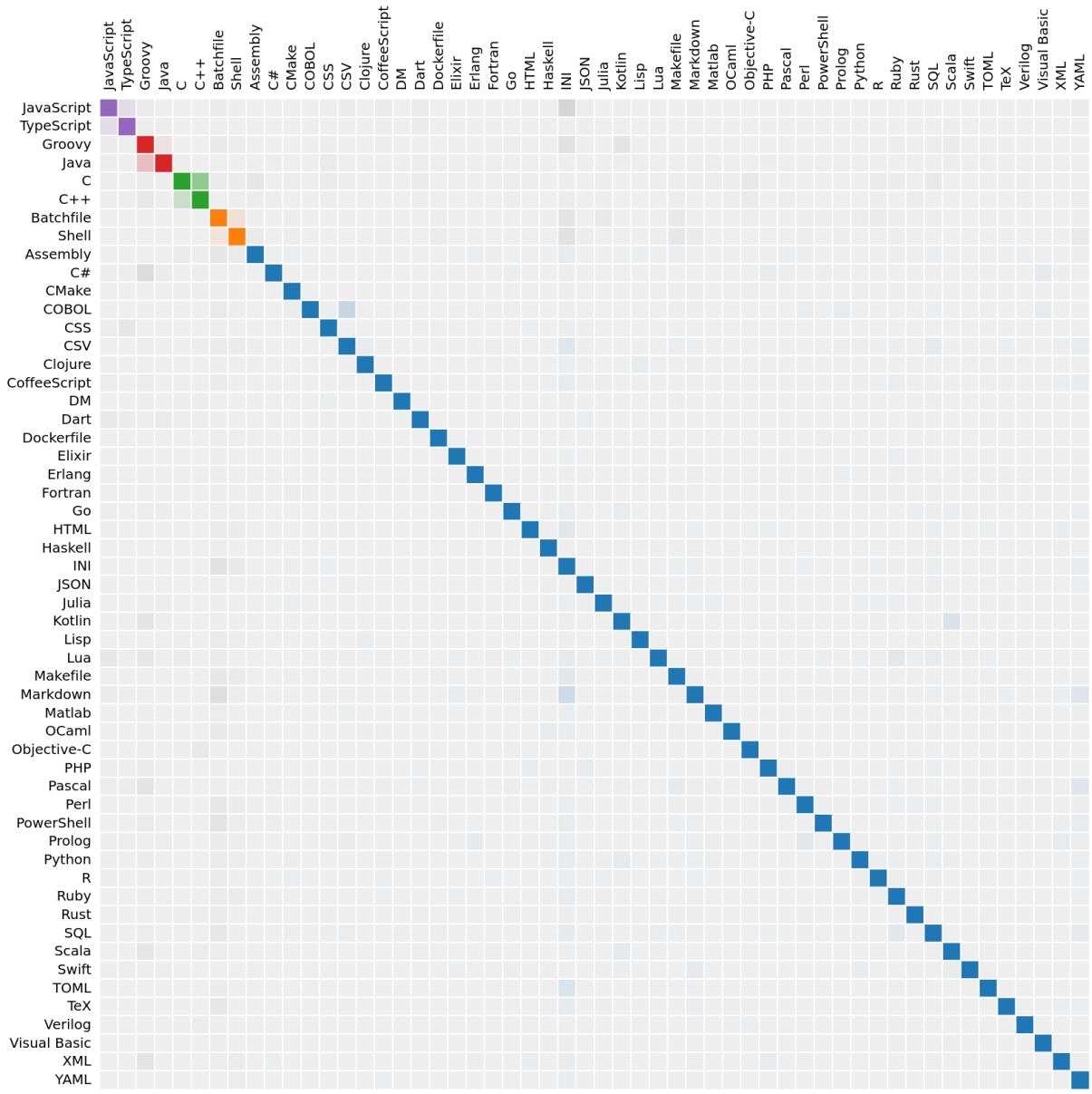

Fig. 2: — Lines: actual languages. Columns: guessed languages.

CHAPTER 4

References

- Guesslang source code is on Github.
- Guesslang is developped with Tensorflow machine learning framework.
- Use GuesslangTools to build your own training dataset.
- The example codes used in this documentation come from Rosetta Code.
- Guesslang logo has been created with Android Asset Studio and Eduardo Tunni's Warnes font.
- Guesslang — Copyright (c) 2021 Y. SOMDA, MIT Licence.
- genindex
- modindex
- search

CHAPTER 5

Guesslang package reference

5.1 guesslang.Guess

class guesslang.**Guess**(*model_dir: Optional[str] = None*)

Guess the programming language of a source code.

