Analyzing Software using Deep Learning

RNN-based Code Completion and Repair

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Summer 2022

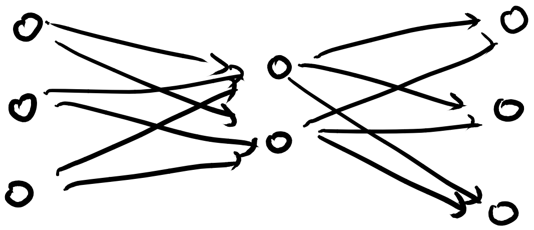
Overview

- **Recurrent neural networks (RNNs)**
- **Code completion with statistical language models**

  Based on PLDI 2014 paper by Raychev et al.
- **Repair of syntax errors**

  Based on ”Automated correction for syntax errors in programming assignments using recurrent neural networks” by Bhatia & Singh, 2016

From Neurons to Layers

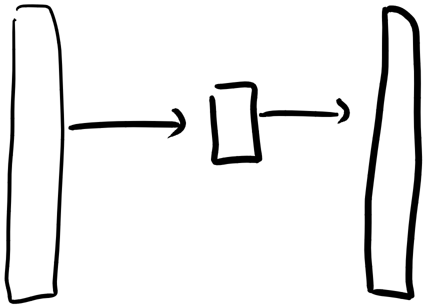


For every neuron:

$$output = f(w \cdot x + b)$$

x, f, b ... scalar, e.g., in $\mathbb{R}$

w ... vector, e.g. $\mathbb{R}^n$



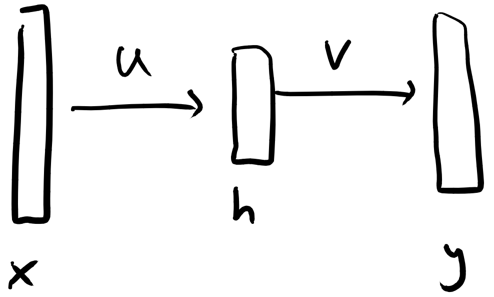
For each layer:

$$output = f(W \cdot x + b)$$

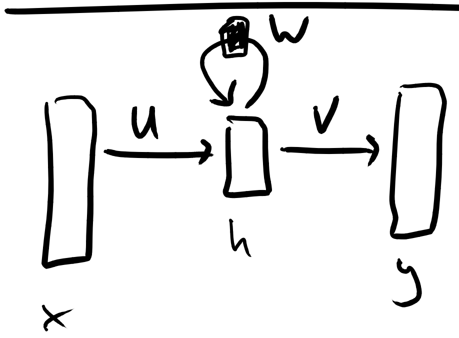
x, f, b ... vectors, e.g., $\mathbb{R}^n$

W ... matrix, e.g., $\mathbb{R}^{m \times n}$

Feedforward networks



Recurrent networks



x, h, y ... input layer, hidden layer, output layer

U, V, W .. weight matrices

$\rightarrow$ ... functions

$\xrightarrow{\boxed{}}$ ... function with <u>delay of single time step</u>

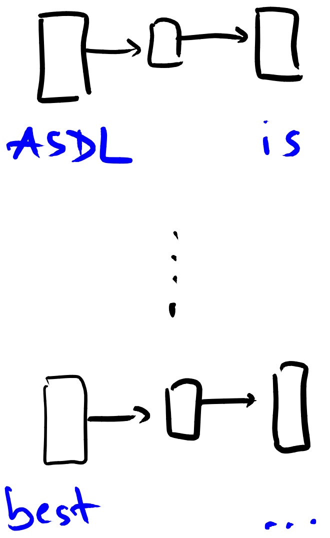
→ useful f. representing sequences of inputs & outputs

