Analyzing Software using Deep Learning

Sequence-to-Sequence Networks and their Applications

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Summer 2022

Overview

- **Sequence-to-sequence networks**
- **API usage sequences for natural language queries**
  Based on ”Deep API learning” by Gu et al., 2016
- **Interpreting Python programs**
  Based on ”Learning to execute” by Zaremba and Sutskever, 2014

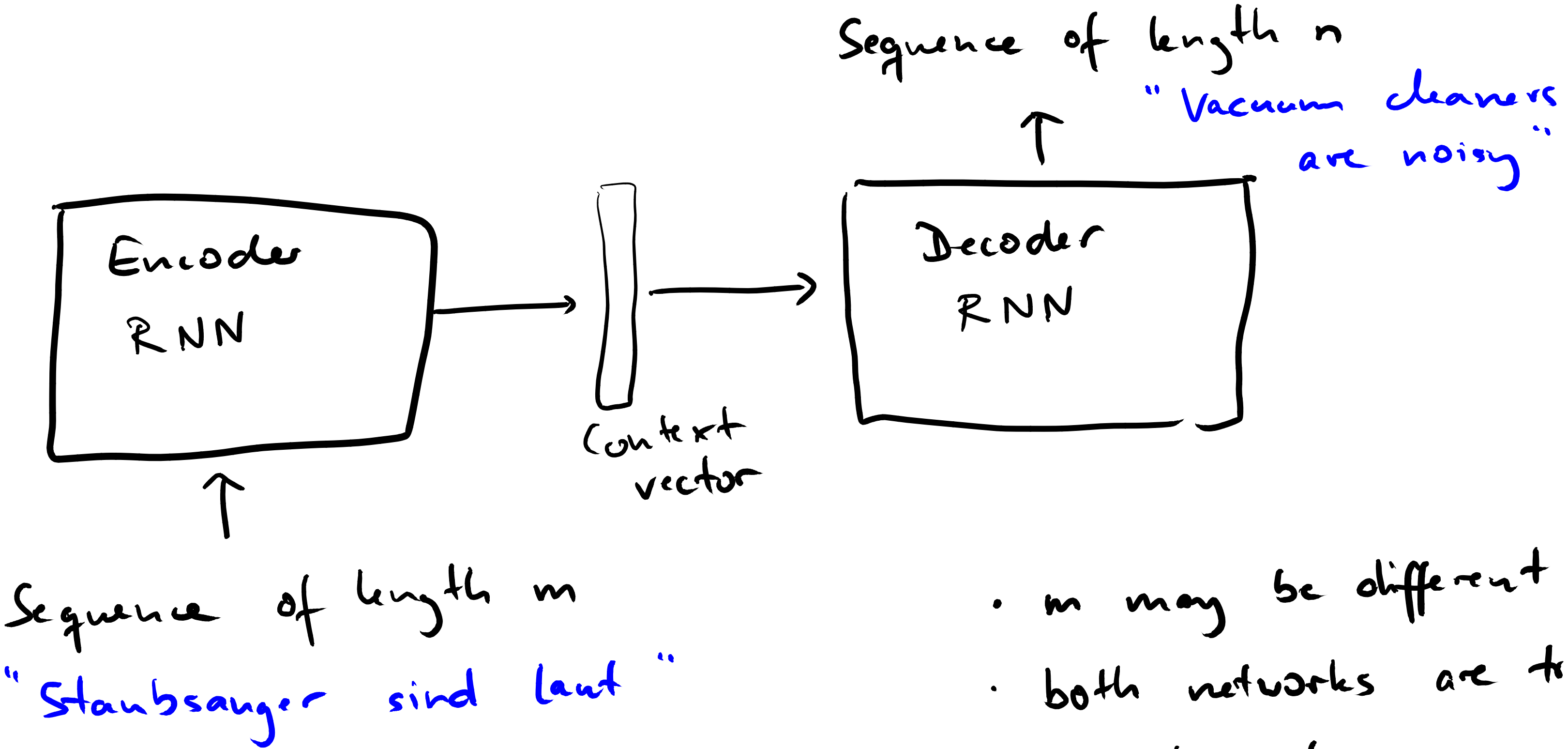
Sequence-to-Sequence

Goal: Translate sequence of items into another sequence of items

Various applications

- Translation between natural languages
- Generate image captions
- Summarize videos into text
- Answer natural language questions