Parameters **model_dir** – Guesslang machine learning model directory.

is_trained

Check if the current machine learning model is trained. Only trained models can be used for prediction.

Returns the model training status.

language_name(*source_code: str*) → Optional[str]

Predict the programming language name of the given source code.

Parameters **source_code** – source code.

Returns the language name or `None` if no programming language is detected.

probabilities(*source_code: str*) → List[Tuple[str, float]]

Gives the probability that the source code is written in each of the supported languages.

The probabilities are sorted from the most to the least probable programming language.

Parameters **source_code** – source code.

Returns list of language names associated with their probability.

supported_languages

List supported programming languages

Returns language name list.

train(*source_files_dir: str*, *max_steps: int*) → float

Train guesslang to recognize programming languages.

The machine learning model is trained from source code files. The files should be split in three subdirectories named "train", "valid" and "test".

Raises **GuesslangError** – when the training cannot be run.

Parameters **source_files_dir** – directory that contains the "train", "valid" and "test" datasets.

Returns training accuracy, a value between 0 and 1.

5.2 guesslang.GuesslangError

Guesslang exception class

CHAPTER 6

Guesslang documentation

Guesslang detects the programming language of a given source code. It supports more than **50** programming **languages** and detects the correct programming language with more than **90% accuracy**.

Guesslang is an open source deep learning software that have been trained with **over a million** source code files.

You can use Guesslang as a command line interface tool or as a Python module:

```
from guesslang import Guess

guess = Guess()

# Guess the language from code
language = guess.language_name("""
    % Quick sort

    -module (recursion).
    -export ([qsort/1]).

    qsort([]) -> [];
    qsort([Pivot|T]) ->
           qsort([X || X <- T, X < Pivot])
           ++ [Pivot] ++
           qsort([X || X <- T, X >= Pivot]).
    """)

print(language)  # --> Erlang
```

Guesslang supports **54** of the **world's most popular** programming languages:

Assembly	Batchfile	C	C#	C++
Clojure	CMake	COBOL	CoffeeScript	CSS
CSV	Dart	DM	Dockerfile	Elixir
Erlang	Fortran	Go	Groovy	Haskell
HTML	INI	Java	JavaScript	JSON
Julia	Kotlin	Lisp	Lua	Makefile
Markdown	Matlab	Objective-C	OCaml	Pascal
Perl	PHP	PowerShell	Prolog	Python
R	Ruby	Rust	Scala	Shell
SQL	Swift	TeX	TOML	TypeScript
Verilog	Visual Basic	XML	YAML	

Guesslang is used by Visual Studio Code to automatically detect the programming language of the source code that you paste into the editor:

Fig. 1: Visual Studio Code automatic language detection.

Guesslang is used by other projects including:

- Chameledit a web-editor that auto-highlights code,
- Pasta a Slack bot that pretty-pastes code,
- GG a guessing game.

CHAPTER 7

Install Guesslang

Guesslang requires **Python 3.7 or later**.

7.1 Install from Pypi

You can run the following command to install Guesslang on your system:

```
pip install guesslang
```

7.2 Install from source code

To install Guesslang from source code, just download the source code from https://github.com/yoeo/guesslang, then run this command:

```
pip install .
```

CHAPTER 8

Usage

8.1 Python package

Guesslang Python library helps you detect the programming language of a given text within your Python program. The Python classes are fully documentation here: *Guesslang package reference*.

8.2 Command line tool

On a terminal emulator, you can detect the programming language of a source code file by running `guesslang /path/to/file`.

As well, you can detect the programming language of a source code provided through the standard input using a pipeline like `some-command | guesslang`.

Examples:

- Detect the programming language of `/etc/bashrc` configuration file

```
guesslang /etc/bashrc

#  Programming language: Shell
```

- Detect the programming language of a source code stored in a file

```
echo "
  class Array
    def quick_sort
      return self if length <= 1
      pivot = self[0]
      less, greatereq = self[1..-1].partition { |x| x < pivot }
      less.quick_sort + [pivot] + greatereq.quick_sort
    end
  end
```

(continues on next page)

(continued from previous page)

```
" > /tmp/quicksort

guesslang /tmp/quicksort

#  Programming language: Ruby
```

