Probabilistic Models of Source Code: An Overview

Miltos Allamanis, University of Edinburgh

November, 2016

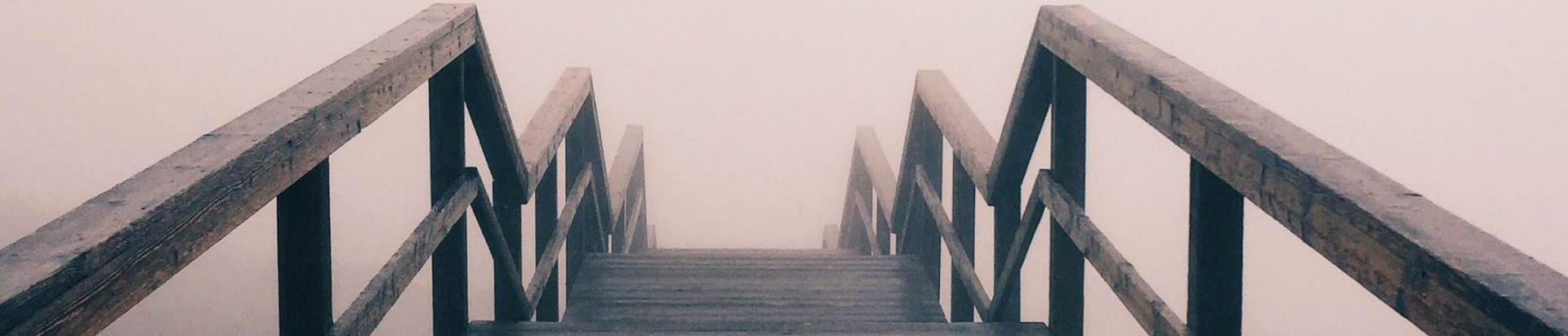
THE UNIVERSITY *of* EDINBURGH
informatics

My PhD is supported by Microsoft® Research

Developers implicitly embed **knowledge** in code that may be useful for the same or other projects.

Mine the hidden knowledge to create **smart** software engineering tools.

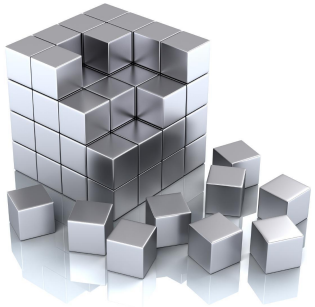
Probabilistic reasoning is a **principled** way of handling **ambiguities** and **partial information**.



Natural Language Processing (NLP)

Some **Knowledge** of **Linguistics**

Data: Corpora of Text, Speech etc



Models of Aspects of Natural Language

Parsing

Named Entity Recognition

Machine Translation

....

- **Resolve** language ambiguities with principled probabilistic models of language.
- Learn models from annotated corpora.

Natural Language Processing with Machine Learning

❯ **Resolve** language ambiguities with principled probabilistic models of language.
❯ Learn models from annotated corpora.

Machine Learning Models of Source Code

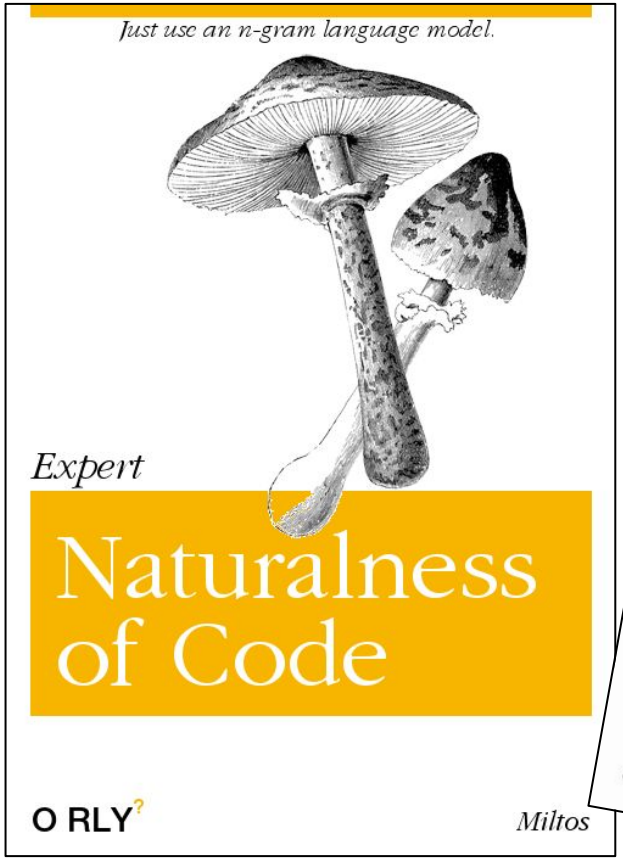
"*All models are wrong, some are useful*" - George Box

