



Secure Your Code Using AI

Dr. Eliezer Kanal
Technical Manager

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Copyright 2019 Carnegie Mellon University. All Rights Reserved.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

Carnegie Mellon® and CERT® are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM19-0416

The SEI is a DoD R&D Federally Funded Research and Development Center



Established in 1984 at Carnegie Mellon University

~650 employees (ft + pt), of whom about 70% are engaged in technical work

Initiated CERT cybersecurity program in 1988

Offices in Pittsburgh and DC, with several locations near customer facilities

~\$140M in annual funding

Natural Language Processing

NLP automates language **acquisition**, **processing**, and **generation**



Acquisition = $f(\text{data, algorithm}) \rightarrow$ new representations

Processing = $f(\text{raw language}) \rightarrow$ usable representation

Generation = $f(\text{representation}) \rightarrow$ new generated “language”

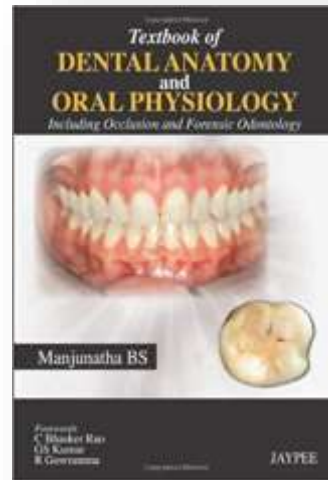
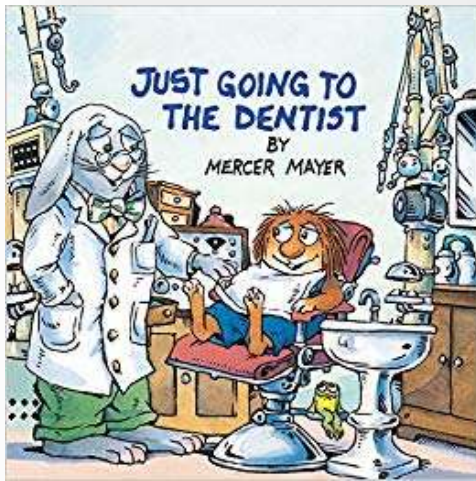
This slide left blank intentionally



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.



A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. Devanbu, “On the naturalness of software,” in *2012 34th International Conference on Software Engineering (ICSE)*, 2012, pp. 837–847.

Natural Language Processing

NLP automates language **acquisition**, **processing**, and **generation**

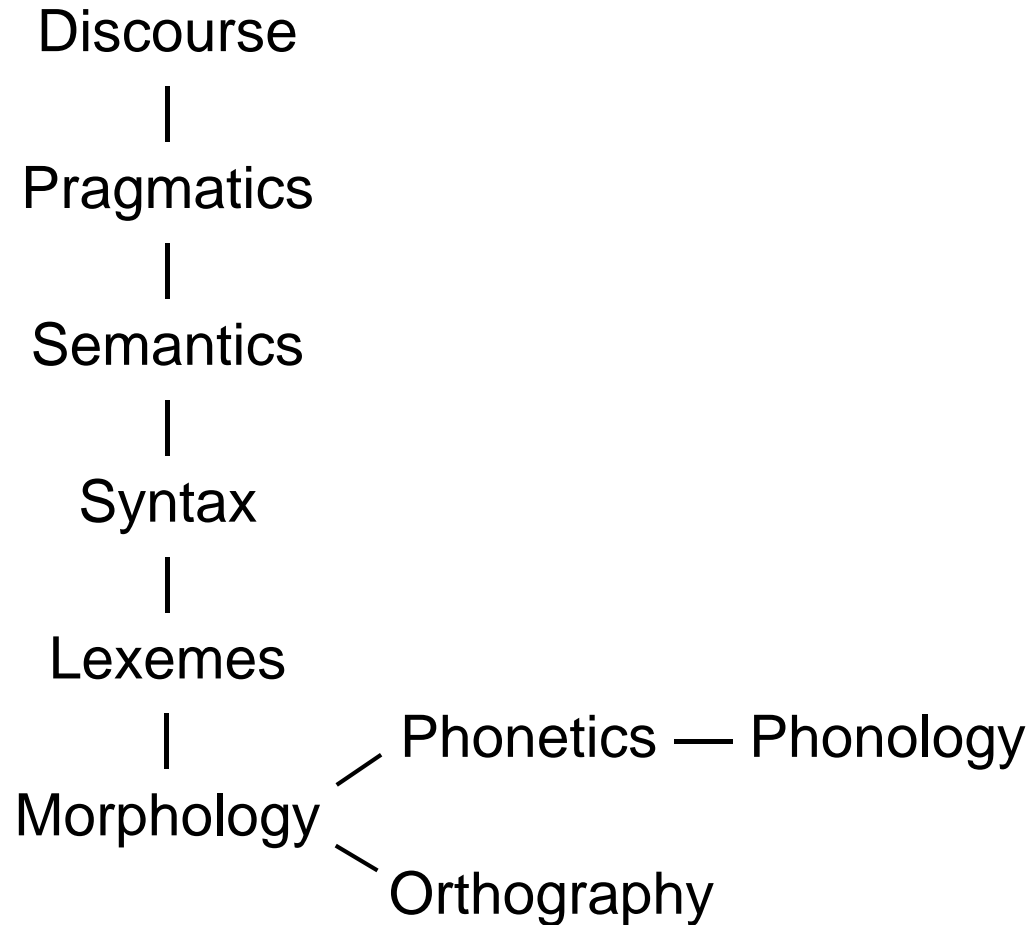


Acquisition = $f(\text{data, algorithm}) \rightarrow$ new representations

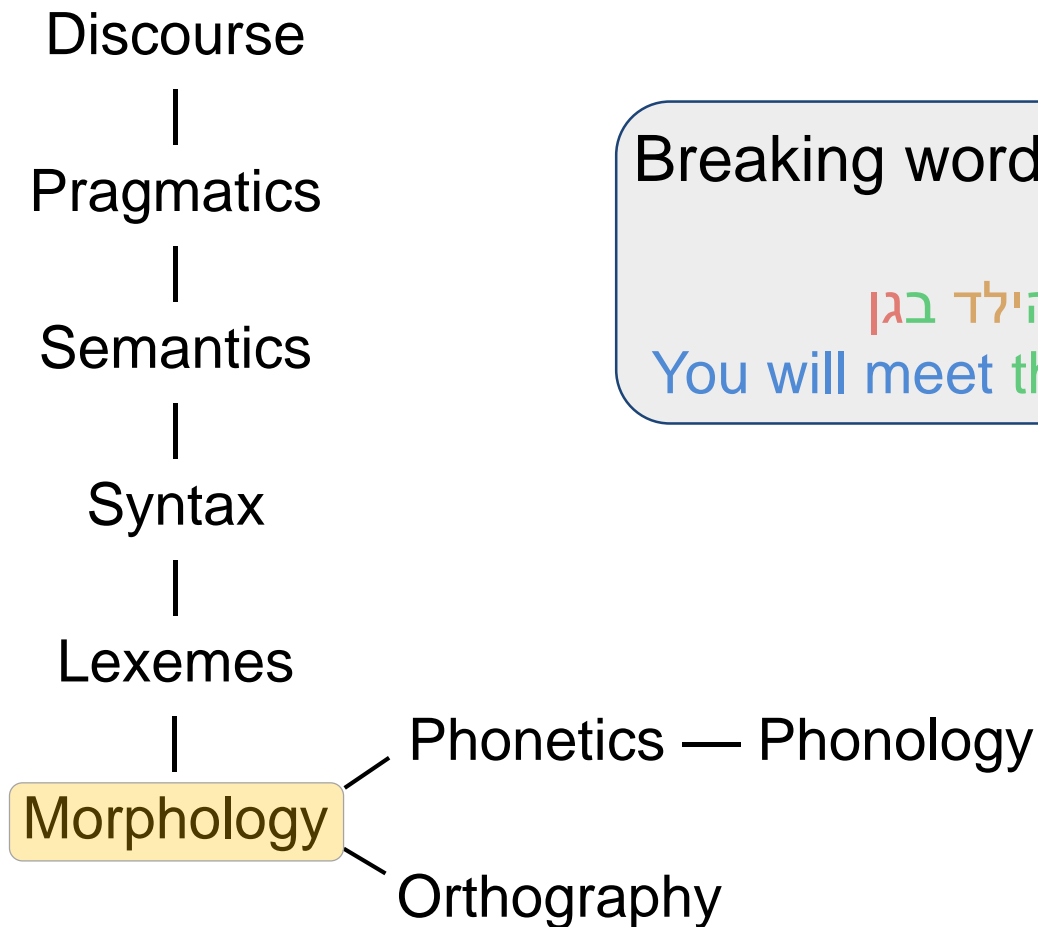
Processing = $f(\text{raw language}) \rightarrow$ usable representation

Generation = $f(\text{representation}) \rightarrow$ new generated “language”

What is “representation”?