→ store information about previous inputs

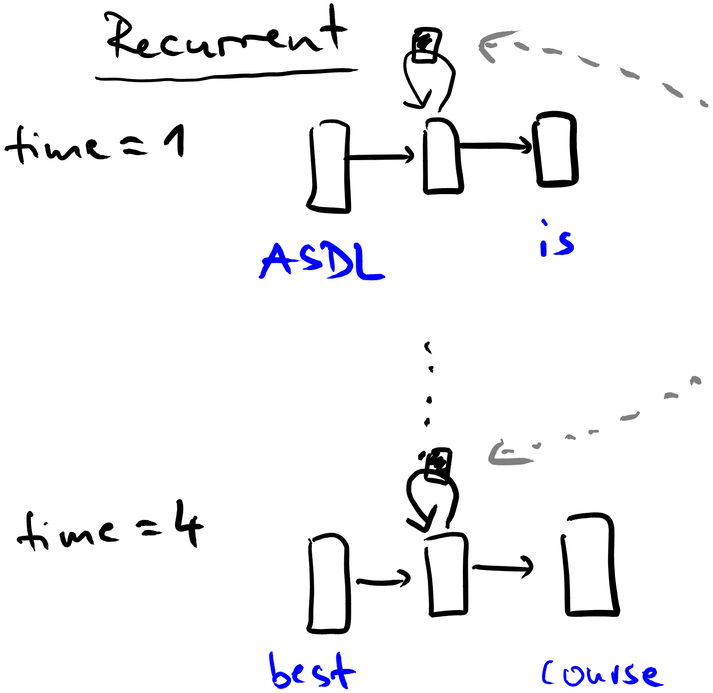
Example: Predict next word in sentence

ASDL is the best ... (course)

Feedforward:

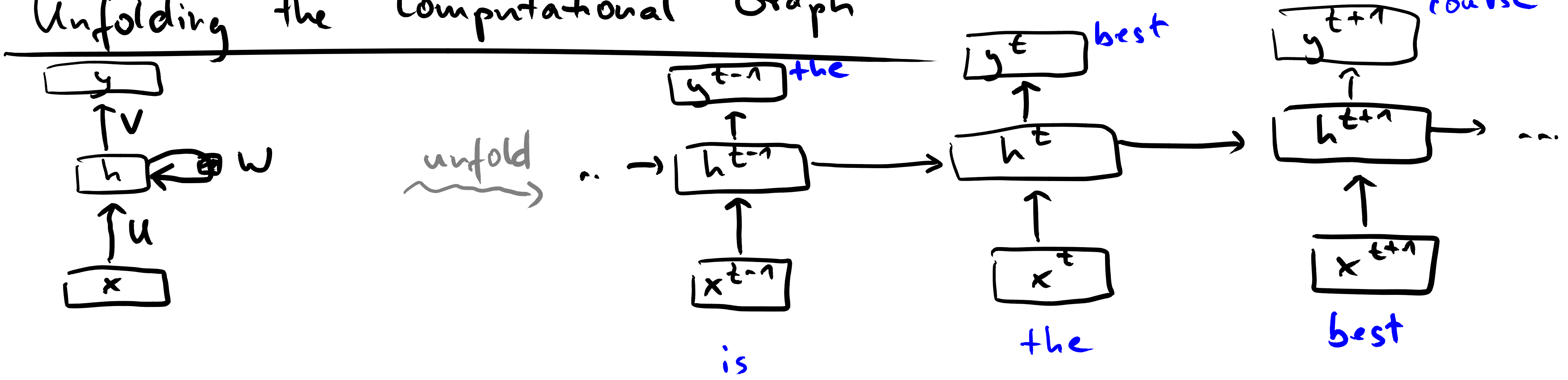


Recurrent



Recurrent connection remembers the begining of the sentence

Unfolding the Computational Graph



$h^t = f(h^{t-1}, x^t)$ .... e.g. $h^t = \tanh(W \cdot h^{t-1} + U \cdot x^t + b)$

$y^t = f(h^t)$ ... e.g. $y^t = \text{softmax}(V \cdot h^t + c)$

Softmax Function

- Goal: Interpret output vector as a probability distribution
- ”Squashes” vector of k values $\in \mathbb{R}$ into vector of k values $\in (0, 1)$ that sum up to 1
- Definition:

$$\sigma(y)_j = \frac{e^{y_j}}{\sum_i^k e^{y_i}} \text{ for } j = 1, .., k$$

- Example:

$$\sigma([1, 2, 3, 4, 1, 2, 3]) = [0.024, 0.064, 0.175, 0.475, 0.024, 0.064, 0.175]$$

Quiz

Which of the following vectors may be the output of the softmax function?

1.) $y = [0.0, 0.0, 0.0, 0.0]$

2.) $y = [0.0, 0.25, 0.25, 0.5]$

3.) $y = [0.0, 1.0, 0.0, 0.0]$

4.) $y = [0.1, 0.1, 0.2, 0.3]$

Quiz

Which of the following vectors may be the output of the softmax function?