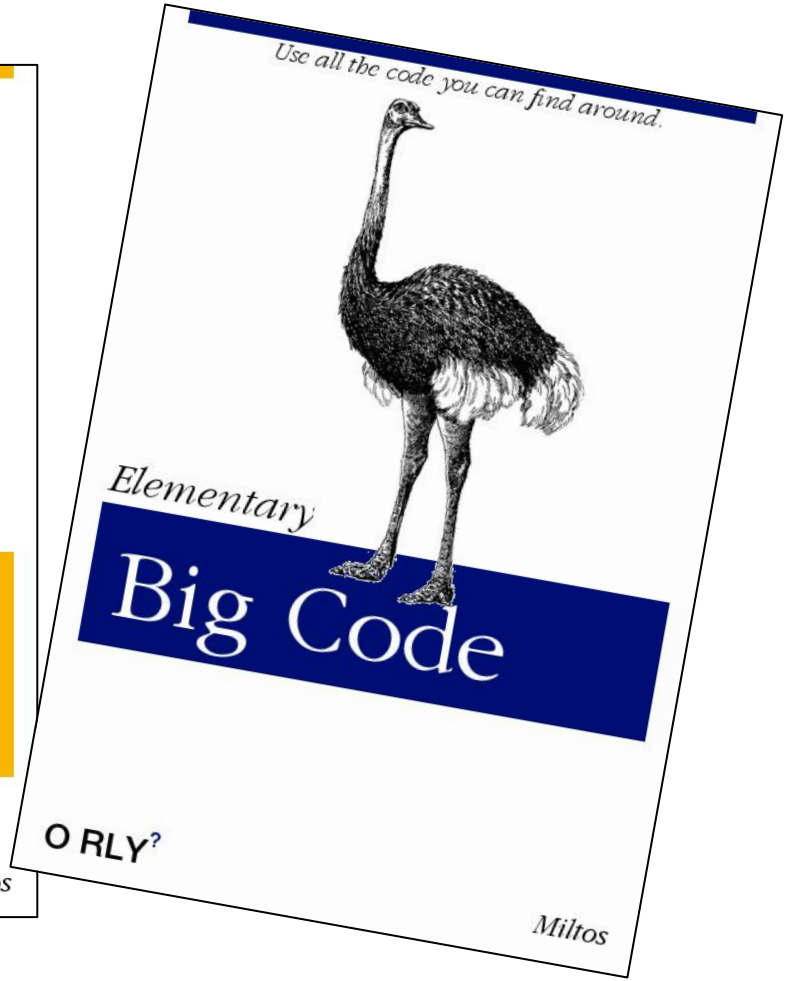
Software Engineers

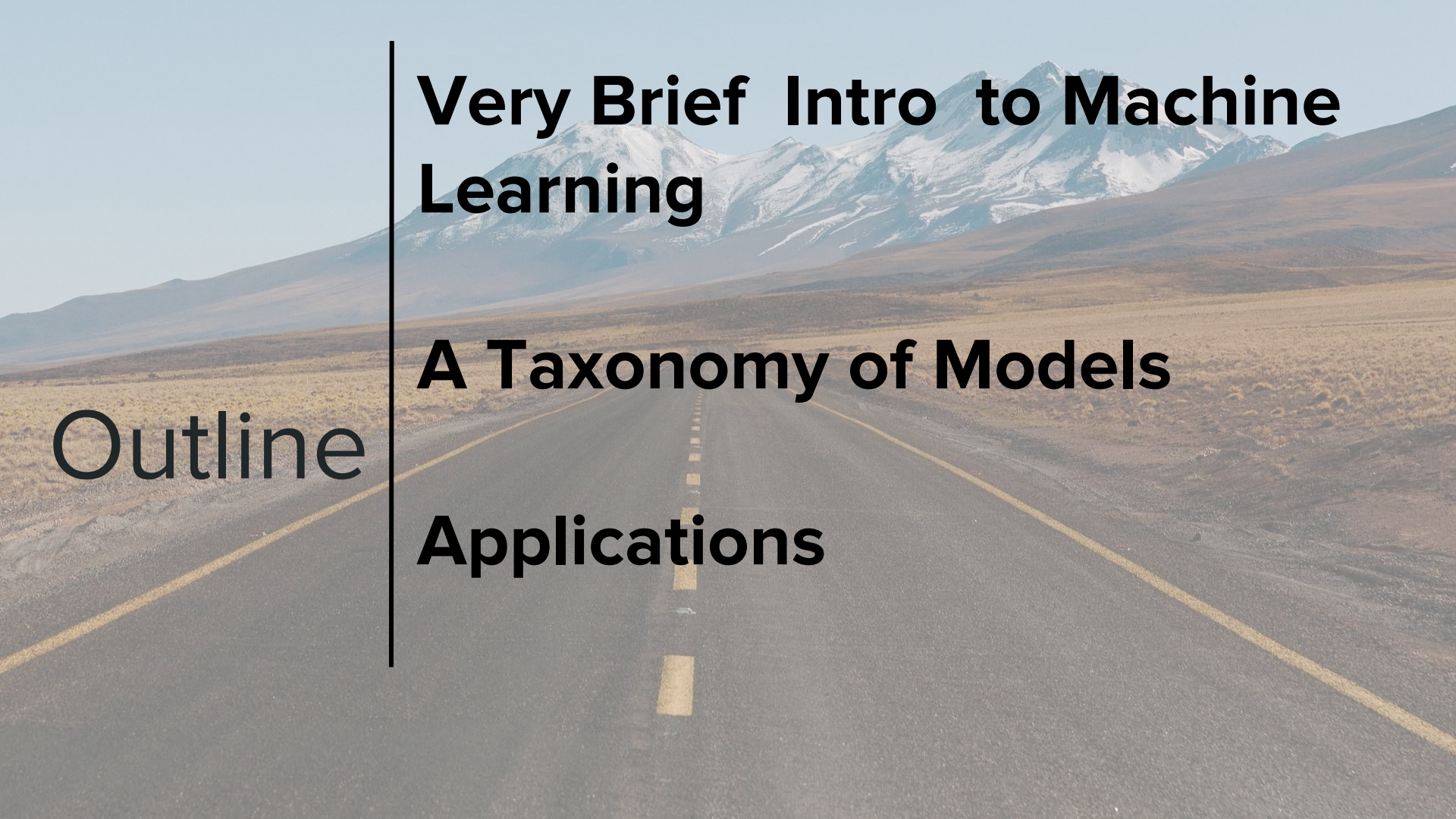
Codebases

Machine Learning Models of **Aspects of Source Code**

Software Engineering Tools


Just use an n-gram language model.
Expert
Naturalness of Code
O RLY?
Miltos


Use all the code you can find around.
Elementary
Big Code
O RLY?
Miltos



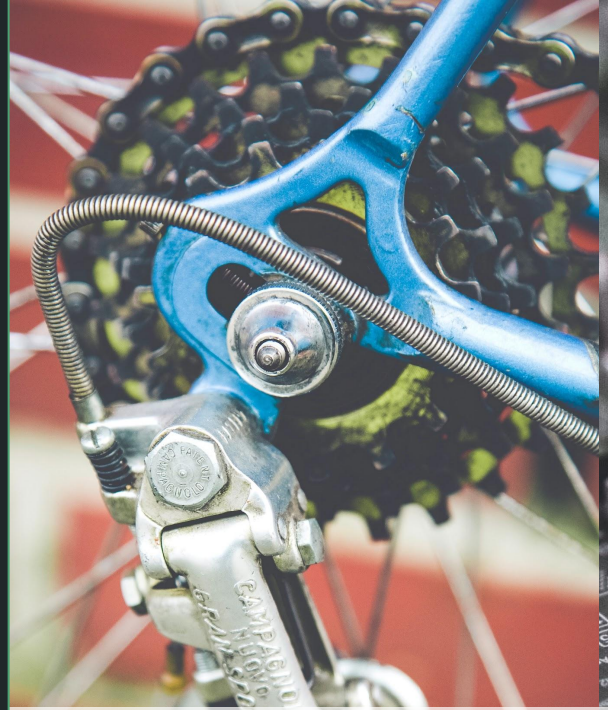
Outline

Very Brief Intro to Machine Learning