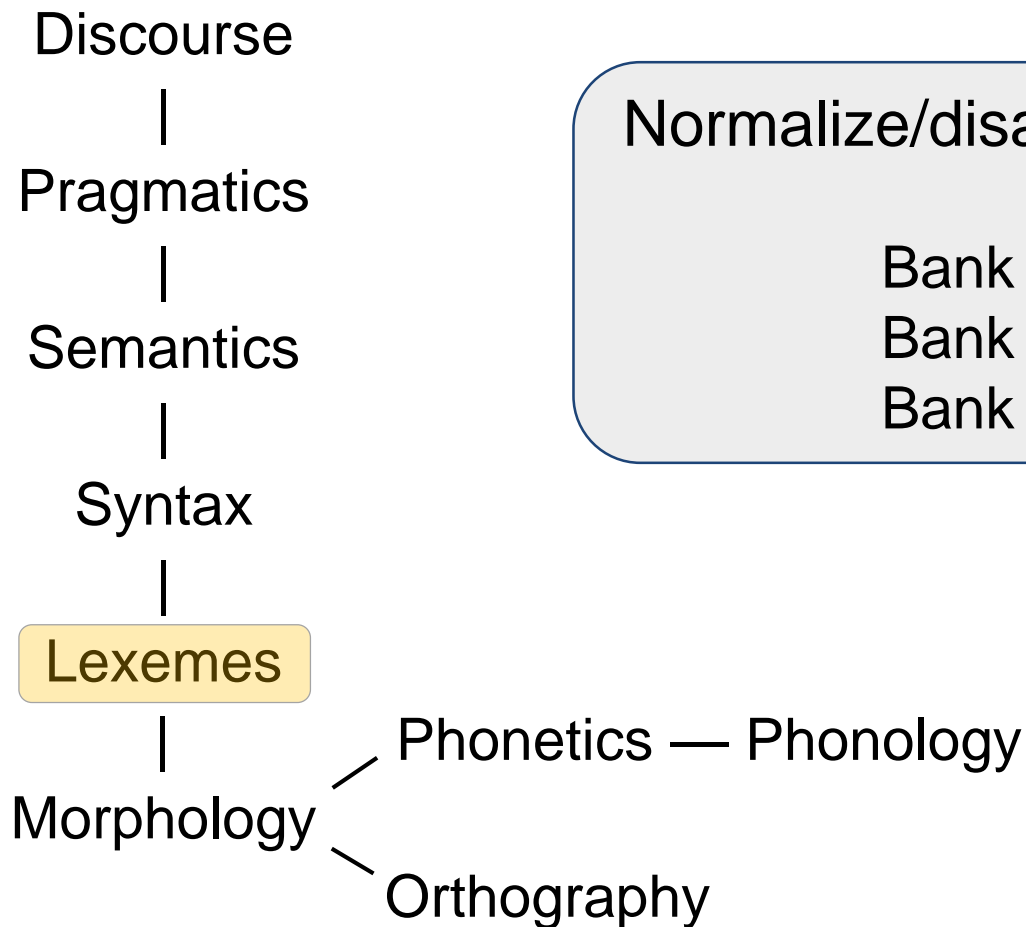
What is “representation”?



Breaking words to components

תפגוש את הילד בגן
You will meet the boy in the park

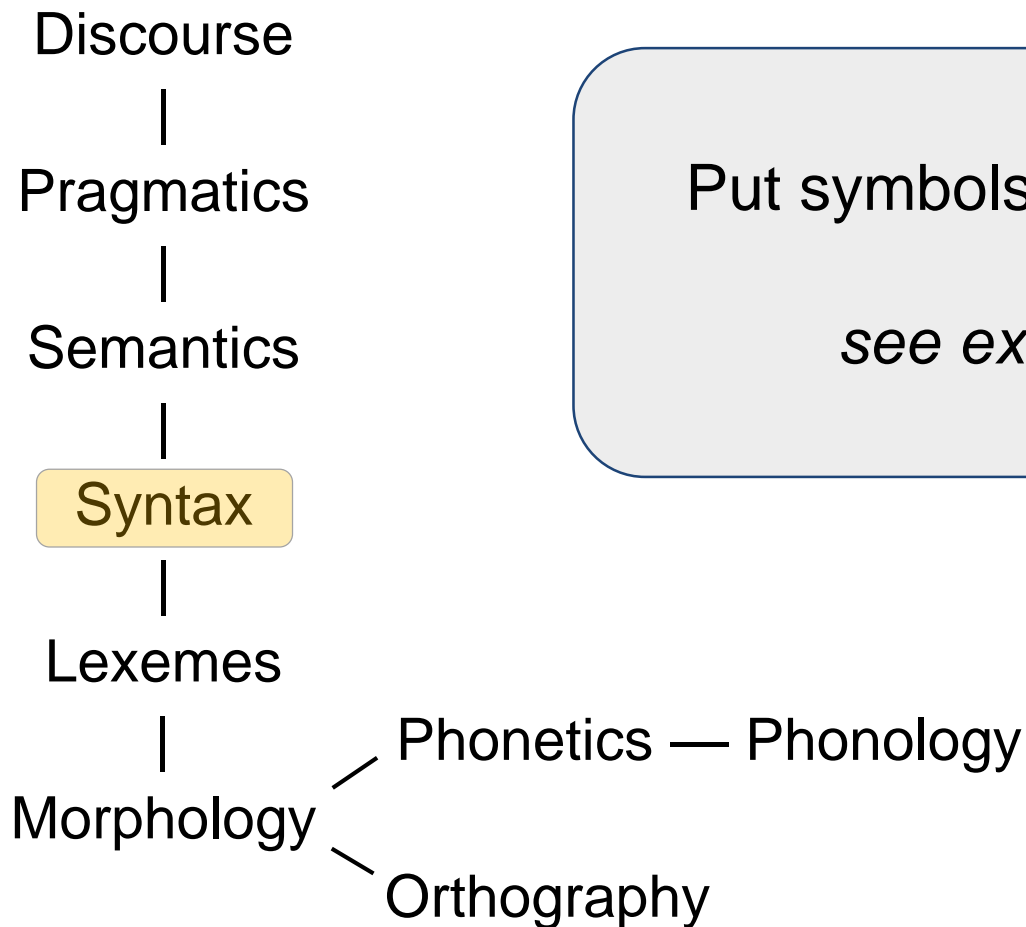
What is “representation”?



Normalize/disambiguate words

Bank (finance)
Bank (river)
Bank (airplane)

What is “representation”?



