Analyzing Software using Deep Learning

Lecture 3: Sequence-to-Sequence Networks and their Applications

Prof. Dr. Michael Pradel
Software Lab, TU Darmstadt
Plan for Today (Part 2)

- 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

Sequence of length \( n \) are noisy

Encoder RNN

\[
\text{context vector}
\]

Decoder RNN

Sequence of length \( m \)

- \( m \) may be different from \( n \)
- both networks trained jointly
- context vector summarizes the input sequence in a way suitable to generate the output sequence

Staubsauger sind laut
Encoder RNN

Time-unfolded network:

\[ h^t = \tanh \left( W h^{t-1} + U x^t + b \right) \]

\[ y^t = V h^t + c \]

Fixed-size vector that represents entire input sequence

\( t = \tau \) final time step

or another activation function
Decoder RNN

Time-unfolded network

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

\[ y^t = \text{softmax} \left( V' h^t + c \right) \]

Fixed-size vector used to generate entire output sequence
Seg-to-seq Architecture

Encoder

Decoder
Training

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

End of sequence marked with \(<\text{EOS}>\)

Example:
\[
x_i = \text{Staubsauger, sird, laut}, <\text{EOS}>
\]
\[
y_i = \text{Vacuum, cleaners, are, noisy}, <\text{EOS}>
\]

Goal of training:

Minimize \[
\frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} - \log \Pr(y_{it} | x_i)
\]

where \(T_i\) = length of each output sequence.

\(\Pr(y_{it} | x_i)\) = probability of word \(y_{it}\) given the input sequence \(x_i\).
Translation

For many applications, want k most likely translations.

1. 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 sentences.

Example: k = 2

(start)

- Vacuum
  - Cleaners
    - are
  - Clean

- Cleaners
  - Pr = 0.18
  - are
    - Pr = 0.15
    - noisy
    - clean

- Vacuum
  - Pr = 0.25
  - are
    - Pr = 0.05
    - vacuum
    - Pr = 0.05

...
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).
Plan for Today (Part 2)

- 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
/***/
* 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;
}
/***
* 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;
}
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:

```java
list.add(23);  
```
AST-based Extraction (1)

- **Constructor call:**
  
  ```java
tenew C() → C.m  (if C is JDK class)
```

- **Method call:**
  
  ```java
obj.m() → C.m  (if type of obj is JDK class)
```

- **Call expressions as arguments:**
  
  ```java
o1.m1(o2.m2()) → C2.m2, C1.m1
```
AST-based Extraction (2)

- **Sequence of statements:**
  
  \[ o1.m1(); \ o2.m2(); \rightarrow C1.m1, C2.m2 \]

- **Conditionals:**
  
  \[
  \text{if}(o1.m1()) \ {
  \begin{array}{l}
  o2.m2(); \\
  \text{else} \\
  o3.m3();
  \end{array}
  \} \rightarrow C1.m1, C2.m2, C3.m3
  \]

- **Loops:**
  
  \[
  \text{while}(o1.m1()) \ {\ o2.m2(); } \} \rightarrow C1.m1, C2.m2
  \]
Putting Everything Together

443k projects — Static analysis

Annotation, call sequence pairs

Sequence of API calls

Encoder RNN

Words in annotations

context vector

Decoder RNN

Developer ✌️

during training

API predictions
Examples

■ "generate md5 hash code"
  ~ MessageDigest.getInstance, MessageDigest.update, MessageDigest.digest

■ "convert int to string"
  ~ Integer.toString

■ "get files in folder"
  ~ File.new, File.list, File.new, File.isDirectory
Plan for Today (Part 2)

- 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:

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

Expected result:

25011
Another Example

Program:

vqppkn
sqdvfljmnc
y2vxdddsepnimcbvubkontpliibtwztbljipcc

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:

```python
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