A Taxonomy of Models

Applications

Machine **Learning**



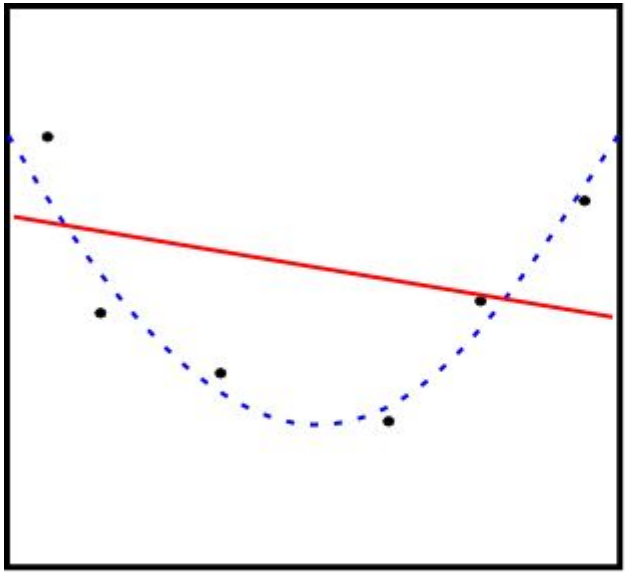
Machine Learning Model

Designed by humans

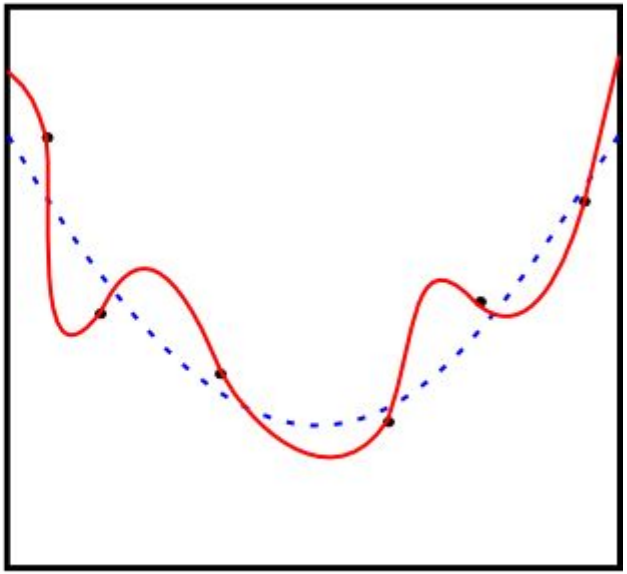
Model Parameters

Learned from data

Finding a good model



Underfitting



Overfitting

graphs from http://antianti.org/?p=175

Learning Model Parameters

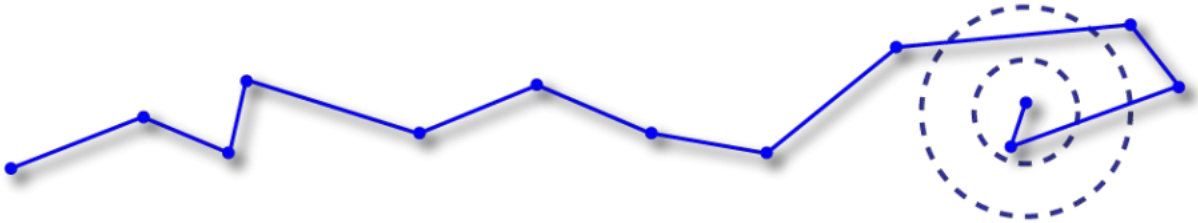
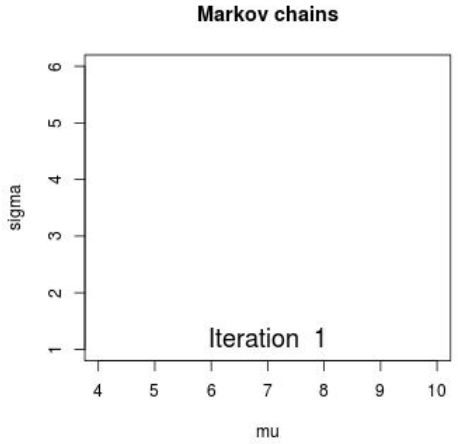


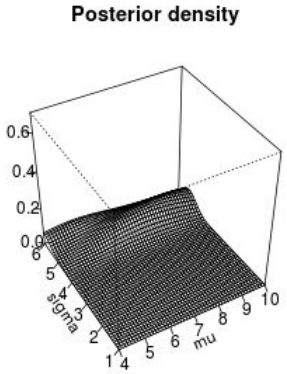
Image from marple.eeb.uconn.edu

- **Optimize** objective function in training set
- Use **computational methods** of optimization

Learning the Distribution of Model Parameters

$$\overbrace{p(\mu \mid Data)}^{\text{posterior}} = \frac{\overbrace{p(Data \mid \mu)}^{\text{likelihood}} \cdot \overbrace{p(\mu)}^{\text{prior}}}{\underbrace{p(Data)}_{\text{marginal likelihood}}}$$