1.) ~~$y = [0.0, 0.0, 0.0, 0.0]$~~ **sum is not 1**

2.) $y = [0.0, 0.25, 0.25, 0.5]$

3.) $y = [0.0, 1.0, 0.0, 0.0]$

4.) ~~$y = [0.1, 0.1, 0.2, 0.3]$~~ **sum is not 1**

Note: Mathematically, 0 and 1 cannot occur. In practice, they may occur due to rounding of floating point numbers.

Applications of RNNs

Useful for tasks where the input (and maybe also the output) is a sequence

For example, predictions about ...

- Code (as a sequence of code tokens)
- Comments (as a sequence of words)
- Runtime trace (as a sequence of events)
- Log files (as a sequence of tokens/words)

Overview

- **Recurrent neural networks (RNNs)**
- **Code completion with statistical language models**

  Based on PLDI 2014 paper by Raychev et al.
- **Repair of syntax errors**

  Based on ”Automated correction for syntax errors in programming assignments using recurrent neural networks” by Bhatia & Singh, 2016

Code Completion

- Given: Partial program with one or more holes
- Goal: Find suitable code to fill into the holes
- Basic variants in most IDEs
- Here: Fill holes with sequences of method calls
  - Which methods to call
  - Which arguments to pass

Example

```
SmsManager smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
  ArrayList<String> msgList =
      smsMgr.divideMsg(message);
  // hole H1
} else {
  // hole H2
}
```

Statistical Language Model

- Dictionary of words
- Sentences : sequences of words
- Model : Probability distrib. over all possible sentences

Example: English

$\Pr(\text{"hello world"}) > \Pr(\text{"world hello"})$

- Most basic model : Predict next word based on all previous words

$$\Pr(s) = \prod_{i=1}^{m} \Pr(w_i \mid h_{i-1}) \qquad \text{where} \quad s = w_1 \cdot \ldots \cdot w_m$$

$$h_i = w_1 \cdot \ldots \cdot w_i$$

Model-based Code Completion

- Program code $\approx$ sentences in a language
- Code completion $\approx$ Finding the most likely completion of the current sentence