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

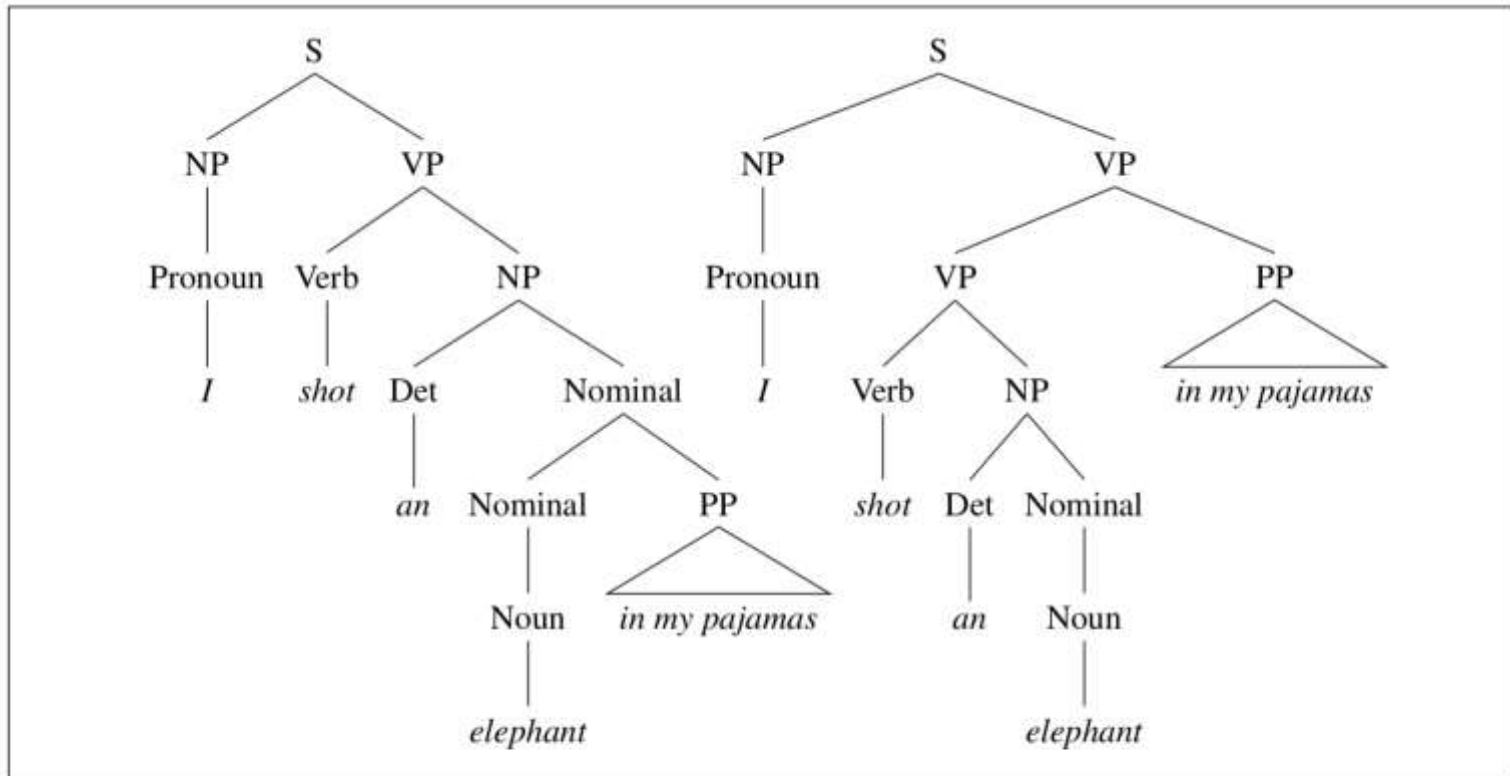
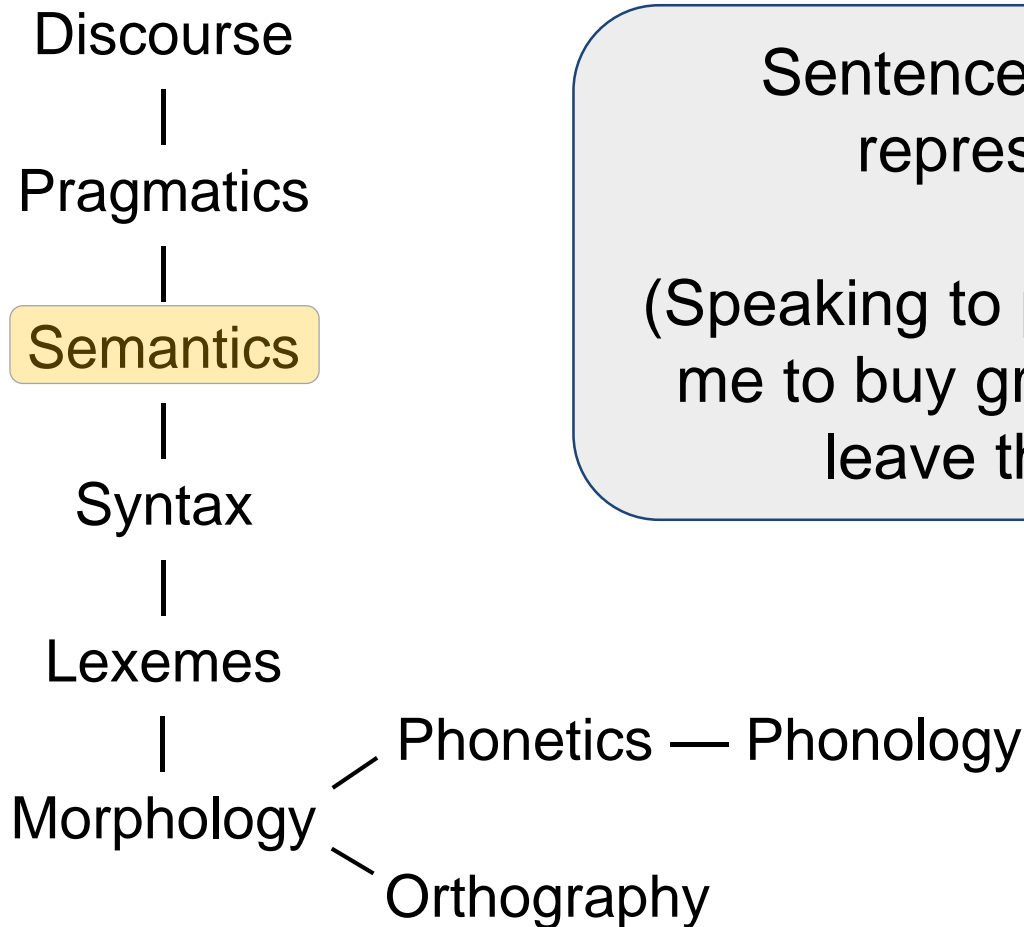


Figure 11.2 Two parse trees for an ambiguous sentence. The parse on the left corresponds to the humorous reading in which the elephant is in the pajamas, the parse on the right corresponds to the reading in which Captain Spaulding did the shooting in his pajamas.

D. Jurafsky and J. H. Martin, *Speech and Language Processing*, Third edition, Draft. Self-published, 2018.

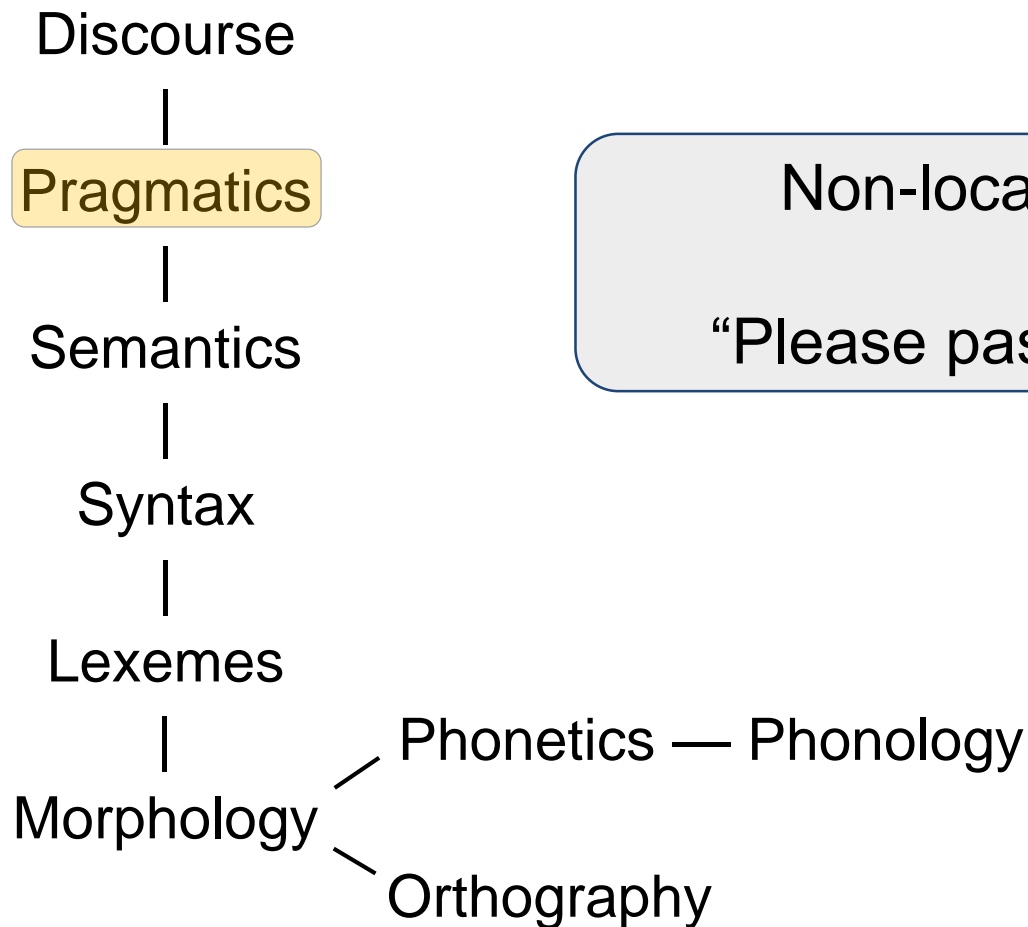
What is “representation”?



