Secure Your Code with AI and NLP

Dr. Eliezer Kanal
Mr. Ben Cohen
Dr. Nathan VanHoudnos
Natural Language Processing

- Raw text
- Machine-friendly representation
- Find patterns, repetition
  - Predictions
  - Generate natural sequences
  - Summarize
  - Translate
  - Classify
Code + NLP = ?
“Naturalness Hypothesis”

Programming languages, in theory, are complex, flexible and powerful, but the programs that real people actually write are mostly simple and rather repetitive, and thus they have usefully predictable statistical properties that can be captured in statistical language models and leveraged for software engineering tasks.

Natural Language Processing

Raw text

Machine-friendly representation

Find patterns, repetition

- Predictions
- Generate natural sequences
- Summarize
- Translate
- Classify
What is “representation”?

Discourse
| Pragmatics
| Semantics
| Syntax
| Lexemes

Morphology

| Phonetics — Phonology
| Orthography
What is “representation”?

Discourse

Pragmatics

Semantics

Syntax

Lexemes

Phonetics — Phonology

Morphology

Orthography

Breaking words to components

תפגוש את הילד בן
You will meet the boy in the park
What is “representation”?

- Discourse
- Pragmatics
- Semantics
- Syntax
- Lexemes
- Phonetics — Phonology
- Morphology
- Orthography

Normalize/disambiguate words

- Bank (finance)
- Bank (river)
- Bank (airplane)
What is “representation”?

- Discourse
- Pragmatics
- Semantics
- Syntax
- Lexemes
  - Phonetics — Phonology
  - Morphology
  - Orthography

Put symbols in a hierarchy

see example....
One morning I shot an elephant in my pajamas. How he got into my pajamas I don’t know.

Groucho Marx, *Animal Crackers*, 1930

![Parse trees for the sentence](image)

**Figure 11.2** Two parse trees for an ambiguous sentence. The parse on the left corresponds to the humorous reading in which the elephant is in the pajamas, the parse on the right corresponds to the reading in which Captain Spaulding did the shooting in his pajamas.

What is “representation”?

- Discourse
- Pragmatics
- Semantics
- Syntax
- Lexemes
- Morphology
- Phonetics — Phonology
- Orthography

Sentences to domain representation

(Speaking to phone) “Remind me to buy groceries when I leave the house”
What is “representation”? 

Discourse

<table>
<thead>
<tr>
<th>Pragmatics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantics</td>
</tr>
<tr>
<td>Syntax</td>
</tr>
<tr>
<td>Lexemes</td>
</tr>
</tbody>
</table>
| Morphology
  | Phonetics — Phonology |
  | Orthography |

Non-local meanings

“Please pass that down.”
What is “representation”?

Discourse

| Pragmatics
| Semantics
| Syntax
| Lexemes

Phonetics — Phonology

Morphology

Orthography

Sequences, Conversation

“I said the *black* shoes.”
“Oh, *black*.”
var exerciseTimer = function (exercises) {
  $("#workouts").hide();

  var time = document.getElementById("time");
  var desc = document.getElementById("desc");

  var i = 0;
  var exercise = exercises.workout[i];
  var tt = setInterval(function () {
    desc.textContent = exercise[0];
    time.textContent = exercise[1];

    document.getElementById("time").textContent = exercise[1].toFixed(0);

    if (exercise[1] <= 0) {
      i++;
      exercise = exercises.workout[i];
      if (i > exercises.workout.length - 1) {
        setTimeout(function () {
          clearInterval(tt);
          desc.textContent = "You're done!";
          time.textContent = "";
          $("#workouts").show();
        }, 1000);
      }
    }
  }, 1000);

  desc.textContent = exercise[0];
  time.textContent = exercise[1];

  https://github.com/eykanal/exerciseTimer/blob/master/js/timer.js
var exerciseTimer = function (exercises) {
    $('#workouts').hide();

    var time = document.getElementById("time");
    var desc = document.getElementById("desc");

    var i = 0;
    var exercise = exercises.workout[i];
    var tt = setInterval(function () {
        desc.textContent = exercise[0];
        time.textContent = exercise[1];

        document.getElementById("time").textContent = exercise[1].toFixed(0);

        if (exercise[1] <= 0) {
            i++;
            exercise = exercises.workout[i];
            if (i > exercises.workout.length - 1) {
                clearInterval(tt);
                desc.textContent = "You're done!";
                time.textContent = "";
                $('#workouts').show();
            }, 1000);
        }
    }, 1000);