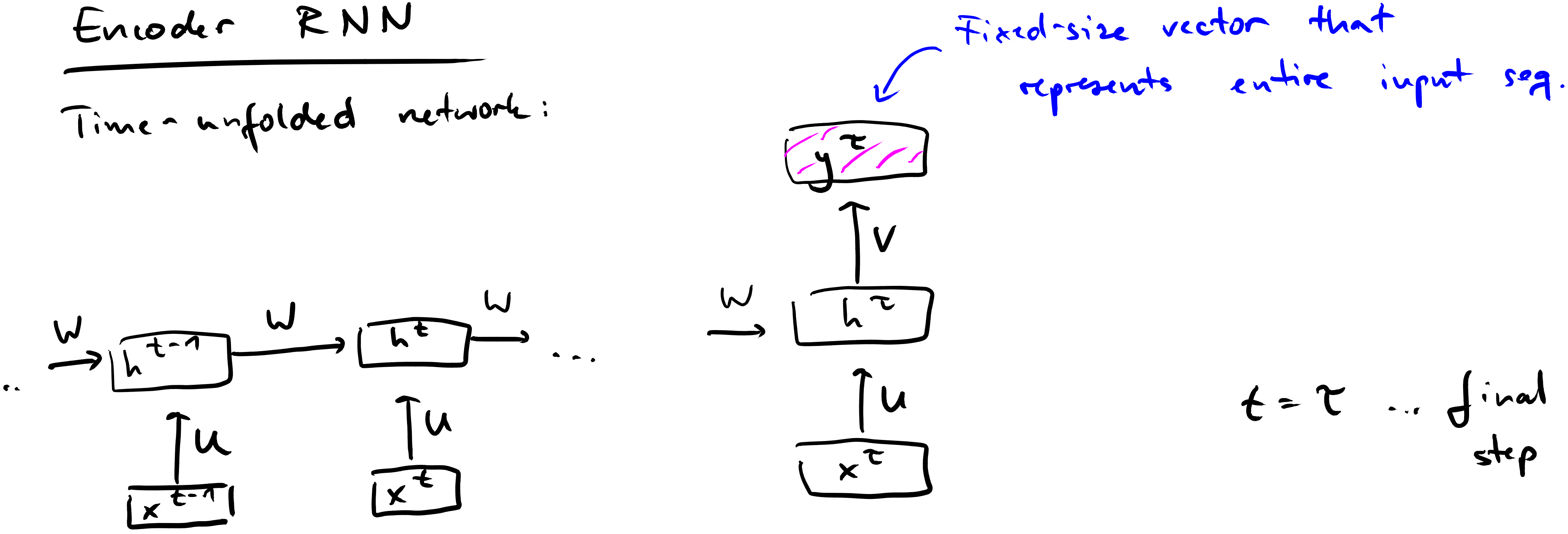
Overview of Sequence-to-Sequence Architecture



- m may be different from n
- both networks are trained jointly
- context vector: summary of input suitable to generate output

Encoder RNN

Time-unfolded network:



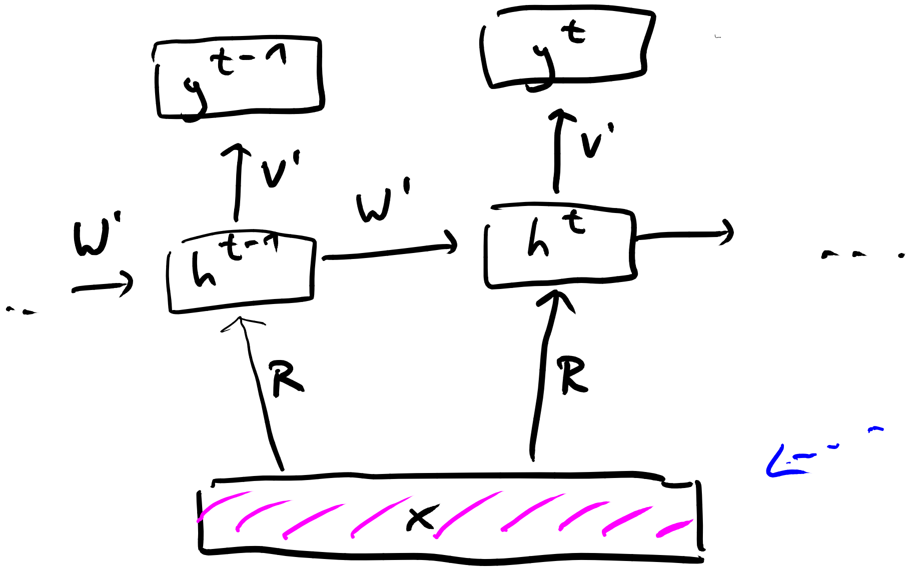
$t = \tau$ ... final step

$$h^t = \tanh(W \cdot h^{t-1} + U \cdot x^t + b)$$

or any other activation fct.