- Execute a command that generates source code then detect the programming language on the fly:

```
echo '
  Array.prototype.quick_sort = function () {
     if (this.length < 2) { return this; }

     var pivot = this[Math.round(this.length / 2)];

     return this.filter(x => x <  pivot)
                .quick_sort()
                .concat(this.filter(x => x == pivot))
                .concat(this.filter(x => x >  pivot).quick_sort());
  };
' | guesslang

#  Programming language: JavaScript
```

- Show the programming language detection confidence score as probabilities:

```
echo "
def qsort(items):
    if not items:
        return []
    else:
        pivot = items[0]
        less = [x for x in items if x <  pivot]
        more = [x for x in items[1:] if x >= pivot]
        return qsort(less) + [pivot] + qsort(more)


if __name__ == '__main__':
    items = [1, 4, 2, 7, 9, 3]
    print(f'Sorted: {qsort(items)}')

" | guesslang --probabilities

# Language name       Probability
#  Python               74.80%
#  Haskell               6.73%
#  CoffeeScript          5.32%
#  Groovy                1.95%
#  Markdown              0.93%
#  ...
```

With Guesslang command line tool you can also show the detection **probabilities** for a given source code and even **train** your **custom** programming language detection model.

Run `guesslang --help` to see all the available options.

CHAPTER 9

How does Guesslang guess?

9.1 Deep learning Model

Guesslang uses a deep learning Tensorflow model built with **1,900,000** unique source code files, randomly picked from **170,000** public Github projects.

Guesslang model is a Deep Neural Network classifier combined with Linear classifier. The model's hyperparameters have been fine tuned to have both the best **performances** and the best **generalization**.

9.2 Training

Having a data set with a **very large** number of **diverse** examples is essential to correctly train a model. This large dataset is built with GuesslangTools. It is used to train, evaluate and test Guesslang's deep learning model.

To avoid overfitting, each repositories is **strictly** associated with only one of the 3 aforementioned tasks. Therefore files from a repository assigned to the training task can only be used to train the model and cannot be used to evaluate nor test it.

The training and evaluation steps are done in a loop, as shown by the following loss curve.

The test in the other hand is done after the last training and evaluation steps to ensure that the final model performs well.

9.3 Accuracy

Guesslang deep learning model performs very well, with **93.45% accuracy**. This accuracy was calculated by testing Guesslang with 230,000 distinct source files.

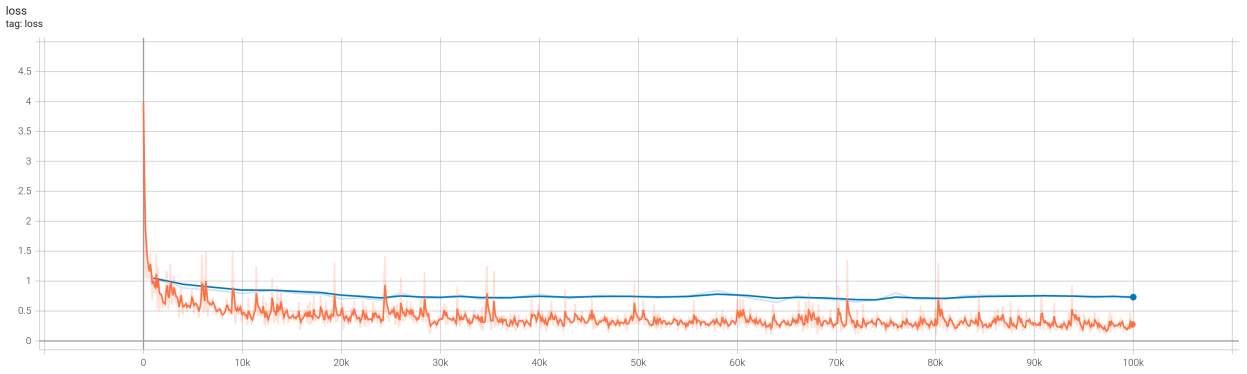

Fig. 1: — *Loss curve, less is better.*
training, evaluation.

9.4 Limitations

Guesslang accuracy is very high but it is not perfect.

Some challenging source codes that are at the border between two languages can fool Guesslang. In fact, a valid C source code is almost always a valid C++ code, and a valid JavaScript source code is always a valid TypeScript code.

This phenomenon is shown by Guesslang's confusion matrix:

In addition to that, Guesslang may not guess the correct programming languages of **very small** code snippets. Small snippets don't always provide enough insights to accurately guess the programming language.