Sentences to domain representation

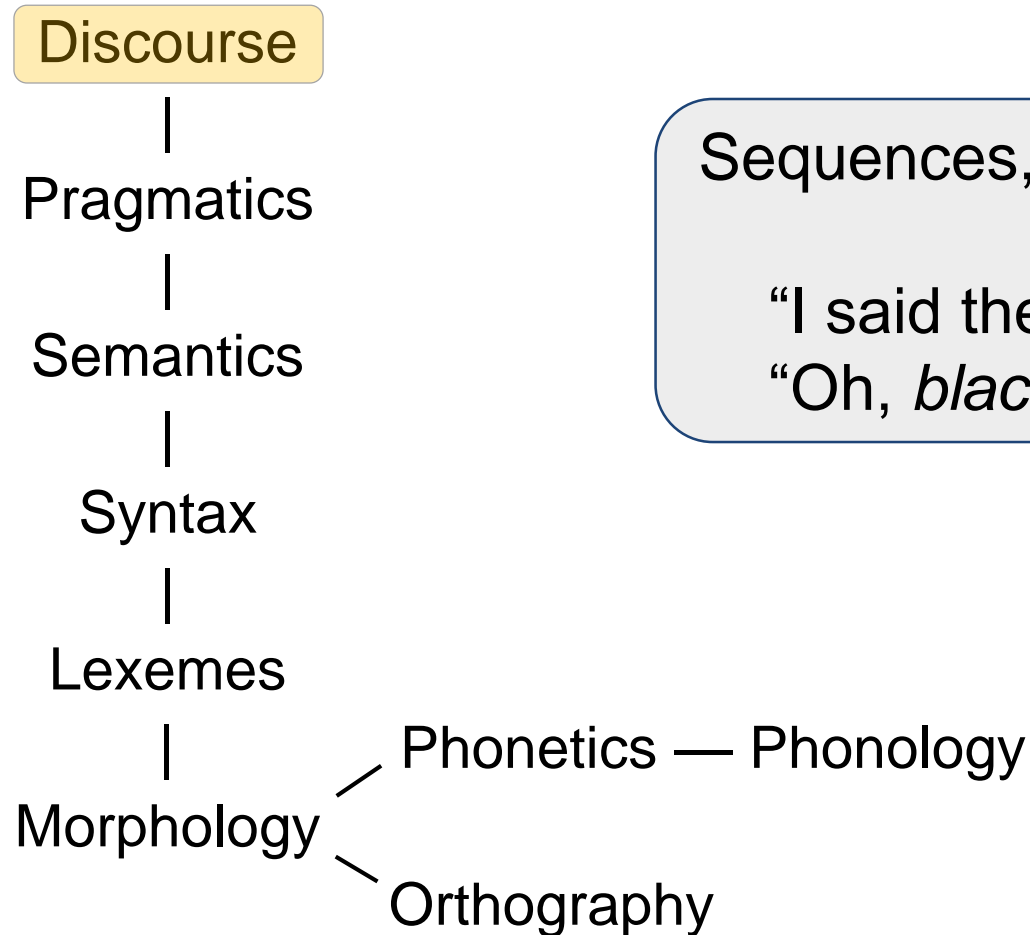
(Speaking to phone) “Remind me to buy groceries when I leave the house”

What is “representation”?



Non-local meanings
“Please pass that down.”

What is “representation”?



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);
        exercise[1] = exercise[1] - 1;

        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);
        exercise[1] = exercise[1] - 1;

        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];
};

```

Symbols (morphology)

<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);
        exercise[1] = exercise[1] - 1;

        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];
};

```

Lexeme (context)

<https://github.com/eykanal/exerciseTimer/blob/master/js/timer.js>

Syntax

We all know this one

```
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 = "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);
        exercise[1] = exercise[1] - 1;

        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];
};

```

Pragmatics, Discourse

Complex apps
APIs

<https://github.com/eykanal/exerciseTimer/blob/master/js/timer.js>

Natural Language Processing

NLP automates language **processing**, **generation**, and **acquisition**



Acquisition = $f(\text{data, algorithm}) \rightarrow$ new representations

Processing = $f(\text{raw language}) \rightarrow$ usable representation

Generation = $f(\text{representation}) \rightarrow$ new generated “language”

With code, also **Manipulation**:

$f(\text{code}) \rightarrow$ representation \rightarrow modifications \rightarrow better code!

NLP for “Big Code”

A Survey of Machine Learning for Big Code and Naturalness

MILTADIS 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

**A++++
WOULD READ AGAIN**

M. Allamanis, E. T. Barr, P. Devanbu, and C. Sutton, “A Survey of Machine Learning for Big Code and Naturalness,” Sep. 2017.

Code generating models – n -grams

“I made a peanut butter and jelly _____.”

Bigram: “jelly _____”

$$P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-1})$$

5-gram: “peanut butter and jelly _____”

$$P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-4}^{n-1})$$

General case:

$$P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-N+1}^{n-1})$$



n-grams

```
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

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

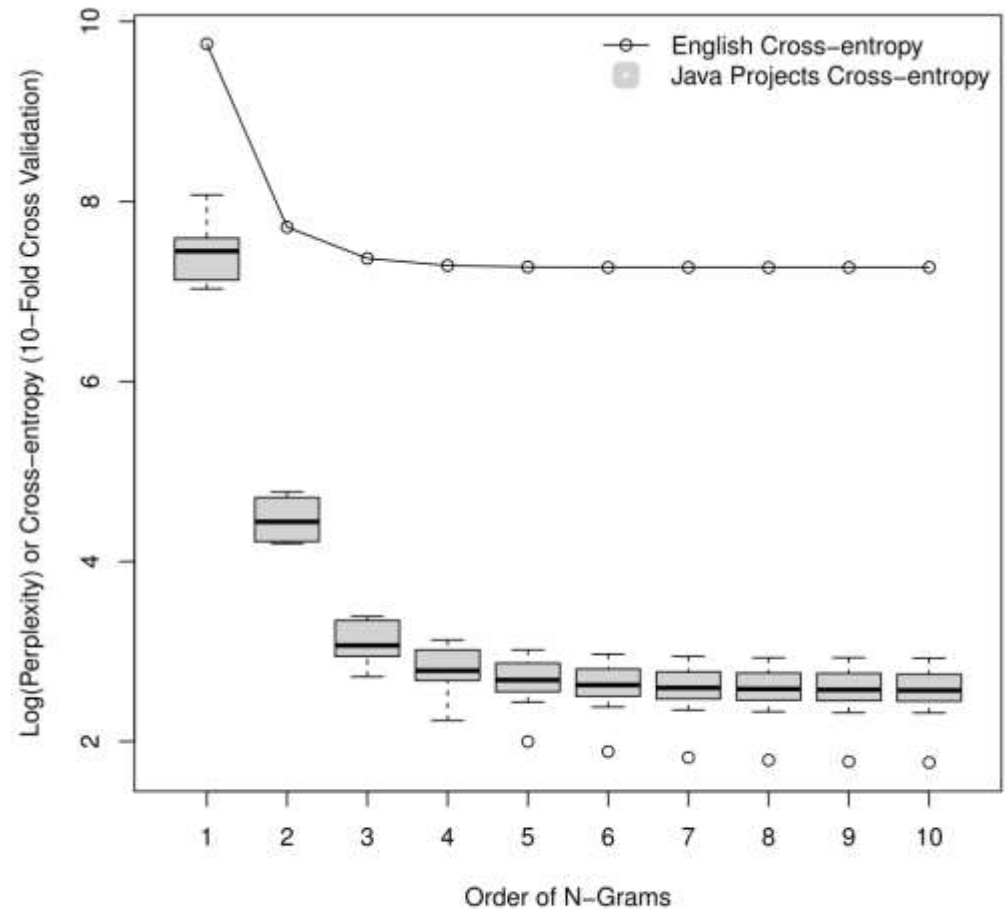
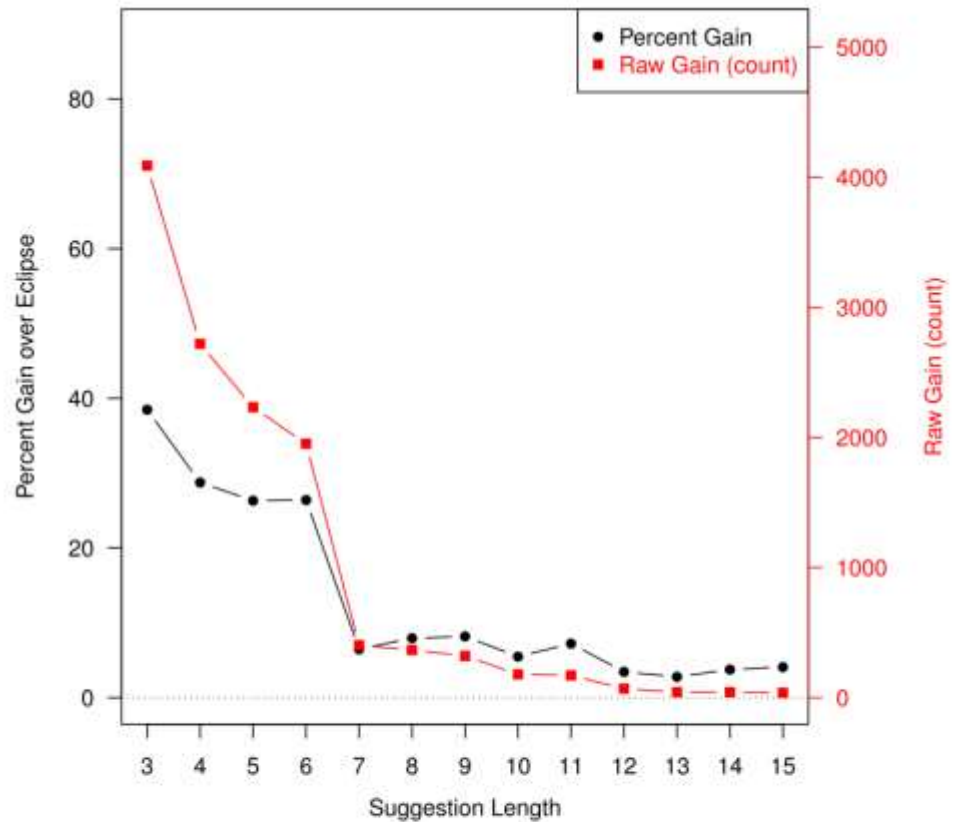


Figure 1. Comparison of English cross-entropy versus the code cross-entropy of 10 Java projects.

A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. Devanbu, "On the naturalness of software," in *2012 34th International Conference on Software Engineering (ICSE)*, 2012, pp. 837–847.

n -grams – Does it work?

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



(c) Gain using top 10 suggestions.

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

A. Hindle, E. T. Barr, Z. Su, M. Gabel, and P. Devanbu, "On the naturalness of software," in *2012 34th International Conference on Software Engineering (ICSE)*, 2012, pp. 837–847.



Annotated n -grams