$$y^\tau = V \cdot h^\tau + c$$

Decoder RNN



Fixed-size used to generate entire output sequence

$$h^t = \tanh\left(W' \cdot h^{t-1} + R \cdot x + b'\right)$$

$$y^t = \mathrm{softmax}\left(V' \cdot h^t + c'\right)$$

Training

Training data: N pairs of sequences (x_i, y_i) for $i = 1, \dots, N$

↳ End of sequence marked with <EOS>

Example:

x_1 = Staubsauger, sind, laut, <EOS>

y_1 = Vacuum, cleaners, are, noisy, <EOS>

Goal of training:

Minimize $\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} - \log \Pr(y_{it} \mid x_i)$.

where T .. length of output sequences

$\Pr(y_{it} \mid x_i)$ ... prob. of word y_{it} given input seq. x_i

Translation

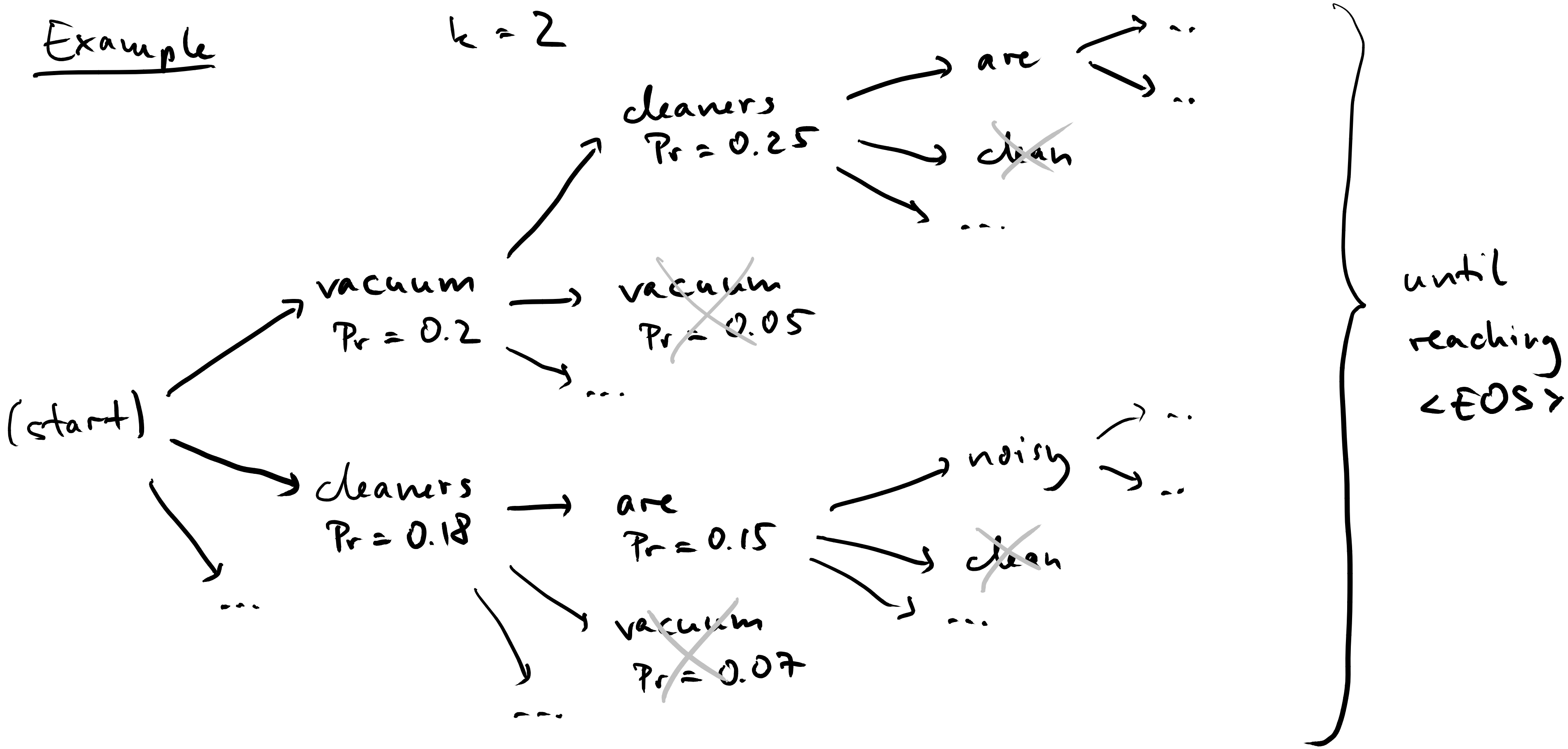
For many applications, want k most likely translations

