ICPC 2017

EARLY RESEARCH ACHIEVEMENTS

REPLICATING PARSER BEHAVIOR USING NEURAL MACHINE TRANSLATION

Carol V. Alexandru, Sebastiano Panichella, Harald C. Gall
{alexandru,panichella,gall}@ifi.uzh.ch
23. May 2017

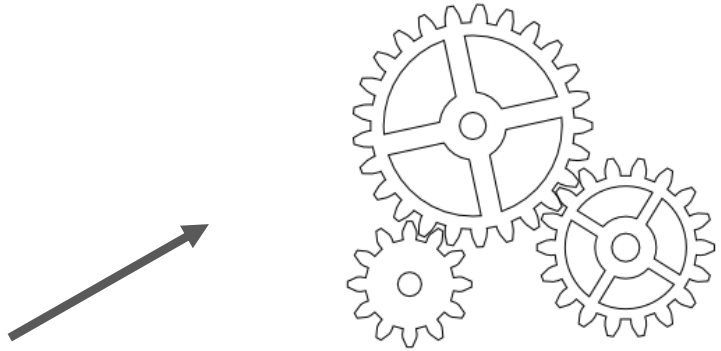
University of Zurich[UZH]

s.e.a.l. software evolution & architecture lab

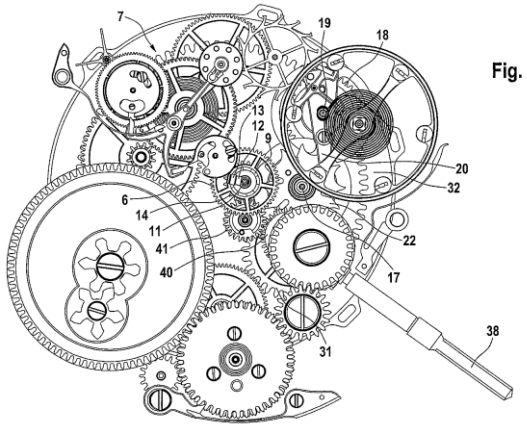
```
public int sum(int[] numbers) {
  int s = 0;
  for (int n : numbers) {
    s = s - n;
  }
  return s;
}
```

```
public int sum(int[] numbers) {
  int s = 0;
  for (int n : numbers) {
    s = s - n;
  }
  return s;
}
```

```
public int sum(int[] numbers) {
  int s = 0;
  for (int n : numbers) {
    s = s - n;
  }
  return s;
}
```

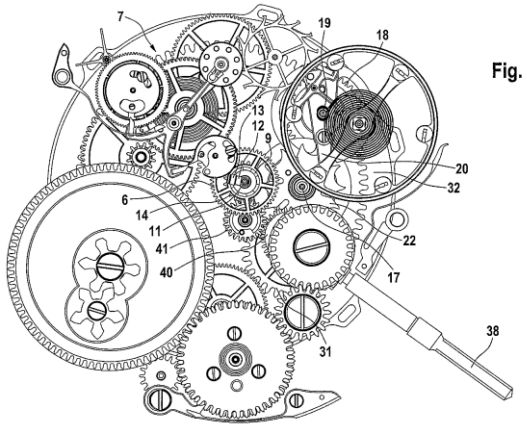


```
public int sum(int[] numbers) {
  int s = 0;
  for (int n : numbers) {
    s = s - n;
  }
  return s;
}
```



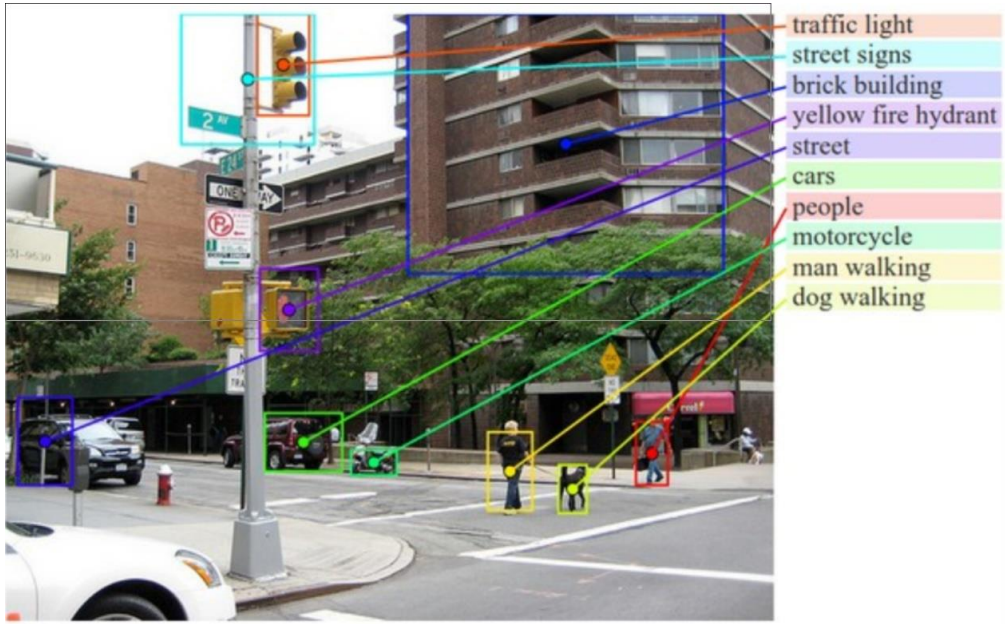
Even "simple" problems need complex solutions

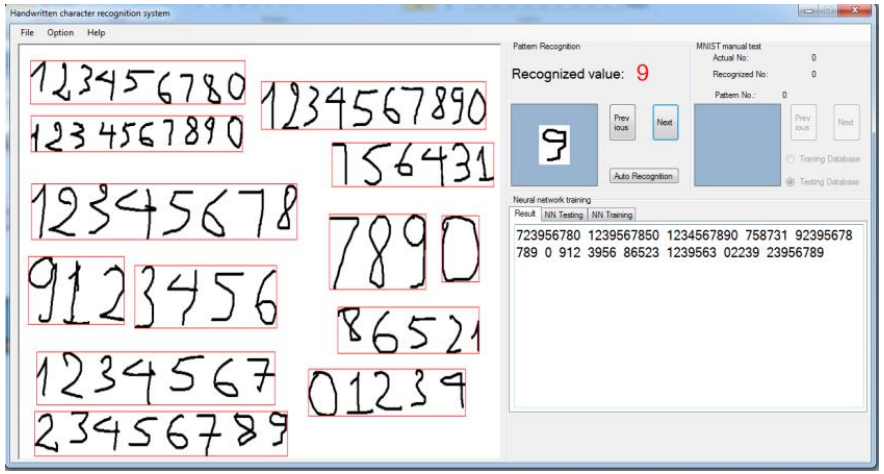
```
public int sum(int[] numbers) {
  int s = 0;
  for (int n : numbers) {
    s = s - n;
  }
  return s;
}
```

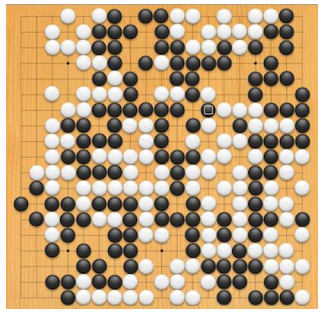


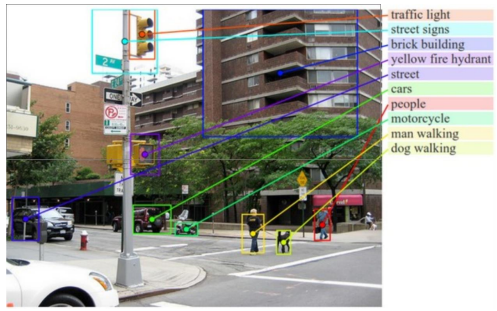
Even "simple" problems need complex solutions