```
len = str.length()
```

n -grams have no context; lets add some!

Paper describes different feature combinations and their effectiveness

Note: they had to build these mappings

Table 1: Lexical Code Tokens from “len = str.length();”

Lexeme	Token Type
len	Identifier
=	Equal (Symbol)
str	Identifier
.	Period (Symbol)
length	Identifier
(Left parenthesis (Symbol)
)	Right parenthesis (Symbol)
;	Semicolon (Symbol)

Table 3: Associated Information of Semantic Code Tokens for len=str.length()

ID	Role	Sememe	Lexeme	Scope	Depend
T1	Variable	VAR[int]	len	C1.M2.B3	[T3,T5]
T2	Operator	OP[assign]	=	C1.M2.B3	NA
T3	Variable	VAR[String]	str	C1.M2.B3	[T1,T5]
T4	Operator	OP[access]	.	C1.M2.B3	NA
T5	Method	CALL[String,...]	length	C1.M2.B3	[T1,T3]

T. T. Nguyen, A. T. Nguyen, H. A. Nguyen, and T. N. Nguyen, “A statistical semantic language model for source code,” in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2013*, 2013, p. 532.

Annotated n -grams – Does it work?

Table 7: Accuracy (%) with Various Configurations

Model	Top-1	Top-2	Top-5	Top-10
1. Lexical n -gram model ([8])	53.6	60.6	66.1	68.8
2. Seman.	58.0	65.8	72.7	76.3
3. Seman. + cache	58.7	66.9	75.7	80.3
4. Seman. + cache + depend.	58.8	67.0	75.8	80.4
5. Seman. + cache + depend. + pair.assoc	59.3	67.5	76.1	81.4
6. Seman. + cache + depend. + LDA	58.9	67.1	76.0	81.3
7. Seman. + cache + depend. + n -gram topic	63.0	70.8	77.1	81.8
8. Seman. + cache + depend. + pair.assoc + n -gram topic [SLAMC]	64.0	71.9	78.2	82.3

Yup!

See paper for language and project specific accuracies

T. T. Nguyen, A. T. Nguyen, H. A. Nguyen, and T. N. Nguyen, "A statistical semantic language model for source code," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2013*, 2013, p. 532.

“Localness”

“A token used in the immediate past is much more likely to be used again soon than its overall probability that n-gram models would predict.”

...

“Our central hypothesis is: Source code is *locally repetitive*, viz. it has useful local regularities that can be captured in a *locally estimated cache* and leveraged for software engineering tasks.”

Z. Tu, Z. Su, and P. Devanbu, “On the localness of software,” in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2014*, 2014, pp. 269–280.

“Localness”

Table 2: Percentage of the *endemic* n -grams (only found in a single file). “Freq.” denotes the frequency of the n -grams in the file. The denominator is the number of n -grams in the corpus.

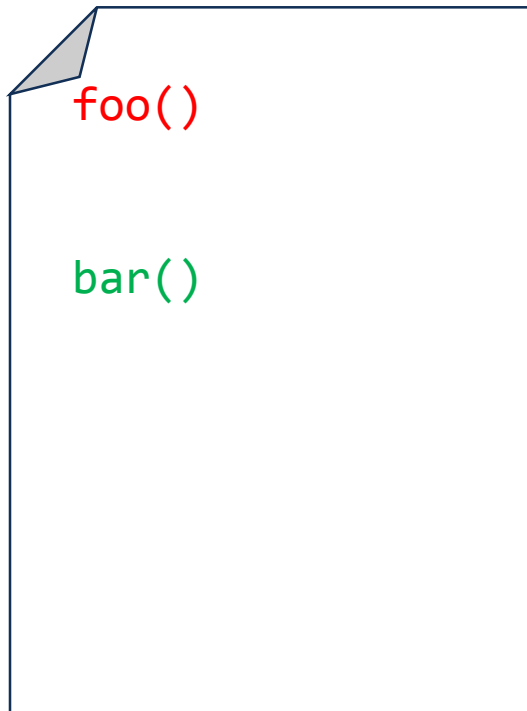
Lang.	Freq.	1-gram	3-gram	6-gram	10-gram
Java	≥ 1	4.65%	25.63%	53.98%	70.22%
	≥ 2	3.71%	11.13%	16.17%	14.34%
Python	≥ 1	8.58%	41.82%	73.65%	83.79%
	≥ 2	5.29%	14.44%	17.52%	12.47%
English	≥ 1	3.28%	72.52%	99.70%	100.00%
	≥ 2	1.39%	3.44%	0.65%	0.20%

Observations:

1. Many strings occur only once in one context and never again
2. Code is different than Natural Language

“Specificity”

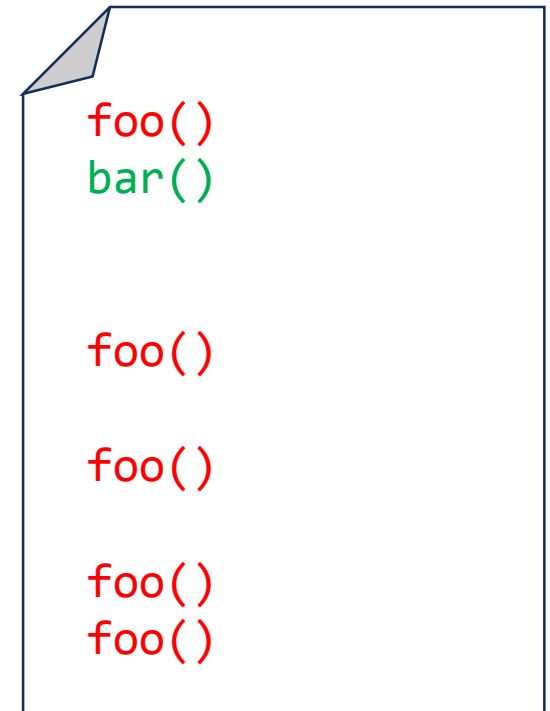
Code in many files, but present many times in just a few files



```
foo()  
  
bar()
```



```
bar()  
bar()  
  
foo()  
  
bar()
```



```
foo()  
bar()  
  
foo()  
  
foo()  
  