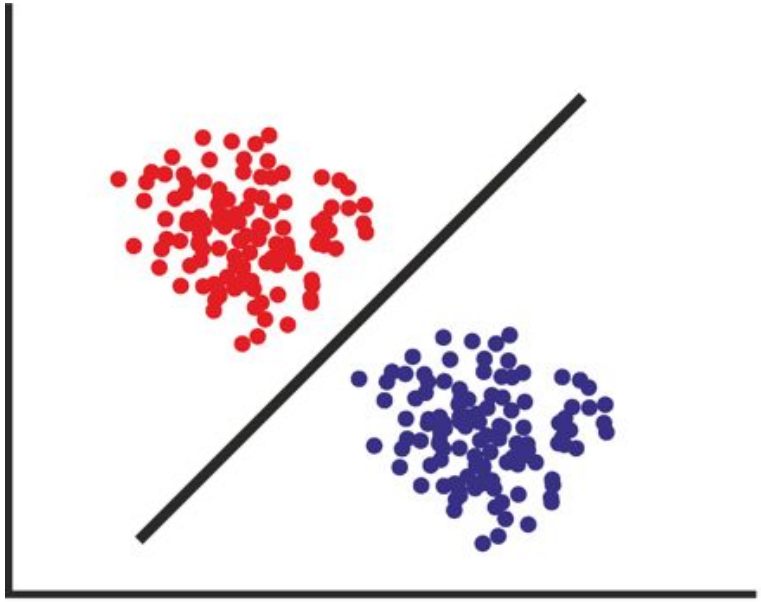




Images from “Bayesian Modelling in Python”

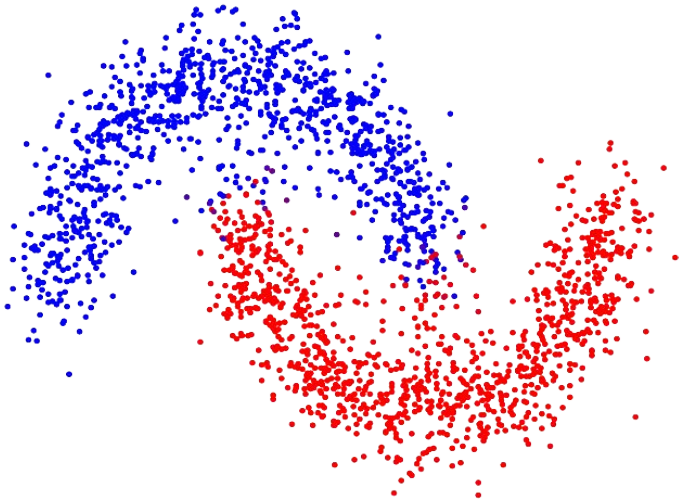
- **Learn** the distribution of parameters given some data.
- Use **computational methods**