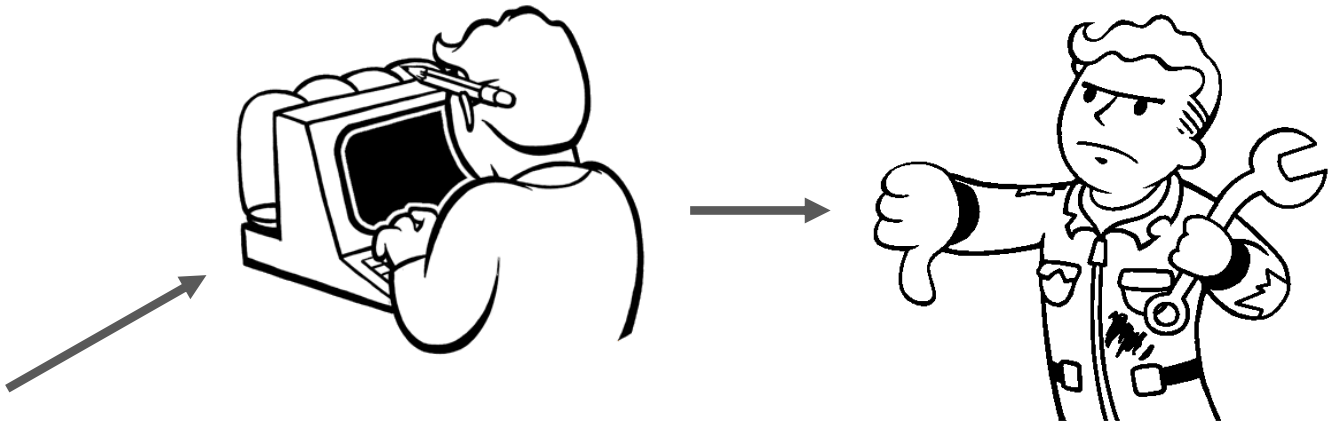
Unclear how exactly humans solve this problem

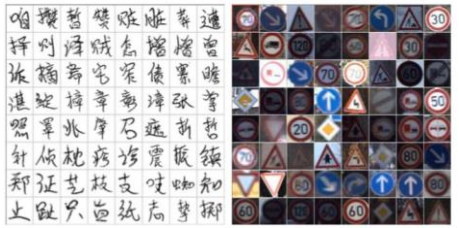

traffic light
street signs
brick building
yellow fire hydrant
street
cars
people
motorcycle
man walking
dog walking

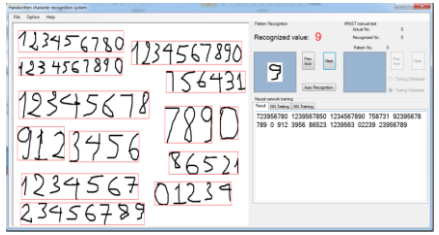

Handwritten character recognition system
File
Option
Help
Pattern Recognition
Recognized value: 9
MNIST manual test
Actual No:
Recognized No:
Pattern No.:
Previous
Next
Auto Recognition
Training Database
Testing Database
Neural network training
Result
NN Testing
NN Training
723956780 1239567850 1234567890 758731 92395678
789 0 912 3956 86523 1239563 02239 23956789

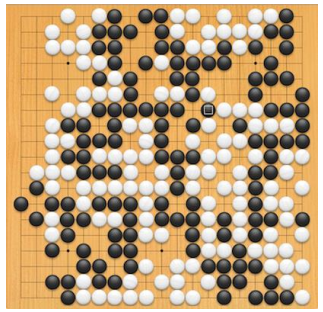


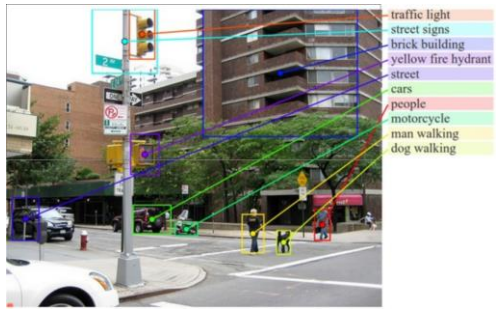

traffic light
street signs
brick building
yellow fire hydrant
street
cars
people
motorcycle
man walking
dog walking

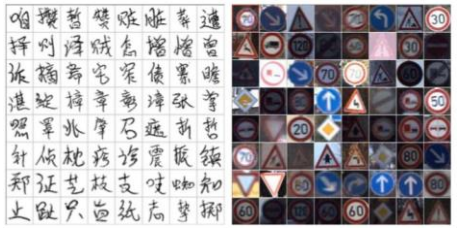


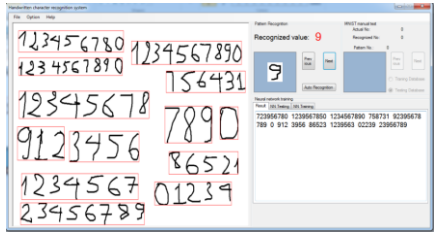


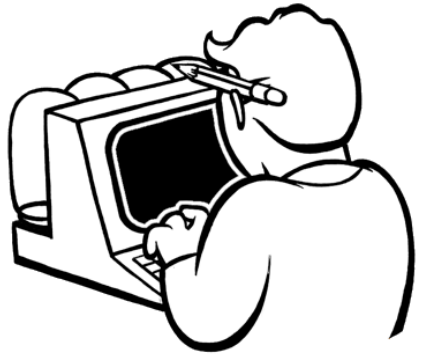

Recognized value: 9

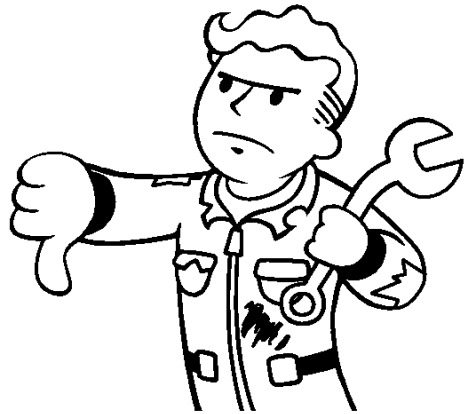


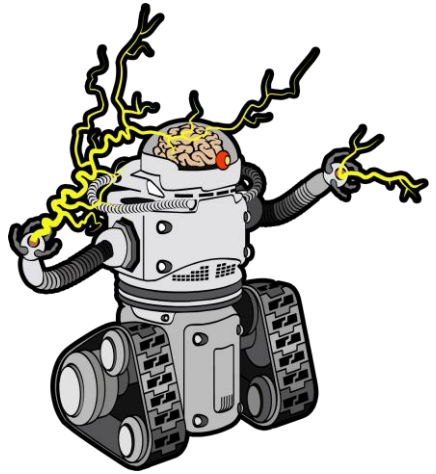

traffic light
street signs
brick building
yellow fire hydrant
street
cars
people
motorcycle
man walking
dog walking




Recognized value: 9







Where to begin?

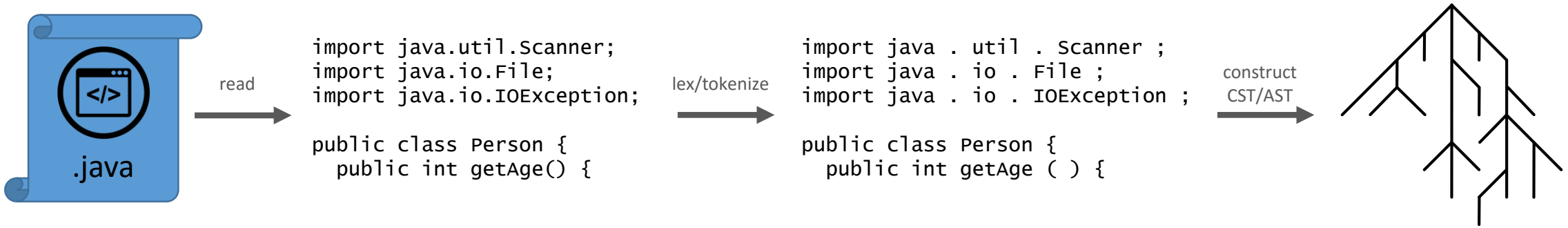
```
print("Hello world")
```

Where to begin?