    desc.textContent = exercise[0];
    time.textContent = exercise[1];

    https://github.com/eykanal/exerciseTimer/blob/master/js/timer.js
```javascript
var exerciseTimer = function (exercises) {
  $('#workouts').hide();
  var time = document.getElementById("time");
  var desc = document.getElementById("desc");

  var i = 0;
  var exercise = exercises.workout[i];
  var tt = setInterval(function () {
    desc.textContent = exercise[0];
    time.textContent = exercise[1];

    document.getElementById("time").textContent = exercise[1].toFixed(0);

    if (exercise[1] <= 0) {
      i++;
      exercise = exercises.workout[i];
      if (i > exercises.workout.length - 1) {
        setTimeout(function () {
          clearInterval(tt);
          desc.textContent = "You're done!";
          time.textContent = "";
          $('#workouts').show();
        }, 1000);
      }
    }
  }, 1000);
  desc.textContent = exercise[0];
  time.textContent = exercise[1];
};
```

https://github.com/eykanal/exerciseTimer/blob/master/js/timer.js
```javascript
var exerciseTimer = function (exercises) {
    $('#workouts').hide();

    var time = document.getElementById('time');
    var desc = document.getElementById('desc');

    var i = 0;
    var exercise = exercises.workout[i];
    var tt = setInterval(function () {
        desc.textContent = exercise[0];
        time.textContent = exercise[1];

        document.getElementById('time').textContent = exercise[1].toFixed(0);

        if (exercise[1] <= 0) {
            i++;
            exercise = exercises.workout[i];
            if (i > exercises.workout.length - 1) {
                setTimeout(function () {
                    clearInterval(tt);
                    desc.textContent = "You're done!";
                    time.textContent = "";
                    $('#workouts').show();
                }, 1000);
            }
        }
    }, 1000);
    desc.textContent = exercise[0];
    time.textContent = exercise[1];
};
```

https://github.com/eykanal/exerciseTimer/blob/master/js/timer.js
var exerciseTimer = function (exercises) {
    $('#workouts').hide();

    var time = document.getElementById('time');
    var desc = document.getElementById('desc');

    var i = 0;
    var exercise = exercises.workout[i];
    var tt = setInterval(function () {
        desc.textContent = exercise[0];
        time.textContent = exercise[1];

        document.getElementById("time").textContent = exercise[1].toFixed(0);

        if (exercise[1] <= 0) {
            i++;
            exercise = exercises.workout[i];
            if (i > exercises.workout.length - 1) {
                setTimeout(function (){
                    clearInterval(tt);
                    desc.textContent = "You're done!";
                    time.textContent = ";"
                    $('#workouts').show();
                }, 1000);
            }
        }
    }, 1000);

    desc.textContent = exercise[0];
    time.textContent = exercise[1];
};

https://github.com/eykanal/exerciseTimer/blob/master/js/timer.js
A Survey of Machine Learning for Big Code and Naturalness

MILTIAKIS ALLAMANIS, Microsoft Research
EARL T. BARR, University College London
PREMKUMAR DEVANBU, University of California, Davis
CHARLES SUTTON, University of Edinburgh and The Alan Turing Institute

NLP for “Big Code”:

• Code-generating models
• Representational models
• Pattern mining models

Code generating models – $n$-grams

“I made a peanut butter and jelly ______.”

Bigram: “jelly ______”

$$P \left( w_n | w_{1}^{n-1} \right) \approx P \left( w_n | w_{n-1} \right)$$

5-gram: “peanut butter and jelly ______”

$$P \left( w_n | w_{1}^{n-1} \right) \approx P \left( w_n | w_{n-4}^{n-1} \right)$$

General case:

$$P \left( w_n | w_{1}^{n-1} \right) \approx P \left( w_n | w_{n-N+1}^{n-1} \right)$$
n-grams

```python
for i in range(10):

    Bigram: "10"

4-gram: "range(10)"

6-gram: "i in range(10)"
```
**n-grams – Does it work?**

3- or 4-grams optimal for both natural language and code

Code 5x more regular (predictable) than natural language

2\textsuperscript{nd} study (not shown) suggests ~62k LOC needed for code language model

![Graph showing comparison of English cross-entropy versus code cross-entropy](image)

Figure 1. Comparison of English cross-entropy versus the code cross-entropy of 10 Java projects.

n-grams – Does it work?