A Spectrum of Common Machine Learning Tasks



Classification

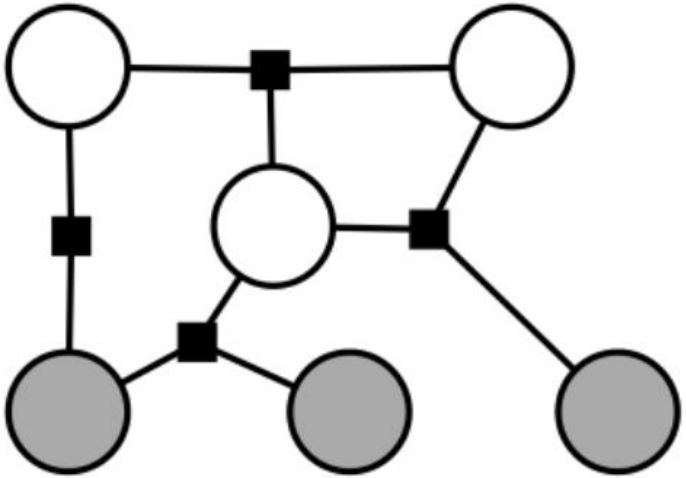
Supervised



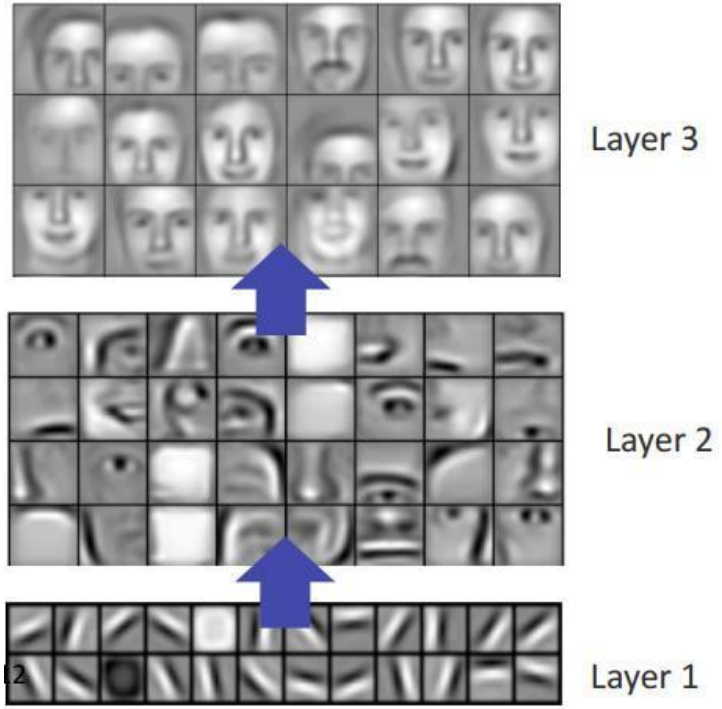
Clustering

Unsupervised

A Spectrum of Common Machine Learning Tasks



Joint Classification



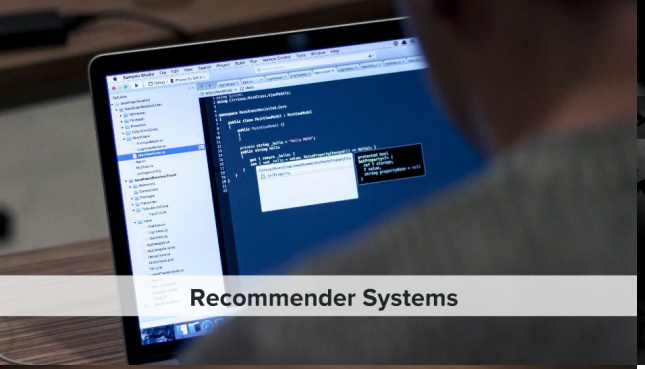
Learning Features

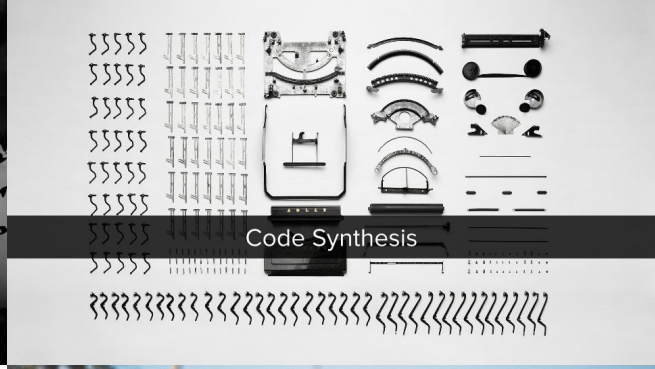
Automatic Evaluation in Machine Learning

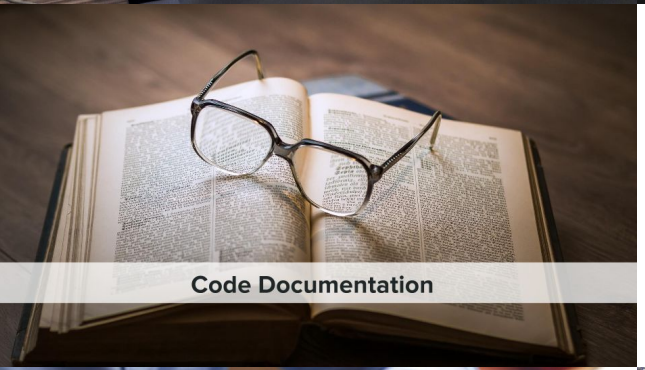
Imperfect measures of performance such as

- Prediction Accuracy
- Model Fit

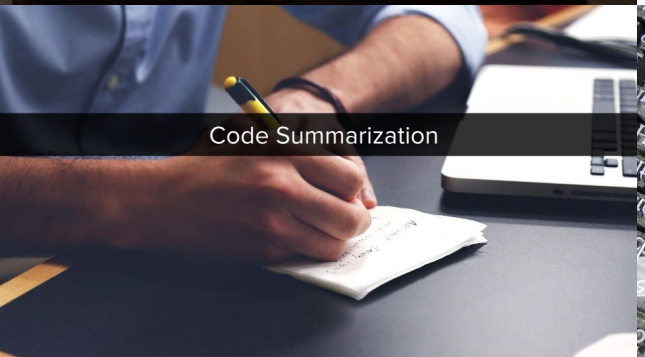
- **Quantify** performance in a **reproducible** manner
- **Drive** improvement of systems in a **measurable** way