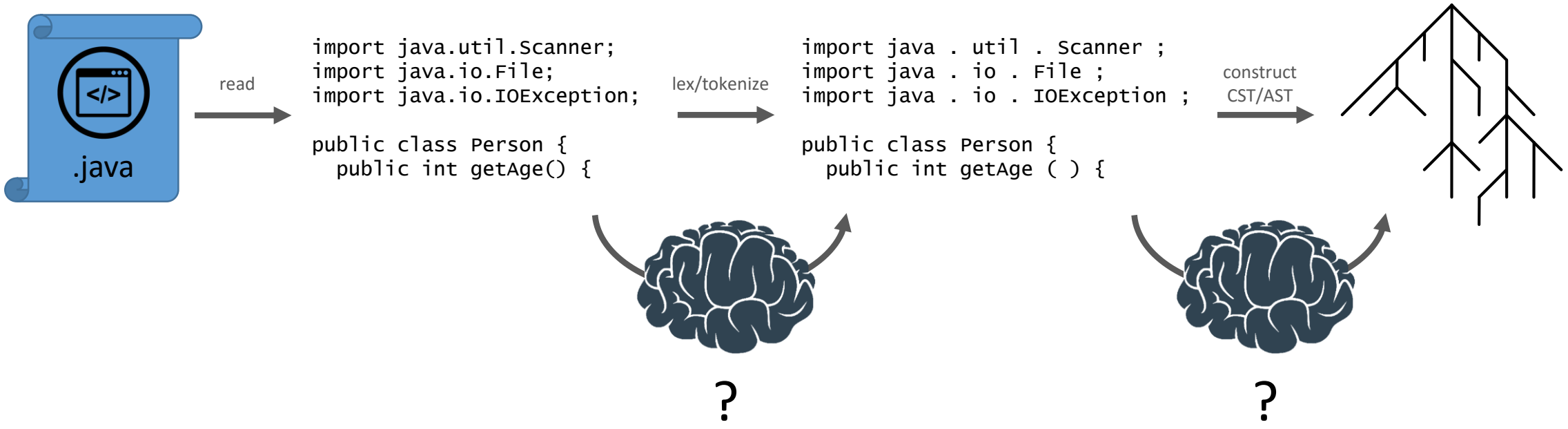
```
print("Hello world")
```

Can we teach a machine to "read" code?

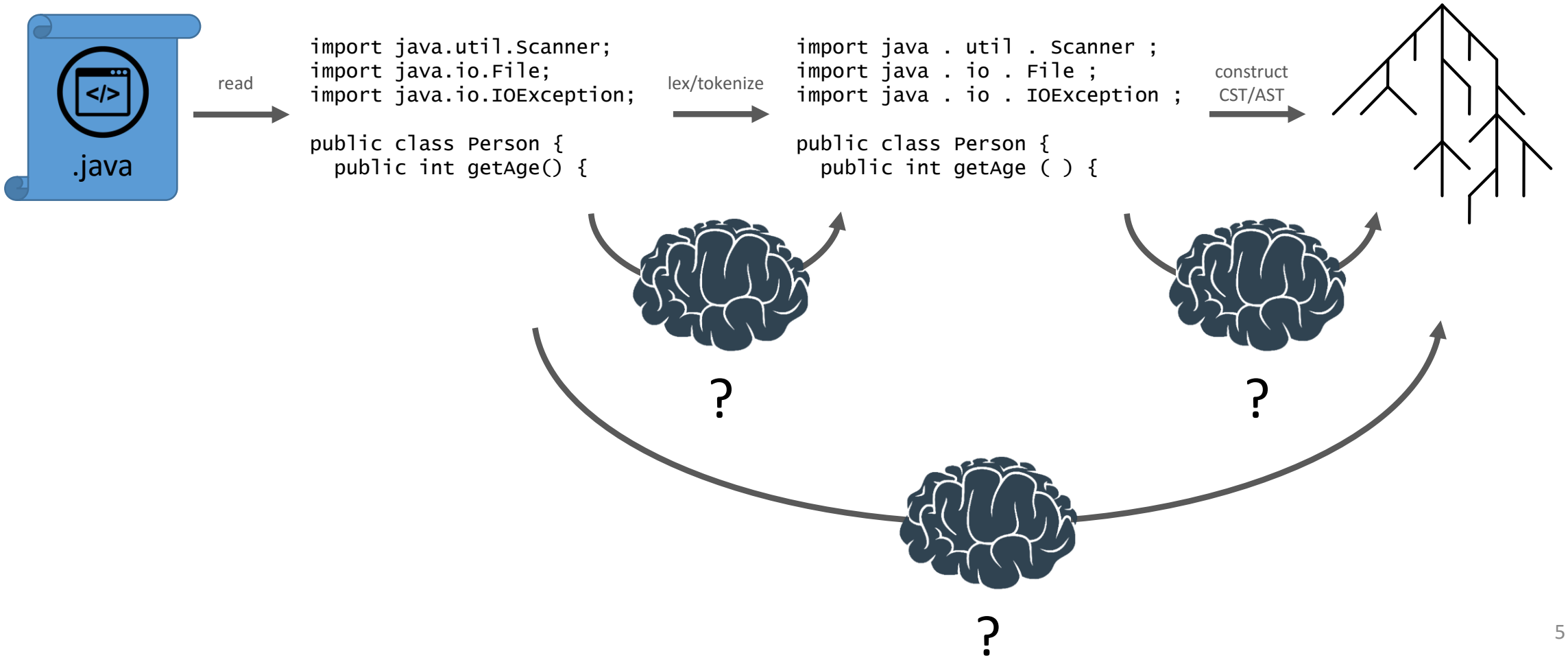
Replicating a Parser



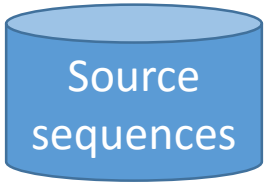
Replicating a Parser



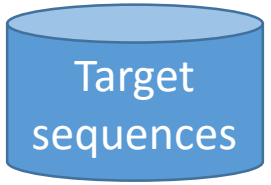
Replicating a Parser



Neural Machine Translation

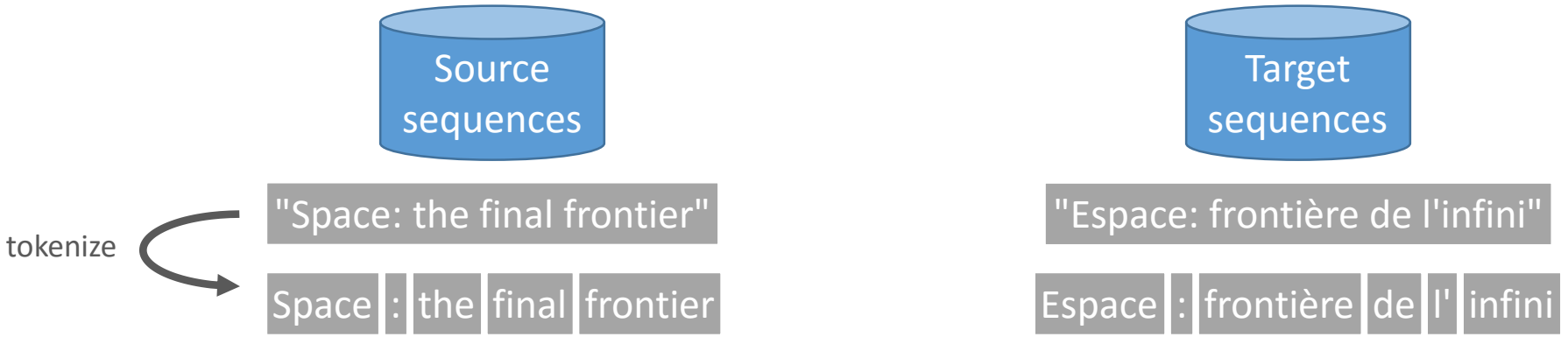


"Space: the final frontier"