foo()  
foo()
```

“Proximity”

n-gram distance = the average # of tokens between occurrences

Observation: tokens locally more likely than uniformly spread across all files

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” = $[0 \ 0 \ 0 \ \dots \ 0 \ 0 \ 0 \ 1 \ 0 \ \dots \ 0 \ 0]$
“motel” = $[0 \ 0 \ 0 \ \dots \ 0 \ 1 \ 0 \ 0 \ 0 \ \dots \ 0 \ 0]$
 - Can’t use typical math tools (dot product, cosine similarity)
 - Expensive to maintain secondary mapping vectors

T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” Jan. 2013.

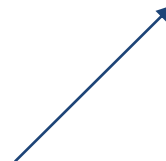
Embeddings – word2vec

“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”

Behemoth: A History of the Factory and the Making of the Modern World, by Joshua B. Freeman
The Idea Factory: Bell Labs and the Great Age of American Innovation, by Jon Gertner

Embeddings – word2vec

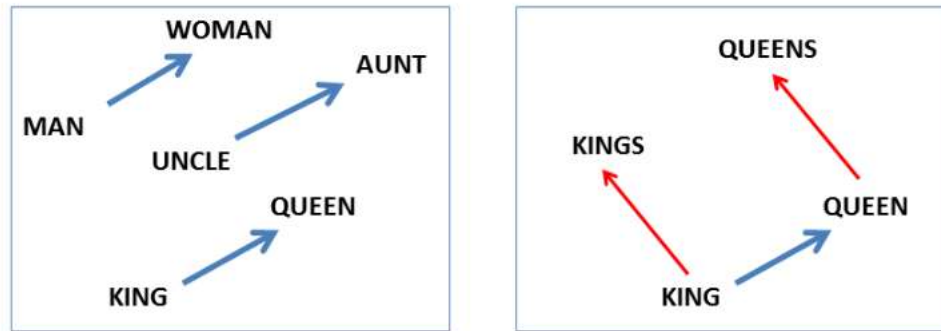


Figure 2: Left panel shows vector offsets for three word pairs illustrating the gender relation. Right panel shows a different projection, and the singular/plural relation for two words. In high-dimensional space, multiple relations can be embedded for a single word.

T. Mikolov, W. Yih, and G. Zweig, "Linguistic Regularities in Continuous Space Word Representations." pp. 746–751, 2013.

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 $vector("King") - vector("Man") + vector("Woman")$ results in a vector that is closest to the vector representation of the word *Queen* [20].

T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Jan. 2013.

Embeddings

- How it works: <https://jalammar.github.io/illustrated-word2vec/>
...also a million other sites
- Advances: [doc2vec](#), [seq2seq](#), numerous others

code2vec – find code vectors!

U. Alon, M. Zilberstein, O. Levy, and E. Yahav, “code2vec: Learning Distributed Representations of Code,” Mar. 2018.



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  
eeee class ".(
```

Entirely dependent on training data!

Step back – Language model



Model built from training codebase

- Code symbols
- Other details in the dataset

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)

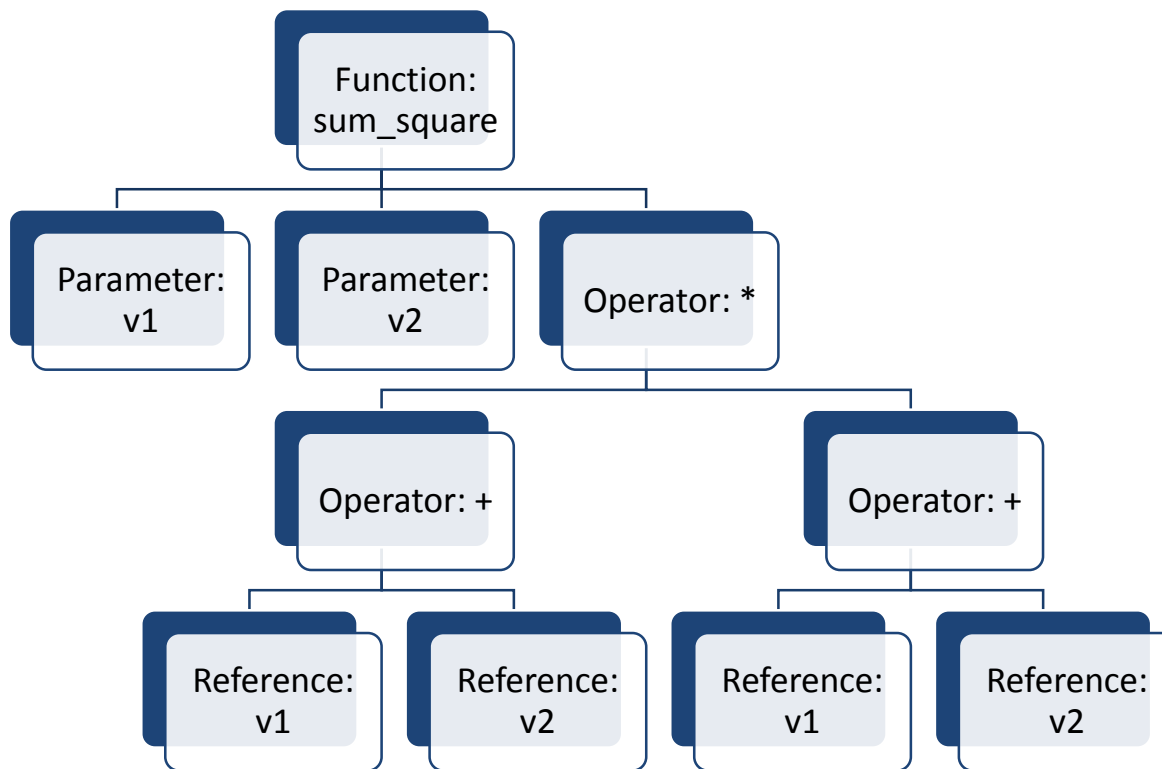
Motivating question: Can we predict a method name simply by looking at the method's code?

Uses tokenized representation of AST (Abstract Syntax Trees) to describe code

Step back (again) – Abstract Syntax Trees

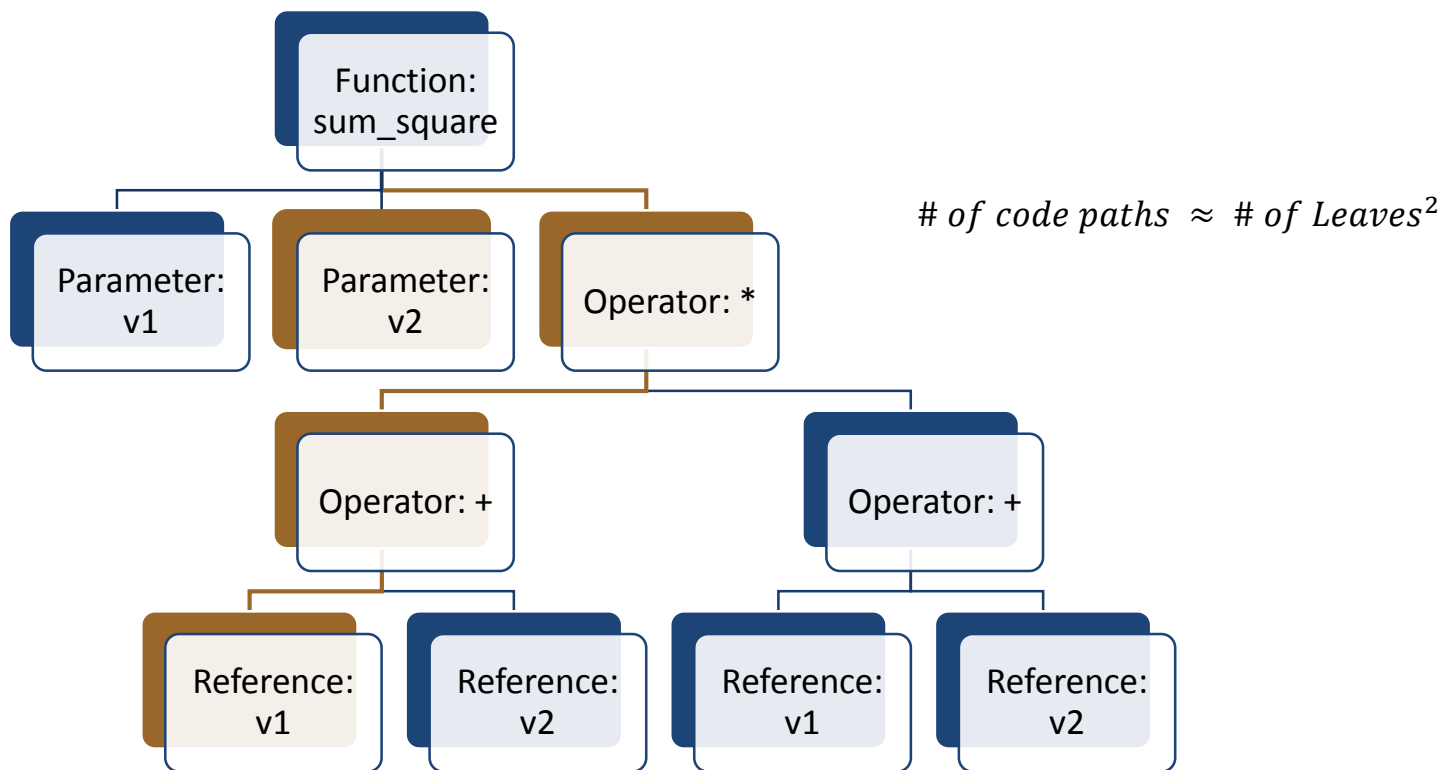


```
int sum_square(int v1, int v2)
{
    return (v1+v2)*(v1+v2);
}
```