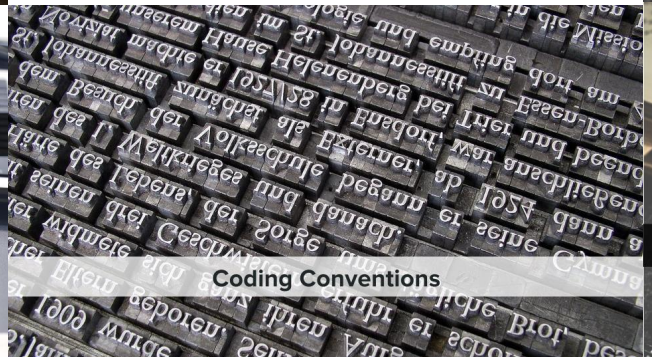


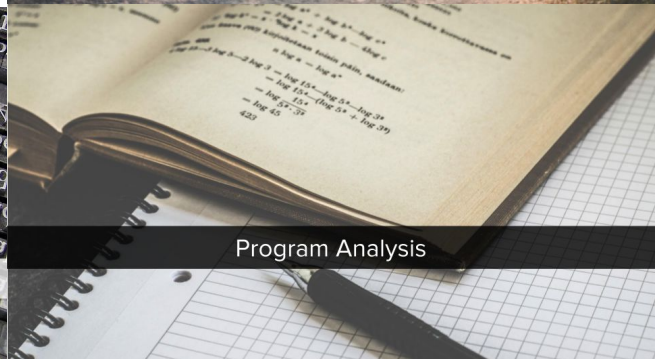


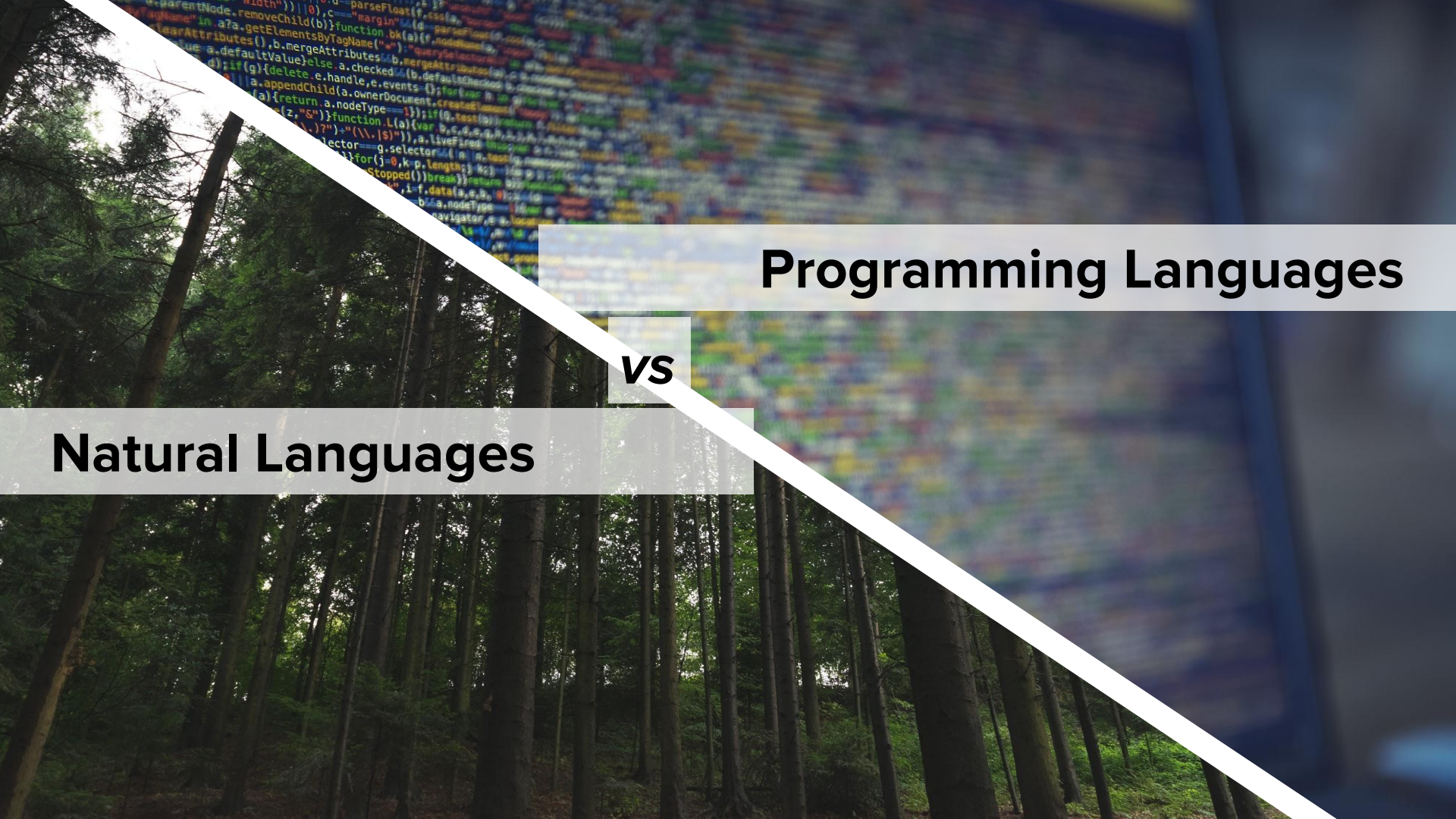


Machine Learning Models








Programming Languages
vs
Natural Languages

Part II

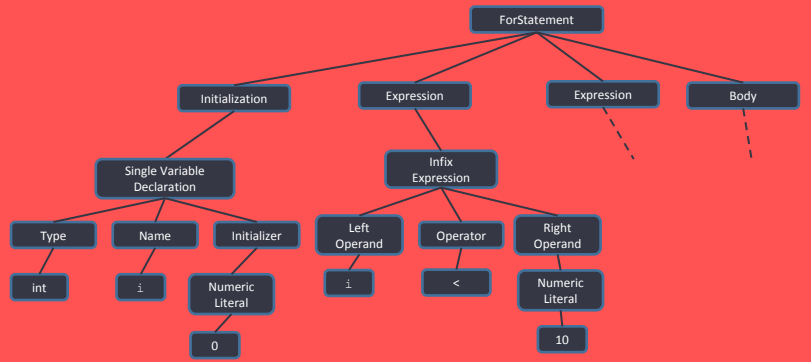
A Taxonomy of Models

By Code Representations

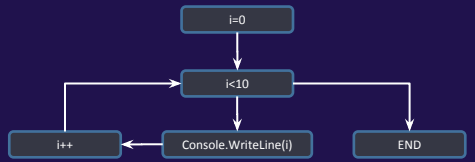
Token

```
for (int i = 0; i < 10; i++){