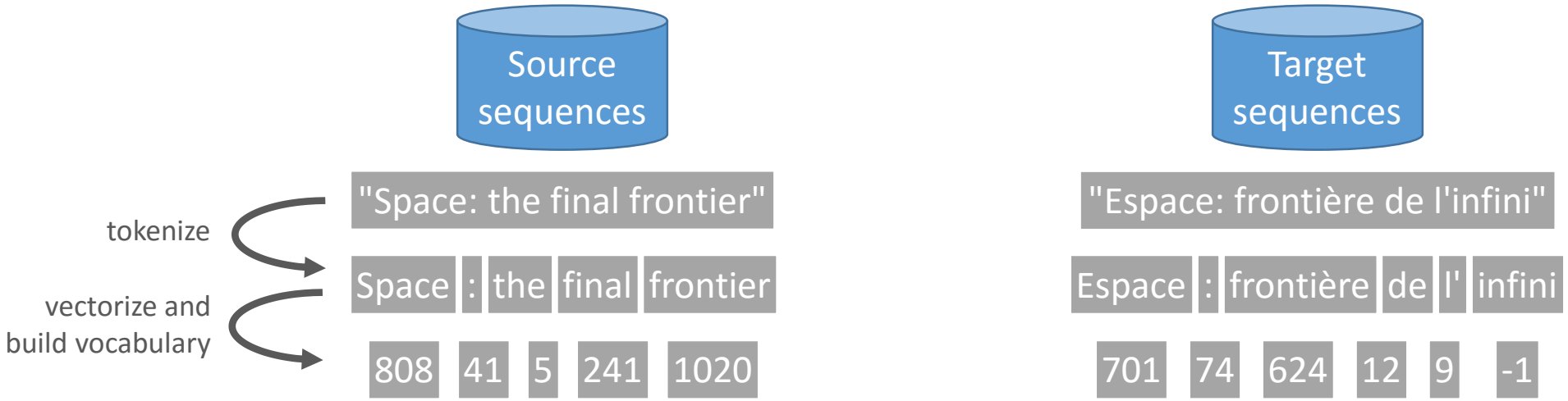


"Espace: frontière de l'infini"