Step back (again) – Abstract Syntax Trees

$v1, [(Ref)v1 \wedge (Op)+ \wedge (Op)^* \wedge (Func) _ (Par)v2], v2$



Step back (again) – Abstract Syntax Trees

```
sum|square v1,(PARM_DECL)^(FUNCTION_DECL)_(PARM_DECL),v2
v1,(PARM_DECL)^(FUNCTION_DECL)_(COMPOUND_STMT)_(RETURN_STMT)_(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v1
v1,(PARM_DECL)^(FUNCTION_DECL)_(COMPOUND_STMT)_(RETURN_STMT)_(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v2
v1,(PARM_DECL)^(FUNCTION_DECL)_(COMPOUND_STMT)_(RETURN_STMT)_(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v1
v1,(PARM_DECL)^(FUNCTION_DECL)_(COMPOUND_STMT)_(RETURN_STMT)_(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v2
v2,(PARM_DECL)^(FUNCTION_DECL)_(PARM_DECL),v1
v2,(PARM_DECL)^(FUNCTION_DECL)_(COMPOUND_STMT)_(RETURN_STMT)_(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v1
v2,(PARM_DECL)^(FUNCTION_DECL)_(COMPOUND_STMT)_(RETURN_STMT)_(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v2
v2,(PARM_DECL)^(FUNCTION_DECL)_(COMPOUND_STMT)_(RETURN_STMT)_(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v1
v2,(PARM_DECL)^(FUNCTION_DECL)_(COMPOUND_STMT)_(RETURN_STMT)_(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v2
v1,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)^(RETURN_STMT)^(COMPOUND_STMT)^(FUNCTION_DECL)_(PARM_DECL),v1
v1,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)^(RETURN_STMT)^(COMPOUND_STMT)^(FUNCTION_DECL)_(PARM_DECL),v2
v1,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v2
v1,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v1
v1,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v2
v2,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)^(RETURN_STMT)^(COMPOUND_STMT)^(FUNCTION_DECL)_(PARM_DECL),v1
v2,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)^(RETURN_STMT)^(COMPOUND_STMT)^(FUNCTION_DECL)_(PARM_DECL),v2
v2,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v1
v2,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v1
v2,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v2
v1,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)^(RETURN_STMT)^(COMPOUND_STMT)^(FUNCTION_DECL)_(PARM_DECL),v1
v1,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)^(RETURN_STMT)^(COMPOUND_STMT)^(FUNCTION_DECL)_(PARM_DECL),v2
v1,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v1
v1,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v2
v1,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v2
v2,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)^(RETURN_STMT)^(COMPOUND_STMT)^(FUNCTION_DECL)_(PARM_DECL),v1
v2,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)^(RETURN_STMT)^(COMPOUND_STMT)^(FUNCTION_DECL)_(PARM_DECL),v2
v2,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v1
v2,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)^(PAREN_EXPR)^(BINARY_OPERATOR:*)_(PAREN_EXPR)_(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v2
v2,(DECL_REF_EXPR)^(UNEXPOSED_EXPR)^(BINARY_OPERATOR:+)_(UNEXPOSED_EXPR)_(DECL_REF_EXPR),v1
```

These are the “words” for code2vec

Embeddings – code2vec

MOST SIMILAR ⓘ

login

...is similar to:

PREDICT

- ★ logOut | 70.09%
- authenticate | 66.47%
- connect | 62.24%
- save | 61.99%
- subscribe | 61.67%

COMBINATIONS ⓘ

equals and toLower

...combined, are similar to:

PREDICT

- ★ equalsIgnoreCase | 78.75%
- isUpperCase | 75.82%
- equiv | 75.72%
- sameAs | 75.31%
- isLowerCase | 74.65%

<https://code2vec.org/>

Embeddings – code2vec

The screenshot shows the 'ANALOGIES' interface on the code2vec website. It features a word analogy task: 'receive is to download as... send'. Below the task is a 'PREDICT' button. A dropdown menu displays the following predicted words and their confidence scores:

Word	Confidence Score
upload	76.38%
delete	71.53%
connect	70.51%
install	68.1%
update	67.03%

<https://code2vec.org/>

This slide also left blank intentionally

This is getting out of hand

ML for clean code

Coding conventions are critical for medium-to-large teams

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

Learning Natural Coding Conventions

Miltiadis Allamanis[†]

[†]School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
{m.allamanis, csutton}@ed.ac.uk

Earl T. Barr[‡]

[‡]Dept. of Computer Science
University College London
London, UK
e.barr@ucl.ac.uk

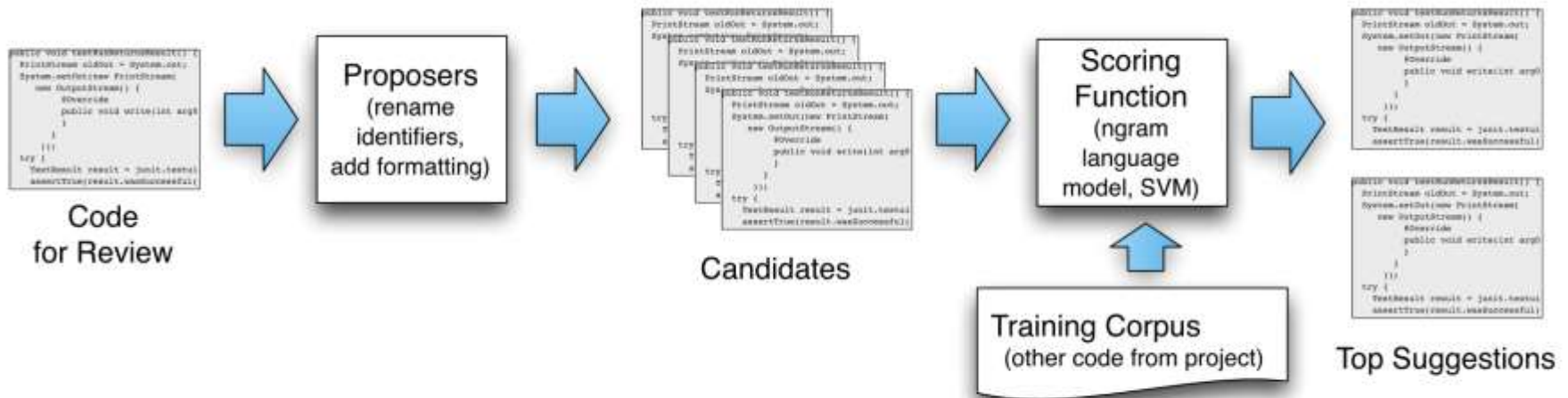
Christian Bird^{*}

^{*}Microsoft Research
Microsoft
Redmond, WA, USA
christian.bird@microsoft.com

Charles Sutton[†]

[†]Microsoft Research
Microsoft
Redmond, WA, USA
christian.bird@microsoft.com

ML for clean code



```
private Test testCaseForClass(Class<?> each) {  
    if (TestCase.class.isAssignableFrom(each)) {  
        return new TestSuite(each.asSubclass(TestCase));  
    } else {  
        return warning(each.getCanonicalName() + "  
    }  
}  
  
/**  
 * Constructs a TestSuite from the given array of classes.  
 *  
 * @param classes the classes to test  
 * @return a TestSuite containing the tests for the given classes  
 */  
public TestSuite testCaseForClasses(Class<?>[] classes) {  
    TestSuite suite = new TestSuite();  
    for (Class<?> c : classes) {  
        suite.addTest(testCaseForClass(c));  
    }  
    return suite;  
}
```

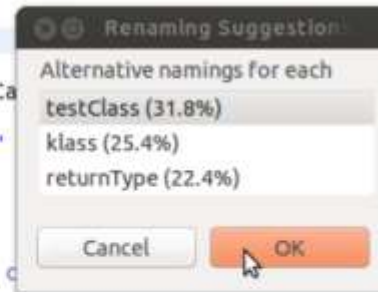


Figure 2: A screenshot of the devstyle Eclipse plugin. The user has requested suggestion for alternate names of the `each` argument.

ML for code security

Find bugs themselves



Automatically write secure code



Create good documentation



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

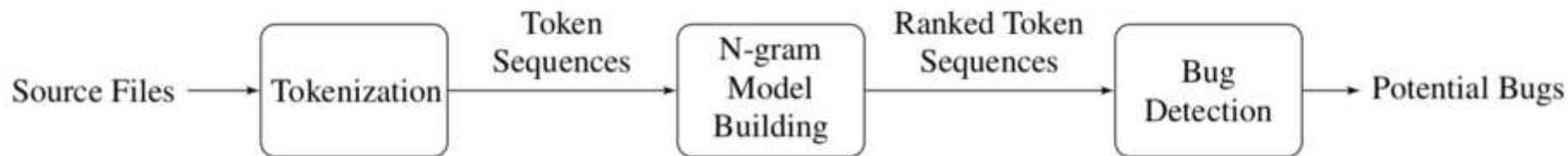


Figure 3: Overview of Bugram

S. Wang, D. Chollak, D. Movshovitz-Attias, and L. Tan, "Bugram: bug detection with n-gram language models," 2016.

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

```
1 if (LOG.isDebugEnabled()) {
2     LOG.debug(indent(depth)+"converting from
3         Pig " + pigType + " " + value +
4         " using " + stringify(schema));
5 }
```

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

```
1 if (LOG.isDebugEnabled()) {
2     LOG.debug(indent(depth)+"converting from
3         Pig " + pigType + " " +
4         toString(value) +
5         " using " + stringify(schema));
6 }
```

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

```
1 | int i = 9;  
2 | if (i == 9) {  
3 |     foo();  
4 |     for (i = 0; i < 10;  
5 |         i++) {  
6 |         bar();  
7 |     }
```

```
1 | int i = 9;  
2 | foo();  
3 | for (i = 0; i < 10; i  
4 |     ++ ) {  
5 |     if (i == 9) {  
6 |     bar();  
7 | }
```

Think back... which techniques would work? Which wouldn't?