For example, `print("Hello world")` is a valid code snippet in several programming languages including Python, Scala, Ruby, Lua, Perl, etc...

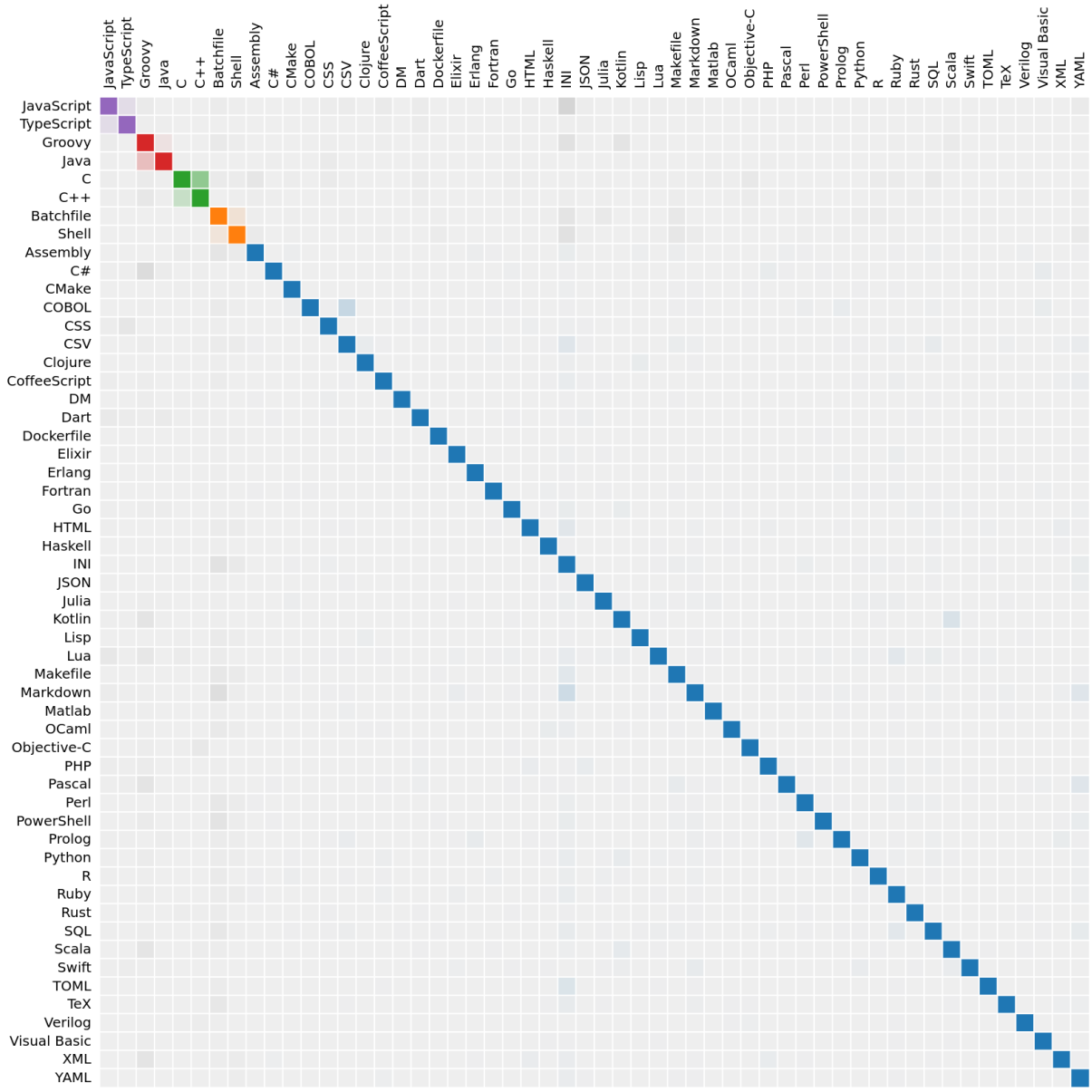

Fig. 2: — Lines: actual languages. Columns: guessed languages.

CHAPTER 10

References

- Guesslang source code is on Github.
- Guesslang is developped with Tensorflow machine learning framework.
- Use GuesslangTools to build your own training dataset.
- The example codes used in this documentation come from Rosetta Code.
- Guesslang logo has been created with Android Asset Studio and Eduardo Tunni's Warnes font.
- Guesslang — Copyright (c) 2021 Y. SOMDA, MIT Licence.
- genindex
- modindex
- search

Python Module Index

g

Index