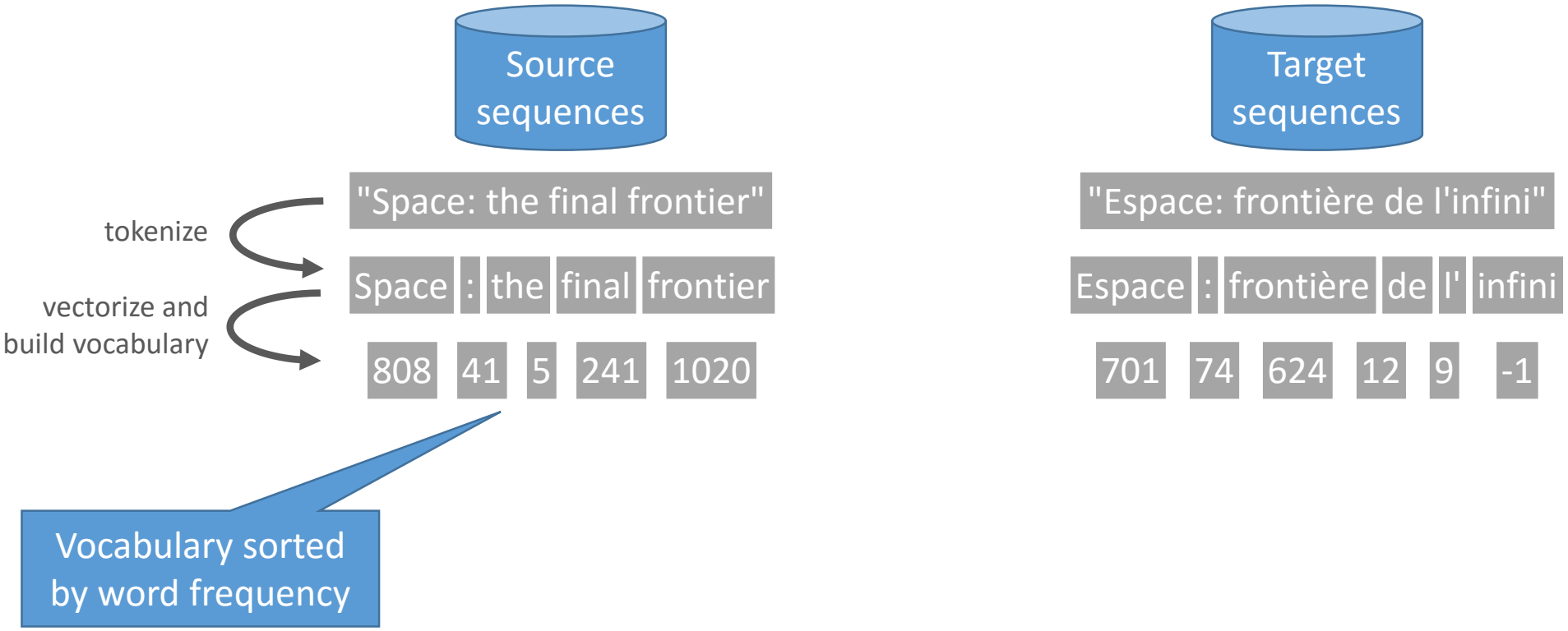
Neural Machine Translation



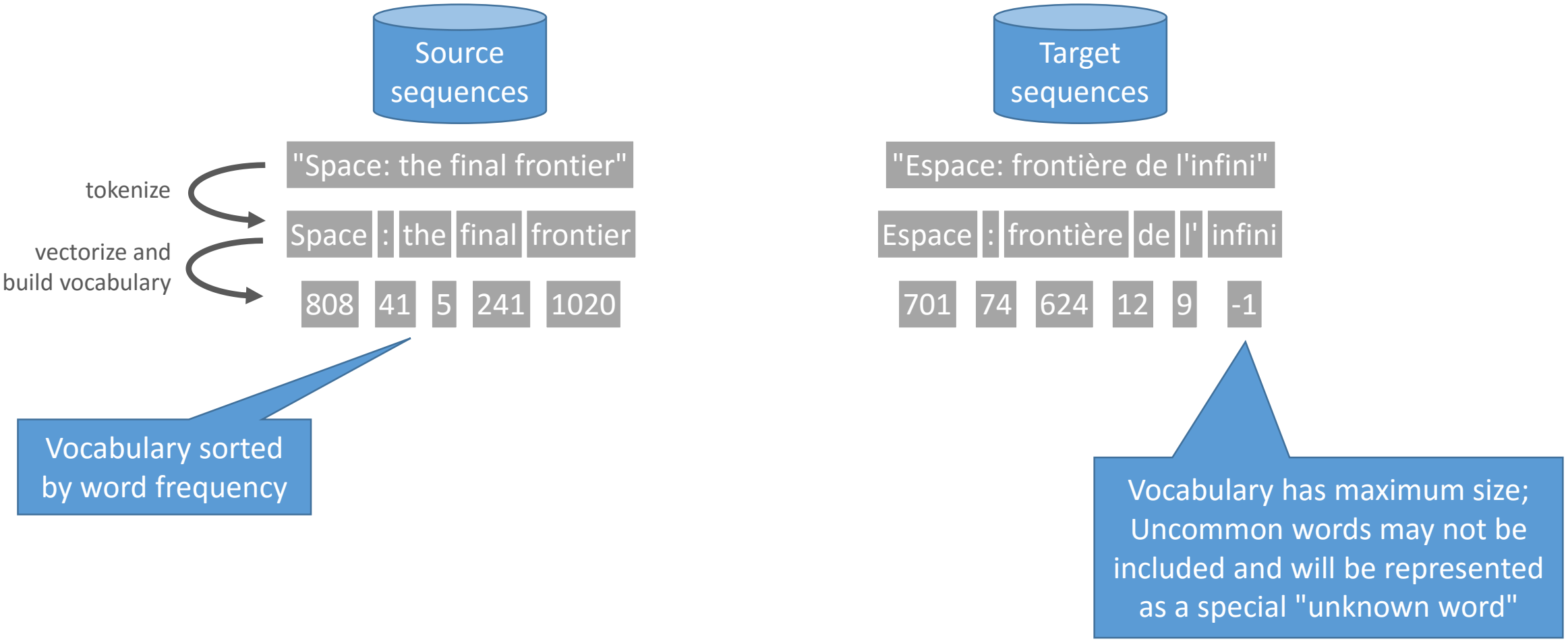
Neural Machine Translation



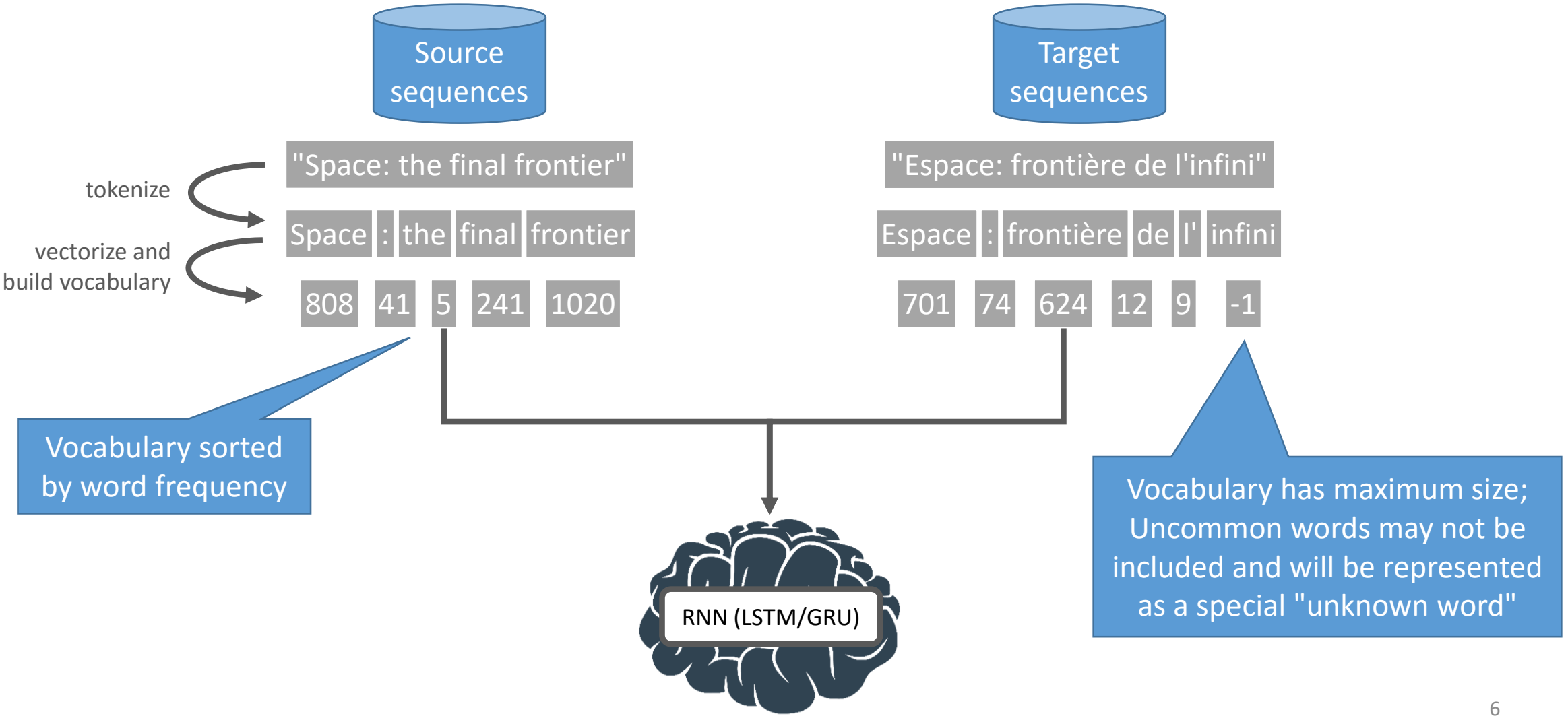
Neural Machine Translation



Neural Machine Translation



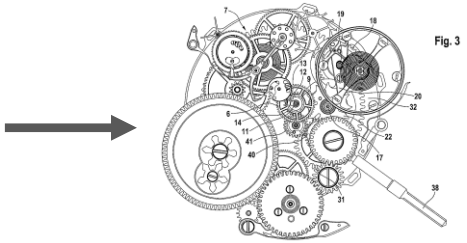
Neural Machine Translation



Data Gathering and Preparation

Data Gathering and Preparation

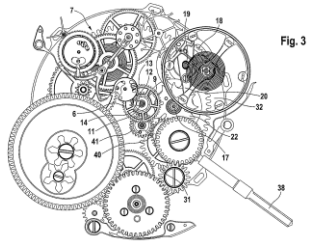
clone 1000 repos
language:java
sort:stars



parse (ANTLR)
and preprocess

Data Gathering and Preparation

clone 1000 repos
language:java
sort:stars



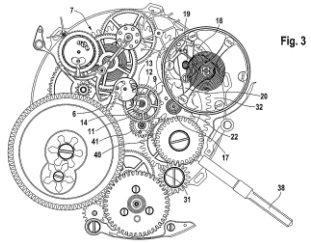
parse (ANTLR)
and preprocess

Plain text source

```
println("Hello World!");
```

Data Gathering and Preparation

clone 1000 repos
language:java
sort:stars



parse (ANTLR)
and preprocess

Plain text source

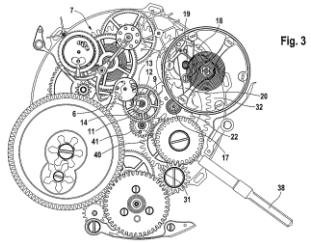
```
println("Hello␣World!");
```

1 char per word

Replace space words with unassigned Unicode char

Data Gathering and Preparation

clone 1000 repos
language:java
sort:stars



parse (ANTLR)
and preprocess

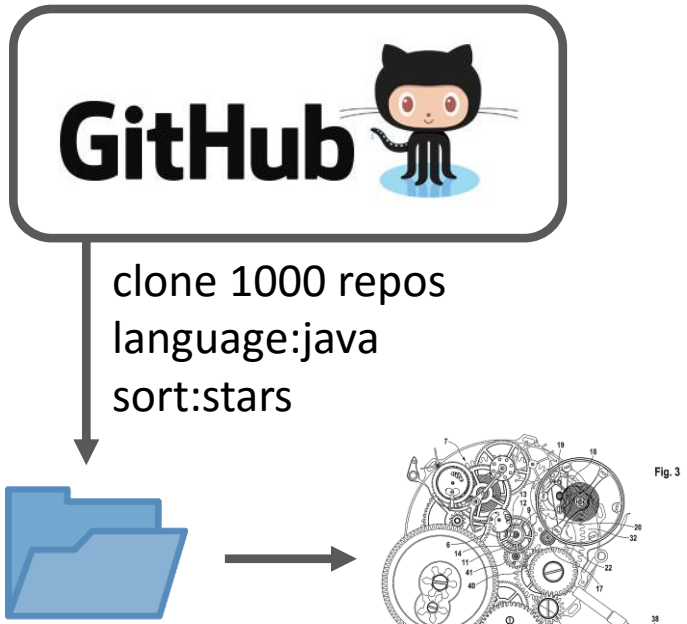
Plain text source

```
p r i n t l n ( " H e l l o ▯ W o r l d ! " ) ;
```

Lexing instructions

```
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
```

Data Gathering and Preparation



Plain text source

p	r	i	n	t	l	n	(	"	H	e	l	l	o	▯	W	o	r	l	d	!	"	)	;

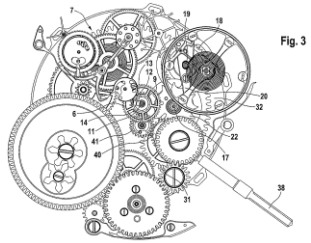
Lexing instructions

0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

0 = continue or start token
1 = end token
blank space = ignore character

Data Gathering and Preparation

clone 1000 repos
language:java
sort:stars



parse (ANTLR)
and preprocess

Plain text source

```
p r i n t l n ( " H e l l o ▯ W o r l d ! " ) ;
```

Lexing instructions

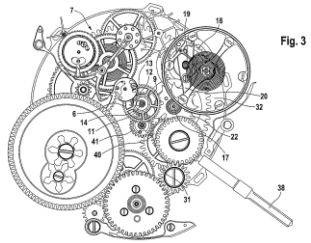
```
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
```

Why not translate to actual tokens?!