S. Wang, T. Liu, and L. Tan, "Automatically learning semantic features for defect prediction," in *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*, 2016, pp. 297–308.

Find bugs themselves

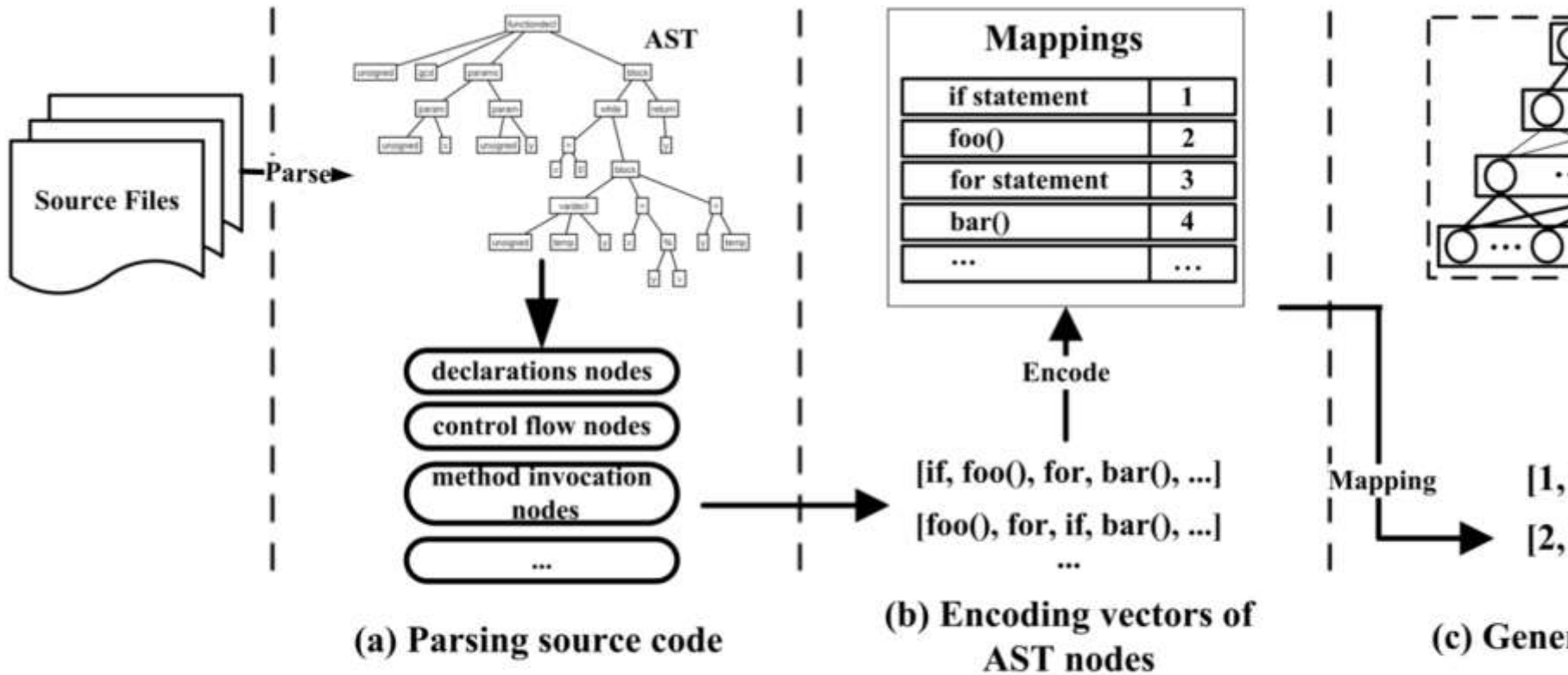


Figure 4: Overview of our proposed DBN-based feature

Code-to-Text – Automated documentation

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

Y. Oda *et al.*, “Learning to Generate Pseudo-Code from Source Code Using Statistical Machine Translation,” in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015, pp. 574–584.

Code-to-Text

“SMT” – Statistical Machine Translation

- Find relationships between tokens in different language models
- Propose many sentences, use statistical models to identify “best”

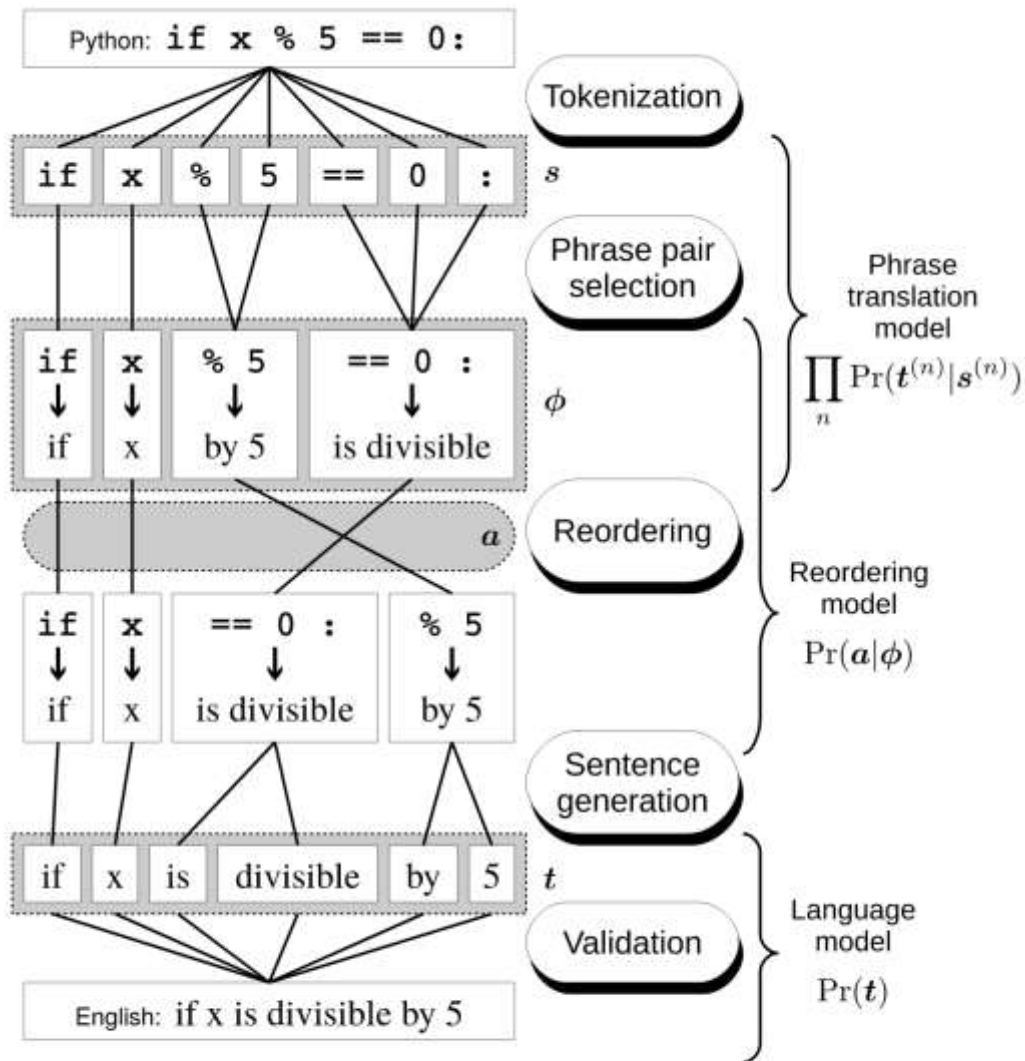


Fig. 2. Example of Python to English PBMT pseudo-code generation.

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, skip-grams
- “Localness” measures
- Embeddings (word2vec, code2vec)

Applications:

- Bug identification
- Code completion
- Documentation generation

Contact Us



Dr. Eliezer Kanal
Technical Manager
CERT Division

412-268-5204

ekanal@sei.cmu.edu

Software Engineering Institute

412-268-5800

888-201-4479

info@sei.cmu.edu

www.sei.cmu.edu