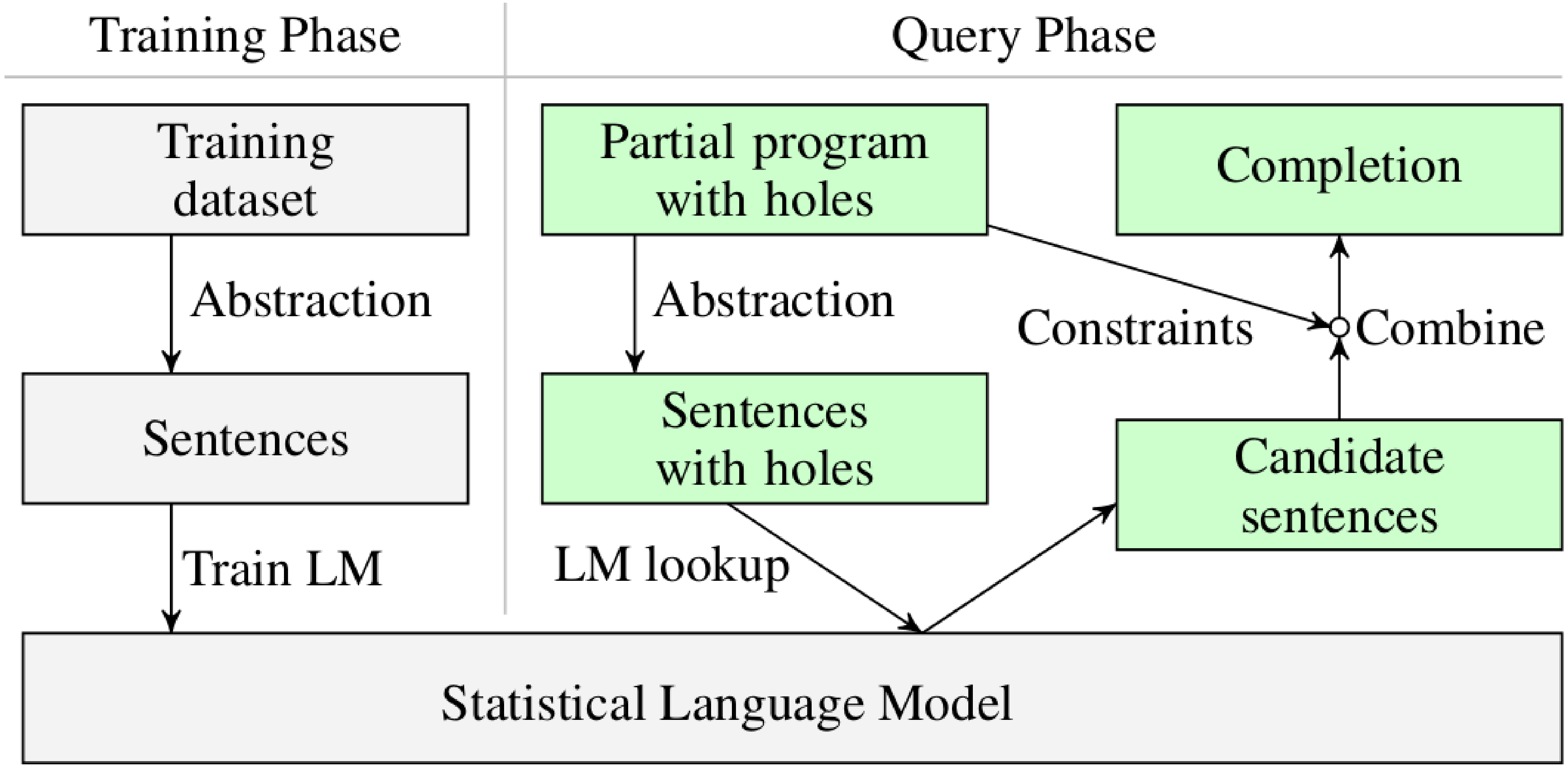
Model-based Code Completion

- **Program code ≈ sentences in a language**
- **Code completion ≈ Finding the most likely completion of the current sentence**

Challenges

- How to abstract code into sentences?
- What kind of language model to use?
- How to efficiently predict a completion

Overview of SLANG Approach



From ”Code Completion with Statistical Language Models”
by Raychev et al., 2014

n-gram Language Model

Pb. with "all history" model: Training data may not contain anything about h_i

Idea: Next word depends on n-1 previous words

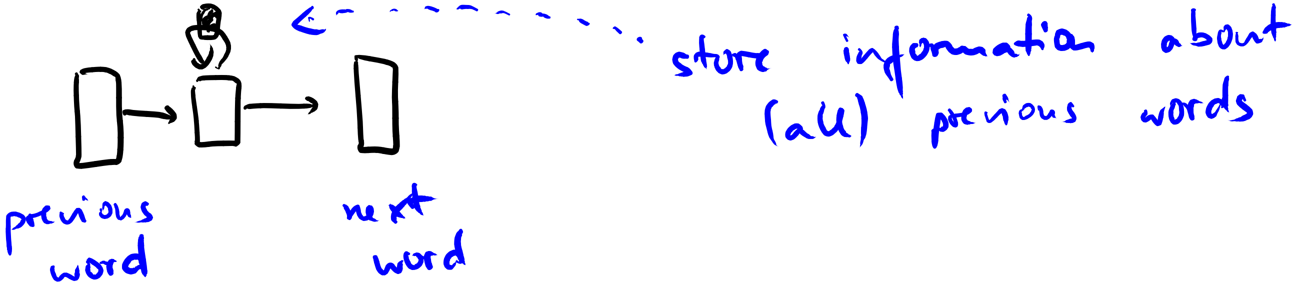
$$Pr(s) = \prod_{i=1}^{m} Pr(w_i \mid w_{i-(n-1)} \cdot \ldots \cdot w_{i-1})$$

Example: $Pr(to \cdot be \cdot or \cdot not \cdot to \cdot be)$ $n=3$

$$= Pr(to \cdot \varepsilon) \cdot Pr(be \mid to) \cdot Pr(or \mid to \cdot be) \cdot \ldots \cdot Pr(be \mid not \cdot to)$$

Probab. of n-grams: Estimated from corpus of training examples

RNN-based model



Encode as vector : One-hot encoding

↳ Length = size of vocabulary

↳ all values are zero , except one

Example: | 0 | 0 | 0 | 0 | 1 | 0 | . . . . | 0 |

↑

Sequences of Method Calls

Abstracting code into sentences

- Method call ≈ word
- Sequence of method calls ≈ sentence
- Separate sequences for each object
- Objects can occur in call as base object, argument, or return value