→ Target vocabulary would not contain all possible tokens (although there are ways around that...)

Data Gathering and Preparation

clone 1000 repos
language:java
sort:stars



parse (ANTLR)
and preprocess

Plain text source

```
p r i n t l n ( " H e l l o ▯ W o r l d ! " ) ;
```

Lexing instructions

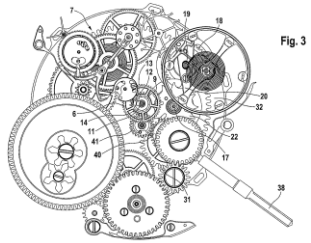
```
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
```

Tokens

```
println ( "Hello,▯world" ) ;
```

Data Gathering and Preparation

clone 1000 repos
language:java
sort:stars



parse (ANTLR)
and preprocess

Plain text source

```
p r i n t l n ( " H e l l o ␣ W o r l d ! " ) ;
```

Lexing instructions

```
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
```

Replace spaces in words with unassigned Unicode char

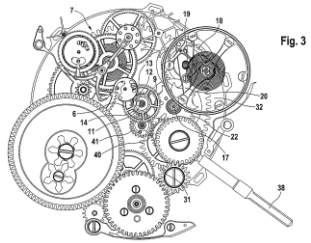
Tokens

```
println ( "Hello,␣world" ) ;
```

1 token per word

Data Gathering and Preparation

clone 1000 repos
language:java
sort:stars



parse (ANTLR)
and preprocess

Plain text source

```
p r i n t l n ( " H e l l o ▯ W o r l d ! " ) ;
```

Lexing instructions

```
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
```

Tokens

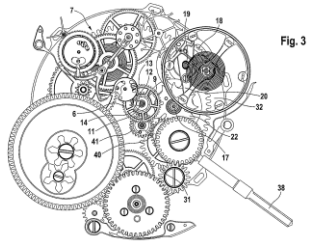
```
println ( "Hello,▯world" ) ;
```

Node type & AST depth annotations

```
Expression|12 Expression|13 Literal|17 Expression|13 Statement|11
```

Data Gathering and Preparation

clone 1000 repos
language:java
sort:stars



parse (ANTLR)
and preprocess

Plain text source

```
p r i n t l n ( " H e l l o , w o r l d ! " ) ;
```

Lexing instructions

```
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
```

Tokens

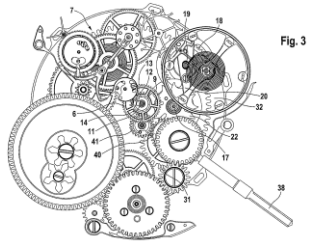
```
println ( "Hello, world" ) ;
```

Node type & AST depth annotations

```
Expression|12 Expression|13 Literal|17 Expression|13 Statement|11
```

Data Gathering and Preparation

clone 1000 repos
language:java
sort:stars



parse (ANTLR)
and preprocess

Plain text source

```
p r i n t l n ( " H e l l o ␣ W o r l d ! " ) ;
```

Lexing instructions

```
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
```

Tokens

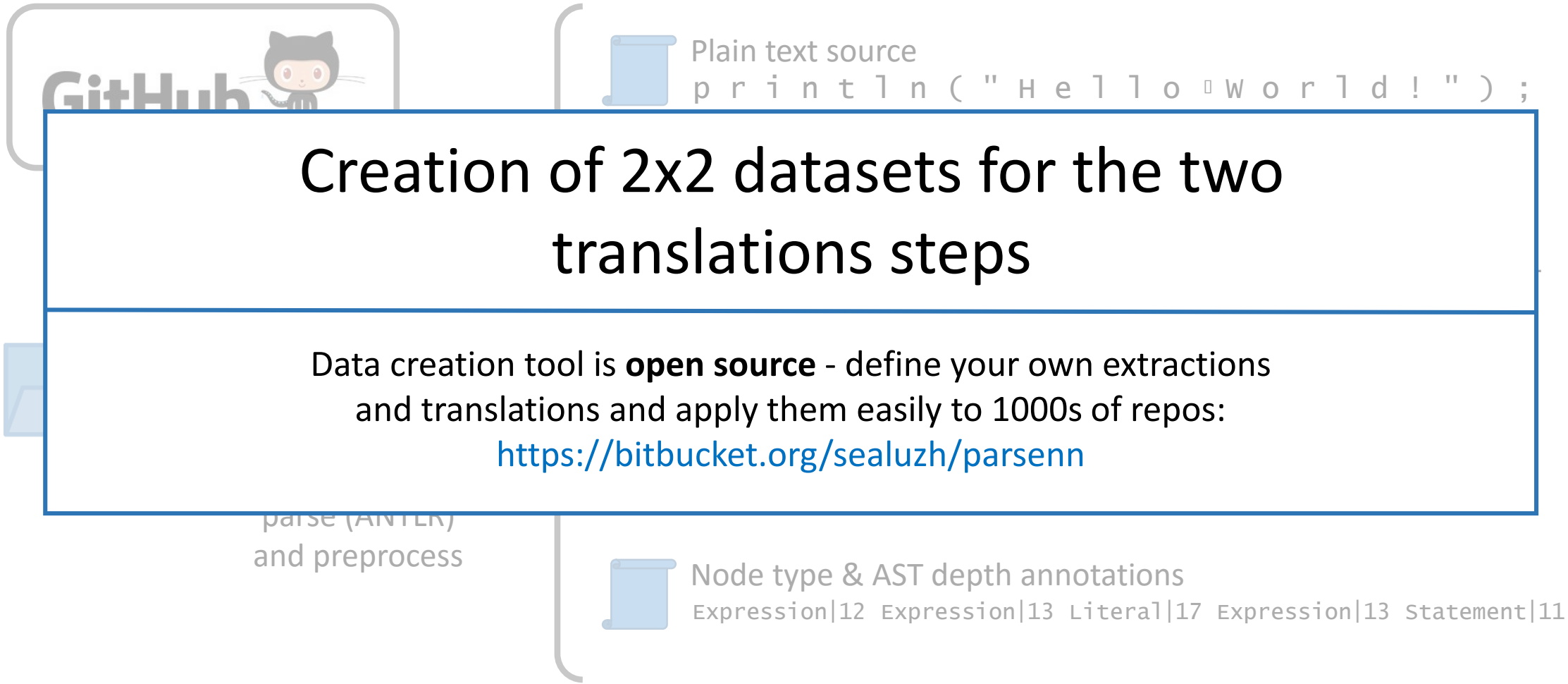
```
println ( "Hello, world" ) ;
```

Only contains AST node types correlating to literal tokens

Node type & AST depth annotations

```
Expression|12 Expression|13 Literal|17 Expression|13 Statement|11
```

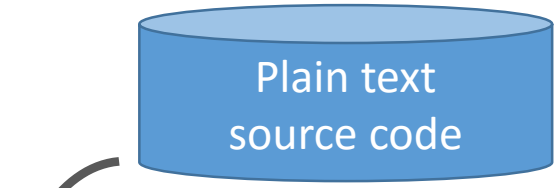
Data Gathering and Preparation



Creation of 2x2 datasets for the two translations steps

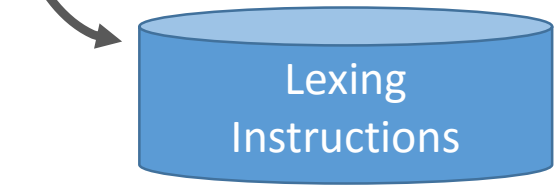
Data creation tool is **open source** - define your own extractions and translations and apply them easily to 1000s of repos:
https://bitbucket.org/sealuzh/parsenn

Results: Tokenization



Vocab size: 2185
Train: 25M samples (1.7Gb)
Validation: 2M samples (140Mb)