Use left-to-right beam search

- For every word, consider k most likely alternatives
- Extend partial sentence in k ways
- After each time step, keep only k most likely partial sequences

Example

$k = 2$


(start)
vacuum
Pr = 0.2
cleaners
Pr = 0.18
cleaners
Pr = 0.25
vacuum
Pr = 0.05
are
Pr = 0.15
vacuum
Pr = 0.07
are
clean
noisy
clean
until reaching <EOS>

Quiz

Which of following sentences is correct (multiple sentences may be correct)?

- The context vector is a potential bottleneck that may prevent the network from effective learning.
- The length of the input sequence must be the same across all instances of the training set.
- The length of the output sequence must be the same across all instances of the training set.
- Each instance in the training set must contain two sequences (input and output).

Quiz

Which of following sentences is correct (multiple sentences may be correct)?

- The context vector is a potential bottleneck that may prevent the network from effective learning.
- The length of the input sequence must be the same across all instances of the training set.
- The length of the output sequence must be the same across all instances of the training set.
- Each instance in the training set must contain two sequences (input and output).

Overview

- **Sequence-to-sequence networks**
- **API usage sequences for natural language queries**

  Based on ”Deep API learning” by Gu et al., 2016
- **Interpreting Python programs**

  Based on ”Learning to execute” by Zaremba and Sutskever, 2014

Motivation

APIs are difficult to use

- Which methods to call?
- In what order to call them?

Developers ask questions, e.g., on stackoverflow.com

- Human effort required to answer them

Goal: Automatically suggest API usages based on natural language query

Idea

Formulate the problem as a translation problem

- Input: Sequence of natural language words
- Output: Sequence of API method calls
- Train and query sequence-to-sequence neural network

Example

Natural language query:

”match regular expressions”

Sequence of API calls expected as (possible) answer:

Pattern.compile, Pattern.matcher, Matcher.group

Training Data

- Analyze 443.000 Java projects from GitHub
- Focus on JDK = APIs of Java standard library
- Extract pairs of annotation and call sequence
- About 7 million extracted pairs
- Use 10.000 for testing and others for training

Example

```
/***
* Copies bytes from a large (over 2GB) InputStream to an
* OutputStream. This method uses the provided buffer, so
* there is no need to use a BufferedInputStream.
* @param input the InputStream to read from
* . . .
*/
public static long copyLarge(final InputStream input,
final OutputStream output, final byte[] buffer)
      throws IOException {
  long count = 0;
  int n;
  while (EOF != (n = input.read(buffer))) {
    output.write(buffer, 0, n);
    count += n;
  }
  return count;
}
```

Example

```
/***
* Copies bytes from a large (over 2GB) InputStream to an
* OutputStream. This method uses the provided buffer, so
* there is no need
* @param input the
* . . .
*/
public static long
final OutputStream
      throws IOExce
   long count = 0;
   int n;
   while (EOF != (n = input.read(buffer))) {
     output.write(buffer, 0, n);
     count += n;
   }
   return count;
}
```

Annotation:

"copies bytes from a large inputstream to an outputstream"

Call sequence:

InputStream.read , OutputStream.write

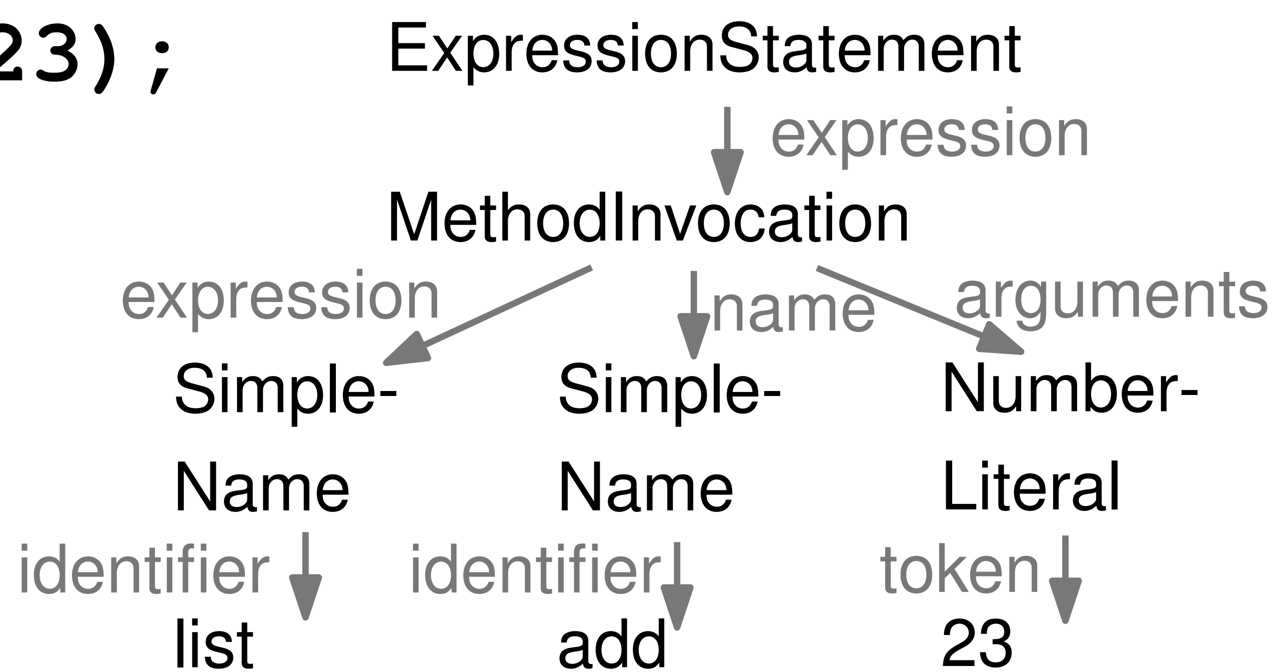
Extracting Annotations

- Extract JavaDoc of each method
- Extract first sentence
- Ignore methods without JavaDoc
- Ignore annotations with ”irregular” comments, e.g., `TODO: ...`

Extracting Call Sequences

- Goal: Lightweight analysis that scales to millions of code files
- Static, AST-based analysis with type bindings
- Example:

```
list.add(23);
```



AST-based Extraction (1)

- Constructor call:

  `new C()` $\rightarrow$ C.new (if `C` is JDK class)

- Method call:

  `obj.m()` $\rightarrow$ C.m (if type of `obj` is JDK class)

- Call expressions as arguments:

  `o1.m1(o2.m2())` $\rightarrow$ C2.m2, C1.m1

AST-based Extraction (2)

- Sequence of statements:

  `o1.m1(); o2.m2();` → C1.m1, C2.m2

- Conditionals:

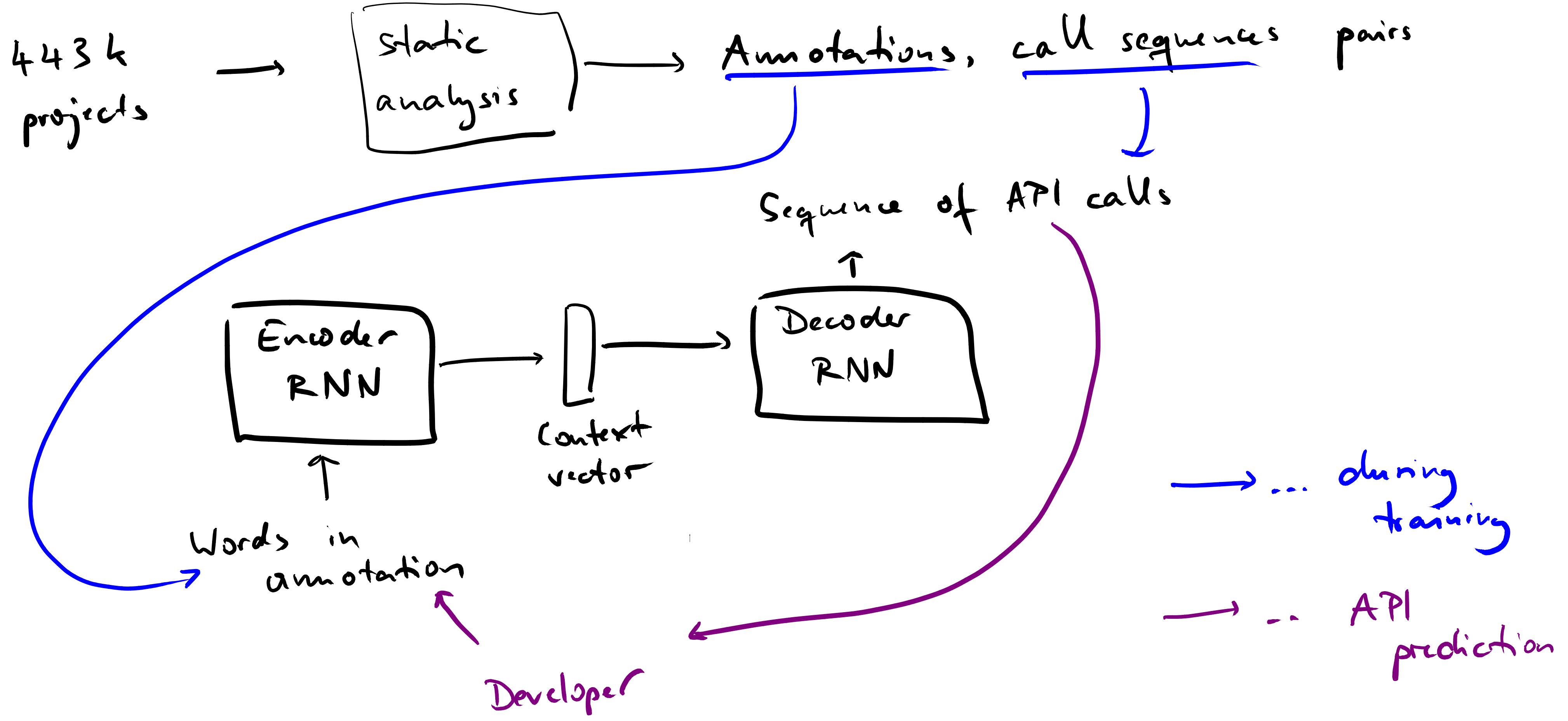
```
if(o1.m1()) {
    o2.m2();
} else {
    o3.m3();
}
```
→ C1.m1, C2.m2, C3.m3

- Loops:

  `while(o1.m1()) { o2.m2(); }`

  → C1.m1, C2.m2

Putting Everything Together



Examples

- ”generate md5 hash code”

  $\rightsquigarrow$ MessageDigest.getInstance, MessageDigest.update, MessageDigest.digest

- ”convert int to string”

  $\rightsquigarrow$ Integer.toString

- ”get files in folder”

  $\rightsquigarrow$ File.new, File.list, File.new, File.isDirectory

Overview

- **Sequence-to-sequence networks**
- **API usage sequences for natural language queries**
  Based on ”Deep API learning” by Gu et al., 2016
- **Interpreting Python programs**
  Based on ”Learning to execute” by Zaremba and Sutskever, 2014

Motivation

In principle, neural networks can express arbitrary computations

Can they interpret a program?

- Real-world interpreters are complex pieces of software
- Non-trivial task

Idea

Formulate as sequence-to-sequence translation problem

- Input: Sequence of characters of the source code
- Output: Sequence of characters of the program output
- Here: Restricted set of programs
  - Can evaluate with single left-to-right pass using constant memory

Example

Program:

```
j=8584
for x in range(8):
  j+=920
b=(1500+j)
print((b+7567))
```

Expected result:

```
25011
```

Another Example

Program:

```
vqppkn
sqdvfljmnc
y2vxdddsepnimcbvubkomhrpliibtwztbljipcc
```

Expected result:

```
hkhpg
```

Characters are obfuscated to illustrate difficulty faced by neural network

Training Data

Inputs:

- Automatically generated Python programs
  - Addition, subtraction, multiplication
  - Variable assignments
  - If statements
  - For loops, but not nested loops
  - Ends with `print` statement

Outputs:

- Behavior of traditional Python interpreter

Results

- Prediction accuracy between 36% and 84%
- Depends on size and complexity of programs
- Example of inaccurate prediction:

```
e=6653
for x in range(14):e+=6311
print(e)
```

  - Predicted output: `94103`
  - Actual output: `95007`

Summary

Sequence-to-sequence networks

- Two jointly trained RNNs combined through context vector
- Translation with arbitrary length of sequences

Applications

- Predict API call sequences for natural language queries
- Interpret programs and predict their output