Option 1: Dynamic Analysis

Execute program and observe each method call

Advantage:

- Precise results

Disadvantage:

- Only analyzes executed code

```
if (getInput() > 5) {  // Suppose always taken
  obj.foo();           // in analyzed execution
} else {
  obj.bar();  // Never gets analyzed
}
```

Option 2: Static Analysis

Reason about execution without executing the code

Advantage:

- Can consider all execution paths

Disadvantage:

- Need to abstract and approximate actual execution

```
if (getInput() > 5) {
  a.foo();  // Does this call ever get executed?
}
b.bar();    // May a and b point to the same object?
```

Static Analysis of Call Sequences

SLANG approach: Static analysis

- Bound the number of analyzed loop iterations
- On control flow joins, take union of possible execution sequences
- Points-to analysis to reason about references to objects

Example

```
SmsManager smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
  ArrayList<String> msgList =
      smsMgr.divideMsg(message);
} else {}
```

Example

```
SmsManager smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
  ArrayList<String> msgList =
      smsMgr.divideMsg(message);
} else {}
```

5 sequences:

Object	Calls
smsMgr	(getDefault, ret)
smsMgr	(getDefault, ret) · (divideMsg, 0)
message	(length, 0)
message	(length, 0) · (divideMsg, 1)
msgList	(divideMsg, ret)

Training Phase

- Training data used for paper:

  3 million methods from various Android projects
- Extract sentences via static analysis
- Train statistical language model
  - Both n-gram and RNN model

Query Phase

- Given: Method with holes
- For each hole:
    - Consider all possible completions of the partial call sequence
    - Query language model to obtain probability
        - Average of n-gram and RNN models
- Return completed code that maximizes overall probability

Example

```
SmsManager smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
  ArrayList<String> msgList =
        smsMgr.divideMsg(message);
  // hole H1
} else {
  // hole H2
}
```

Example

```
SmsManager smsMgr = SmsManager.getDefault();
int length = message.length();
if (length > MAX_SMS_MESSAGE_LENGTH) {
  ArrayList<String> msgList =
      smsMgr.divideMsg(message);
  smsMgr.sendMultipartTextMessage(..., msgList, ...);
} else {
  smsMgr.sendTextMessage(..., message, ...);
}
```

Scalability Tricks

Search space of possible completions:
Too large to explore in reasonable time

Refinements to reduce space

- Users may provide hints
  - How many calls to insert
  - Which objects to use
- Replace infrequent words with ”unknown”
- Obtain candidate calls using bi-gram model
- Query language model only for candidates

Overview

- **Recurrent neural networks (RNNs)**
- **Code completion with statistical language models**

  Based on PLDI 2014 paper by Raychev et al.
- **Repair of syntax errors**

  Based on ”Automated correction for syntax errors in programming assignments using recurrent neural networks” by Bhatia & Singh, 2016

Motivation

- Given: Program with syntax error
- Goal: Find a fix that removes syntax error
- Possible application context: MOOCs with automated feedback on programming tasks

Example (1)

```
def recPower (base , exp):
  if exp <= 0:
    return 1
  return base * recPower (base , exp - 1
```

Example (1)

```
def recPower (base , exp):
  if exp <= 0:
    return 1
  return base * recPower (base , exp - 1)
```

Example (2)

```
def recurPower (base , exp):
  if exp == 0:
    return = exp + 1
  else:
    return (base * recurPower (base ,exp - 1))
```

Example (2)