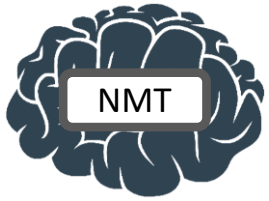
```
i m p o r t | a n d r o i d . g r a p h i c s . B i t m a p ;
i m p o r t | c o m . f a c e b o o k . c o m m o n . r e f e r e n c e s . R e s o u r c
p u b l i c | c l a s s | S i m p l e B i t m a p R e l e a s e r | i m p l e m e n t s | R
p r i v a t e | s t a t i c | S i m p l e B i t m a p R e l e a s e r | s I n s t a n c e ;
p u b l i c | s t a t i c | S i m p l e B i t m a p R e l e a s e r | g e t I n s t a n c e
i f | ( s I n s t a n c e | = = | n u l l ) | {
s I n s t a n c e | = | n e w | S i m p l e B i t m a p R e l e a s e r ( ) ;
}
```



Vocab size: 3
Train: 25M samples (1.7Gb)
Validation: 2M samples (140Mb)

```
0 0 0 0 0 1 | 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1
0 0 0 0 0 1 | 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0
0 0 0 0 0 1 | 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 1 | 0
0 0 0 0 0 0 1 | 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 1 | 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 1
0 1 | 1 0 0 0 0 0 0 0 0 1 | 0 1 | 0 0 0 1 1 | 1
0 0 0 0 0 0 0 0 1 | 1 | 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
1
```

Results: Tokenization



Bi-RNN
7 epochs
7 days
Perplexity: 1.11

Plain text source code

Vocab size: 2185
Train: 25M samples (1.7Gb)
Validation: 2M samples (140Mb)

```
i m p o r t | a n d r o i d . g r a p h i c s . B i t m a p ;
i m p o r t | c o m . f a c e b o o k . c o m m o n . r e f e r e n c e s . R e s o u r c
p u b l i c | c l a s s | S i m p l e B i t m a p R e l e a s e r | i m p l e m e n t s | R
p r i v a t e | s t a t i c | S i m p l e B i t m a p R e l e a s e r | s I n s t a n c e ;
p u b l i c | s t a t i c | S i m p l e B i t m a p R e l e a s e r | g e t I n s t a n c e
i f | ( s I n s t a n c e | = = | n u l l ) | {
s I n s t a n c e | = | n e w | S i m p l e B i t m a p R e l e a s e r ( ) ;
}
```

Lexing Instructions

Vocab size: 3
Train: 25M samples (1.7Gb)
Validation: 2M samples (140Mb)

```
0 0 0 0 0 1 | 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1
0 0 0 0 0 1 | 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 (
0 0 0 0 0 1 | 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 1 | 0 (
0 0 0 0 0 0 1 | 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 1 | 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 0 1
0 1 | 1 0 0 0 0 0 0 0 0 1 | 0 1 | 0 0 0 1 1 | 1
0 0 0 0 0 0 0 0 1 | 1 | 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
1
```

Results: Tokenization

What is **perplexity**?

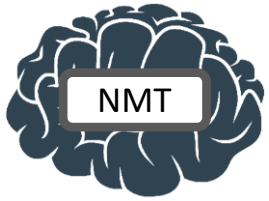
In the context of NMT:

Perplexity describes how "confused" a probability model is on a given test data set. A perfect model has perplexity 1.

Lower Perplexity is better

Meaning of perplexity value depends on **target vocab size**

Results: Tokenization



Bi-RNN
7 epochs
7 days
Perplexity: 1.11

Plain text source code

Vocab size: 2185
Train: 25M samples (1.7Gb)
Validation: 2M samples (140Mb)

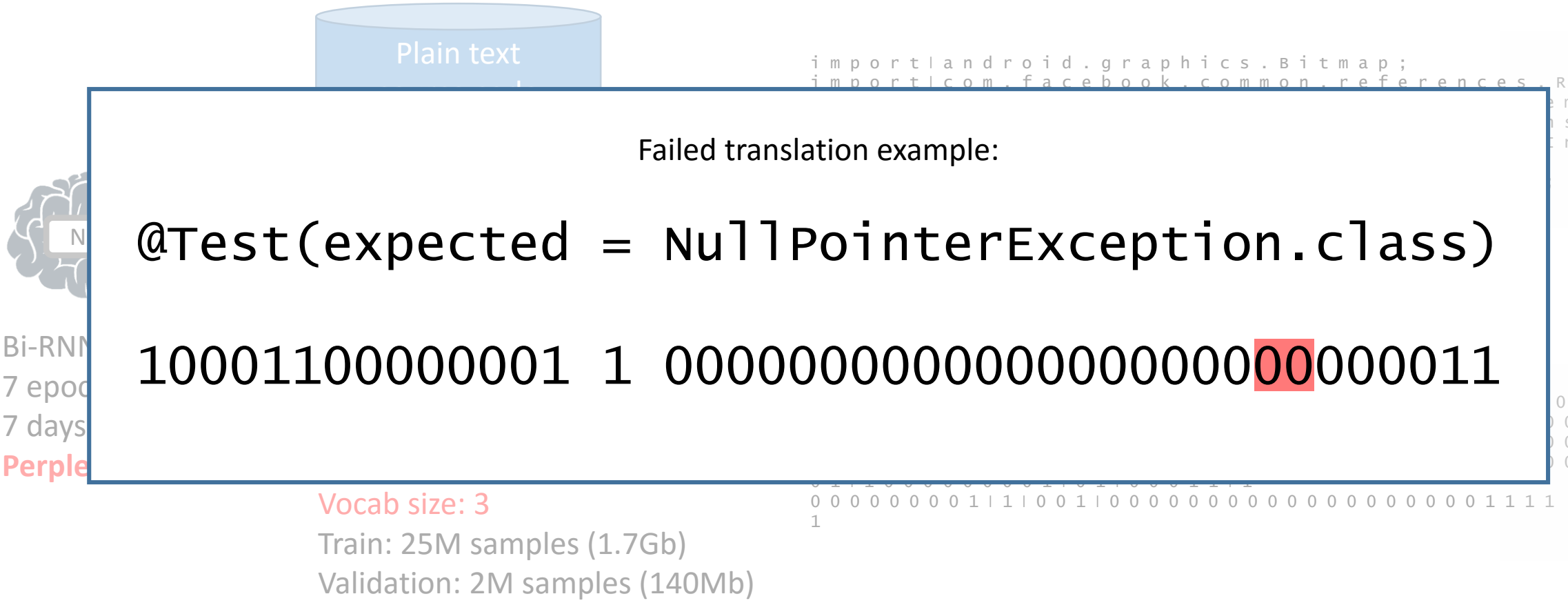
```
i m p o r t | a n d r o i d . g r a p h i c s . B i t m a p ;
i m p o r t | c o m . f a c e b o o k . c o m m o n . r e f e r e n c e s . R e s o u r c
p u b l i c | c l a s s | S i m p l e B i t m a p R e l e a s e r | i m p l e m e n t s | R
p r i v a t e | s t a t i c | S i m p l e B i t m a p R e l e a s e r | s I n s t a n c e ;
p u b l i c | s t a t i c | S i m p l e B i t m a p R e l e a s e r | g e t I n s t a n c e
i f | ( s I n s t a n c e | = = | n u l l ) | {
s I n s t a n c e | = | n e w | S i m p l e B i t m a p R e l e a s e r ( ) ;
}
```

Lexing Instructions

Vocab size: 3
Train: 25M samples (1.7Gb)
Validation: 2M samples (140Mb)

```
0 0 0 0 0 1 | 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1
0 0 0 0 0 1 | 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 (
0 0 0 0 0 1 | 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 1 | 0 (
0 0 0 0 0 0 1 | 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 1 | 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 | 0 0 0 0 0 0 0 0 0 0 1
0 1 | 1 0 0 0 0 0 0 0 0 1 | 0 1 | 0 0 0 1 1 | 1
0 0 0 0 0 0 0 0 1 | 1 | 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
1
```