Built autocomplete augmenter first 2, 6, or 10 suggestions from ngrams model (10 shown)

![Graph showing suggestion gains from merging n-gram suggestions into those of Eclipse.](image)

Figure 4. Suggestion gains from merging n-gram suggestions into those of Eclipse.

Embeddings – word2vec

• How do computers represent what a word “means”?

• Ontologies (e.g., WordNet) – list all words & relationships
  - tedious (read: expensive) to build
  - often miss relationships
  - impossible to keep up-to-date

• Basic problem: discrete representation of words fails
  - e.g., “hotel” = [0 0 0 ... 0 0 0 1 0 ... 0 0]
    “motel” = [0 0 0 ... 0 1 0 0 0 ... 0 0]
  - Can’t use typical math tools (dot product, cosine similarity)
  - Expensive to maintain secondary mapping vectors

Embeddings – word2vec

“You shall know a word by the company it keeps”
(Firth, J. R. 1957:11)

**word2vec**: represent meaning by frequency of words appearing in similar context

Usually, the large-scale *factory* is portrayed as a product of capitalism…

At the magnetron workshop in the old biscuit *factory*, Fisk sometimes wore a striped…

These words will represent “factory”
Embeddings - Maps

https://www.benfrederickson.com/multidimensional-scaling/
Embeddings – word2vec

Figure 2: Left panel shows vector offsets for three word pairs illustrating the gender relation. Right panel shows a different projection, and the singular/plural relation for two words. In high-dimensional space, multiple relations can be embedded for a single word.

Somewhat surprisingly, it was found that similarity of word representations goes beyond simple syntactic regularities. Using a word offset technique where simple algebraic operations are performed on the word vectors, it was shown for example that \texttt{vector("King") - vector("Man") + vector("Woman") results in a vector that is closest to the vector representation of the word Queen [20].}


Embeddings

- How it works: [https://jalammar.github.io/illustrated-word2vec/](https://jalammar.github.io/illustrated-word2vec/)
  
  ...also a million other sites

- Advances: [doc2vec](https://jalammar.github.io/illustrated-word2vec/), [seq2seq](https://jalammar.github.io/illustrated-word2vec/), numerous others

**code2vec** – find code vectors!

Step back – Language model

“Assign a probability to a sequence of words”

Language:

Roethlisberger is a better QB than Brady
Colorless green ideas sleep furiously

Code:

for i in range(10):
    print(i)

52 var % function eeeee class ".("
Step back – Language model

Model* built from training codebase
- Code symbols
- Other details in the dataset

* Assign a probability to a sequence of words

Possible uses?
- Examine frequency of symbols
- Given some code, what is “similar” code?
- Given non-code input (e.g., comments, requirements), what code best matches input?
Embeddings – code2vec

**code2vec: Learning Distributed Representations of Code**

URI ALON, Technion  
MEITAL ZILBERSTEIN, Technion  
OMER LEVY, Facebook AI Research  
ERAN YAHAV, Technion

Grabbed a ton of code from Github (>10k Java code repos)

**Motivating question:** Can we predict a method name simply by looking at the method’s code?

Uses tokenized representation of AST (Abstract Syntax Trees) to describe code
int sum_square(int v1, int v2)
{
    return (v1+v2)*(v1+v2);
}
Step back (again) – Abstract Syntax Trees

\[ v_1, [(\text{Ref})v_1 \ ^ (\text{Op})+ \ ^ (\text{Op})* \ ^ (\text{Func}) _ (\text{Par})v_2], v_2 \]

**Diagram:**

- **Function:** `sum_square`
  - **Parameter:** `v1`
  - **Parameter:** `v2`
  - **Operator:** `*`
    - **Operator:** `+`
      - **Reference:** `v1`
      - **Reference:** `v2`
    - **Operator:** `+`
      - **Reference:** `v1`
      - **Reference:** `v2`

**Equation:**

\[ # \text{ of code paths} \approx # \text{ of Leaves}^2 \]
These are the “words” for code2vec
Embeddings – code2vec

MOST SIMILAR

...is similar to:

- login

COMBINATIONS

...combined, are similar to:

- equals
- and
- toLowerCase

logOut | 70.09%
authenticate | 66.47%
connect | 62.24%
save | 61.99%
subscribe | 61.67%
equalsIgnoreCase | 78.75%
isUpperCase | 75.82%
equiv | 75.72%
sameAs | 75.31%
isLowerCase | 74.65%

https://code2vec.org/
Embeddings – code2vec

**ANALOGIES**

- receive is to download as send

...is to:

- **upload** | 76.38%
- **delete** | 71.53%
- **connect** | 70.51%
- **install** | 68.1%
- **update** | 67.03%

[https://code2vec.org/](https://code2vec.org/)
ML for clean code

Coding conventions are critical for medium-to-large teams

- Prevent bugs
- Make code easier to read, navigate, & maintain

<table>
<thead>
<tr>
<th>Learning Natural Coding Conventions</th>
</tr>
</thead>
</table>
| Miltiadis Allamanis†<br>School of Informatics<br>University of Edinburgh<br>Edinburgh, EH8 9AB, UK<br>{m.allamanis, csutton}@ed.ac.uk | Earl T. Barr‡<br>Dept. of Computer Science<br>University College London<br>London, UK<br>e.barr@ucl.ac.uk | Christian Bird§<br>Microsoft Research<br>Redmond, WA, USA<br>christian.bird@microsoft.com | Charles Sutton†<br>Microsoft
ML for clean code

Figure 2: A screenshot of the devstyle Eclipse plugin. The user has requested suggestion for alternate names of the each argument.
## ML for code security

<table>
<thead>
<tr>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find bugs themselves</td>
</tr>
<tr>
<td>Automatically write secure code</td>
</tr>
<tr>
<td>Create good documentation</td>
</tr>
<tr>
<td>AI also brews a good cup of coffee</td>
</tr>
</tbody>
</table>

AI also brews a good cup of coffee.
Find bugs themselves

- Most of your code is (probably) correct
- Buggy code is rare
- If you see rare code similar to common code, it’s probably buggy
Find bugs themselves

Bugram: Bug Detection with N-gram Language Models

Song Wang*, Devin Chollak†, Dana Movshovitz-Attias†, Lin Tan*
*Electrical and Computer Engineering, University of Waterloo, Canada
†Computer Science Department, Carnegie Mellon University, USA
*{song.wang, dchollak, lintan}@uwaterloo.ca, †dma@cs.cmu.edu

Source Files → Tokenization → Token Sequences → N-gram Model Building → Ranked Token Sequences → Bug Detection → Potential Bugs

Figure 3: Overview of Bugram

(a) Method call sequence from a buggy code snippet (appears once): [isDebugEnabled(), debug(), indent(), stringify()]

```java
if (LOG.isDebugEnabled()) {
  LOG.debug(indent(depth)+"converting from
  Pig " + pigType + " " + value +
  " using " + stringify(schema));
}
```

(b) A similar but correct method call sequence (appears three times): [isDebugEnabled(), debug(), indent(), toString()]

```java
if (LOG.isDebugEnabled()) {
  LOG.debug(indent(depth)+"converting from
  Pig " + pigType + " " +
  toString(value) +
  " using " + stringify(schema));
}
```

Figure 2: A motivating example from the latest version 0.15.0 of the project Pig. Bugram automatically detected a real bug in (a), which has been confirmed and fixed by Pig developers after we reported it.
Find bugs themselves

Similar to previous work (same authors), Deep Belief Networks instead of $n$-grams

Motivating example: case where bag-of-words would fail

Think back… which techniques would work? Which wouldn’t?

Find bugs themselves

Figure 4: Overview of our proposed DBN-based feature generation and defect prediction
Learning to Generate Pseudo-code from Source Code using Statistical Machine Translation

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura
Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
{oda.yusuke.on9, fudaba.hiroyuki.ev6, neubig, hata, ssakti, tomoki, s-nakamura}@is.naist.jp

Four requirements listed:

- Accuracy
- Speed
- Automated
- On-demand

Code-to-Text

“SMT” – Statistical Machine Translation

• Find relationships between tokens in different language models

• Propose many sentences, use statistical models to identify “best”
Fig. 2. Example of Python to English PBMT pseudo-code generation.
Code-to-Text

- Very impressive application of NLP to software domain
- Limitations: text is very pedantic, misses “big picture”
- More work described in Allamanis survey paper
Summary

NLP concepts can apply to code (“naturalness hypothesis”)

Techniques we discussed:

- $n$-grams, Annotated $n$-grams
- Embeddings (word2vec, code2vec)

Applications:

- Bug identification
- Code completion
- Documentation generation
Contact Us

Carnegie Mellon University
Software Engineering Institute
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
412-268-5800
888-201-4479
info@sei.cmu.edu
www.sei.cmu.edu