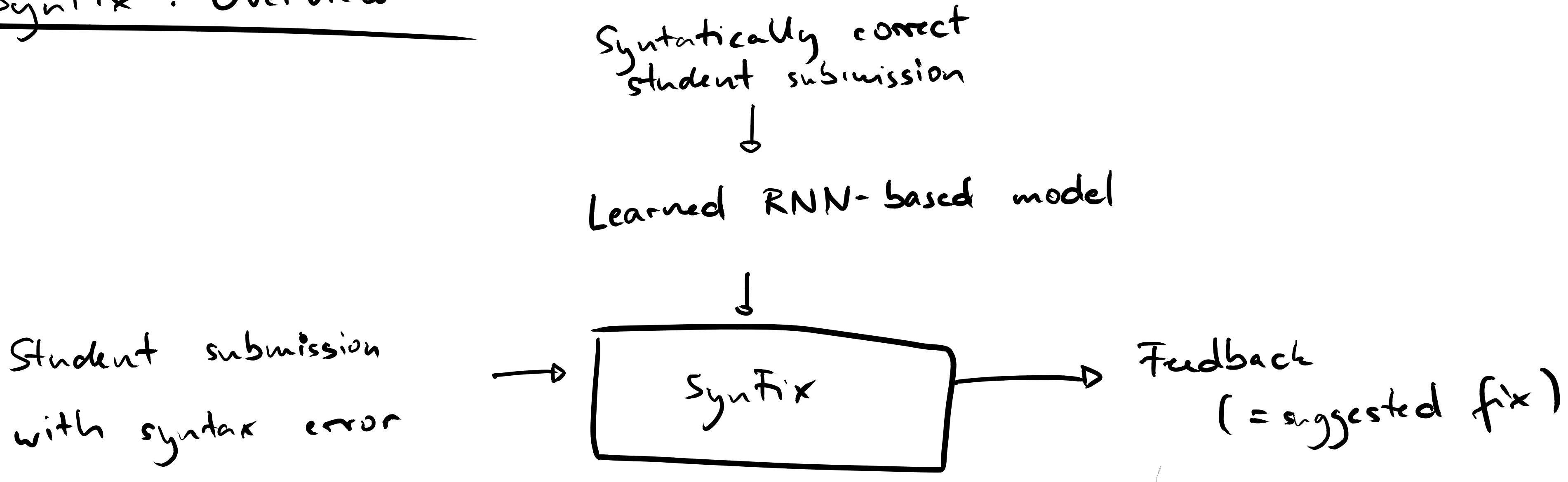
```
def recurPower (base , exp):
  if exp == 0:
    return base ←
  else:
    return (base * recurPower (base ,exp - 1))
```

Example (2)

```
def recurPower (base , exp):
  if exp == 0:
    return base ←
  else:
    return (base * recurPower (base ,exp - 1))
```

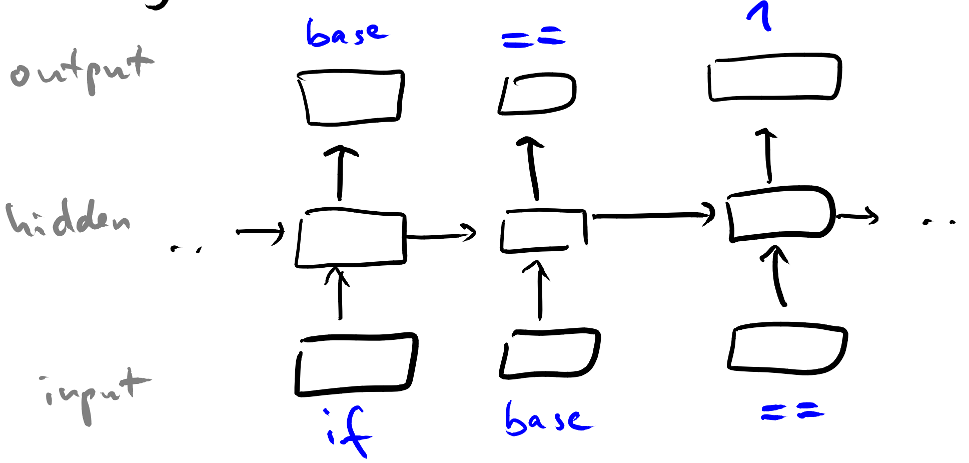
Beware: Fix of syntax error may not be the semantically correct fix

SynFix : Overview


Syntatically correct student submission
Learned RNN-based model
Student submission with syntax error
SynFix
Feedback (= suggested fix)

RNN-based Model

Program = Sequence of tokens



- Training: Expected output sequence = Input sequence shifted by one
- Prediction: Provide partial program until error location & generate next token(s)

SynFix Algorithm

Given: Program with syntax error + error location

Steps:

- Parse and tokenize program
- Query network with prefix of tokens until error location
- Try if inserting or replacing one or more tokens fixes the error
- If not: Delete line with error and query network with prefix until the error line
- Try if inserting predicted tokens fixes the error

Summary

- Recurrent Neural Networks (RNNs)
  - Powerful class of neural networks
  - Most effective for inputs (and outputs) that are sequences
- Two applications
  - Code completion:
    Predict next calls based on previous calls
  - Repair of syntax errors:
    Predict correct tokens based on previous tokens