Results: Tokenization

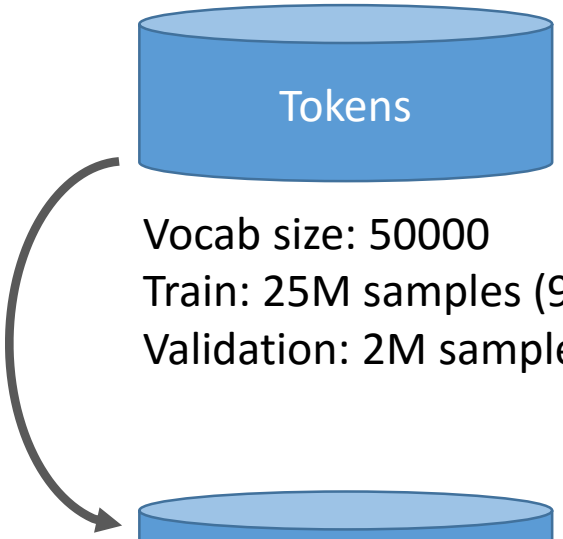


Failed translation example:

```
@Test(expected = NullPointerException.class)

10001100000001 1 00000000000000000000011
```

Results: Token Annotation



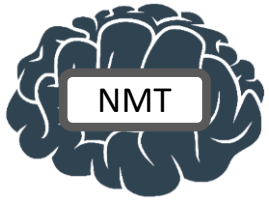
Vocab size: 50000
Train: 25M samples (904Mb)
Validation: 2M samples (76M)

```
import android . graphics . Bitmap ;
import com . facebook . common . references . ResourceRereleaser ;
public class SimpleBitmapReleaser implements ResourceReleaser < Bitmap > {
private static SimpleBitmapReleaser sInstance ;
public static SimpleBitmapReleaser getInstance ( ) {
if ( sInstance == null ) {
sInstance = new SimpleBitmapReleaser ( ) ;
}
```

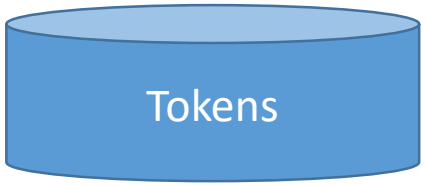
Vocab size: 87|4459
Train: 25M samples (3Gb)
Validation: 2M samples (251Mb)

```
ImportDeclaration|2 QualifiedName|3 QualifiedName|3 QualifiedName|3 QualifiedName|3 Qualif
ImportDeclaration|2 QualifiedName|3 QualifiedName|3 QualifiedName|3 QualifiedName|3 Qualif
ClassOrInterfaceModifier|3 ClassDeclaration|3 ClassDeclaration|3 ClassDeclaration|3 Classo
ClassOrInterfaceModifier|7 ClassOrInterfaceModifier|7 ClassOrInterfaceType|9 VariableDecla
ClassOrInterfaceModifier|7 ClassOrInterfaceModifier|7 ClassOrInterfaceType|9 MethodDeclara
IfStatement|12 ParExpression|13 Primary|16 Expression|14 Literal|17 ParExpression|13
Block|14
```

Results: Token Annotation

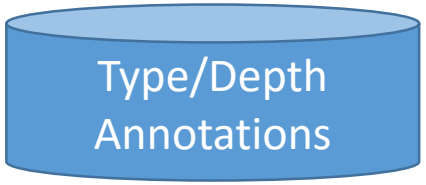


RNN
11 epochs
2 days
Perplexity: 1.28



Vocab size: 50000
Train: 25M samples (904Mb)
Validation: 2M samples (76M)

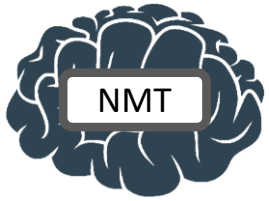
```
import android . graphics . Bitmap ;
import com . facebook . common . references . ResourceReleaser ;
public class SimpleBitmapReleaser implements ResourceReleaser < Bitmap > {
private static SimpleBitmapReleaser sInstance ;
public static SimpleBitmapReleaser getInstance ( ) {
if ( sInstance == null ) {
sInstance = new SimpleBitmapReleaser ( ) ;
}
```



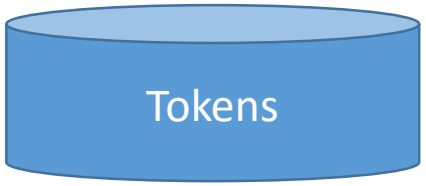
Vocab size: 87|4459
Train: 25M samples (3Gb)
Validation: 2M samples (251Mb)

```
ImportDeclaration|2 QualifiedName|3 QualifiedName|3 QualifiedName|3 QualifiedName|3 Qualif
ImportDeclaration|2 QualifiedName|3 QualifiedName|3 QualifiedName|3 QualifiedName|3 Qualif
ClassOrInterfaceModifier|3 ClassDeclaration|3 ClassDeclaration|3 ClassDeclaration|3 Classo
ClassOrInterfaceModifier|7 ClassOrInterfaceModifier|7 ClassOrInterfaceType|9 VariableDecla
ClassOrInterfaceModifier|7 ClassOrInterfaceModifier|7 ClassOrInterfaceType|9 MethodDeclara
IfStatement|12 ParExpression|13 Primary|16 Expression|14 Literal|17 ParExpression|13
Block|14
```

Results: Token Annotation

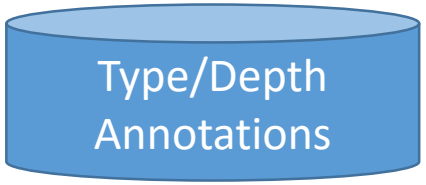


RNN
11 epochs
2 days
Perplexity: 1.28



Vocab size: 50000
Train: 25M samples (904Mb)
Validation: 2M samples (76M)

```
import android . graphics . Bitmap ;
import com . facebook . common . references . ResourceReleaser ;
public class SimpleBitmapReleaser implements ResourceReleaser < Bitmap > {
private static SimpleBitmapReleaser sInstance ;
public static SimpleBitmapReleaser getInstance ( ) {
if ( sInstance == null ) {
sInstance = new SimpleBitmapReleaser ( ) ;
}
```



Vocab size: 87|4459
Train: 25M samples (3Gb)
Validation: 2M samples (251Mb)

```
ImportDeclaration|2 QualifiedName|3 QualifiedName|3 QualifiedName|3 QualifiedName|3 Qualif
ImportDeclaration|2 QualifiedName|3 QualifiedName|3 QualifiedName|3 QualifiedName|3 Qualif
ClassOrInterfaceModifier|3 ClassDeclaration|3 ClassDeclaration|3 ClassDeclaration|3 Classo
ClassOrInterfaceModifier|7 ClassOrInterfaceModifier|7 ClassOrInterfaceType|9 VariableDecla
ClassOrInterfaceModifier|7 ClassOrInterfaceModifier|7 ClassOrInterfaceType|9 MethodDeclara
IfStatement|12 ParExpression|13 Primary|16 Expression|14 Literal|17 ParExpression|13
Block|14
```

Results: Token Annotation

A successful example:

List	<Throwable>	errors	=	TestHelper	.	trackPluginErrors	(	)	;
0001	1	000000001	1	000001	1	000000001	1	000000000000000	1	1	1

[ClassOrInterfaceType|14] [TypeArguments|15] [ClassOrInterfaceType|18] [TypeArguments|15] [VariableDeclaratorId|15] [VariableDeclarator|14]
[Primary|19] [Expression|17] [Expression|17] [Expression|16] [Expression|16] [LocalVariableDeclarationStatement|11]

Vocab size: 87|4459
Train: 25M samples (3Gb)
Validation: 2M samples (251Mb)

IfStatement|12 ParExpression|13 Primary|16 Expression|14 Literal|17 ParExpression|13 Block|14

Take-home messages:

- NN can learn to "read" code (tokens / syntactic elements)

  → What else could we teach? Type resolution? Calls & attribute access? Inheritance?

  → Could we follow the "human path" of learning to program to teach an AI?

- *"If only we had good data"*

  → Bug reports, commit messages etc. are *still* unstructured. This needs to change if we want to leverage deep learning in SE and PC.

ICPC 2017

EARLY RESEARCH ACHIEVEMENTS

THANKS FOR LISTENING

Data creation tool: **t.uzh.ch/Hb**

Paper: **t.uzh.ch/Hc**

University of Zurich[UZH]

s.e.a.l. software evolution & architecture lab

Carol V. Alexandru, Sebastiano Panichella, Harald C. Gall
{alexandru,panichella,gall}@ifi.uzh.ch

23. May 2017