    Console.WriteLine(i);
}
```

Syntax

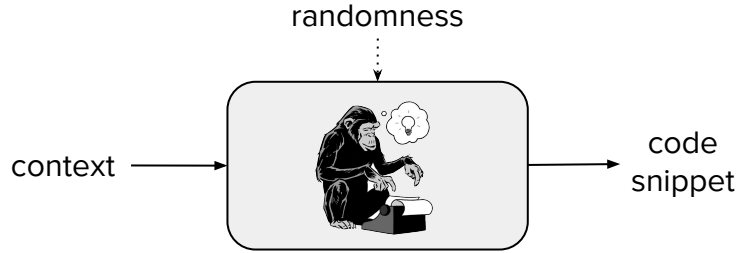


Graph



Probabilistic Models of Source Code

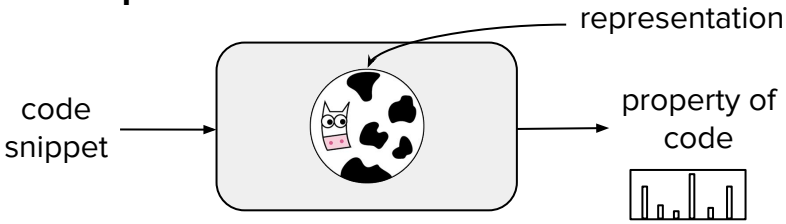
Code Generating Models



$$P_{\mathcal{D}}(\mathbb{c} | C(\mathbb{c}))$$

- Language Models
- Translation Models
- Multimodal Models

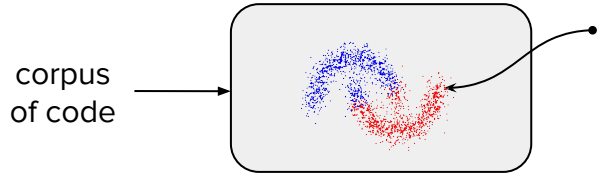
Code Representational Models



$$P_{\mathcal{D}}(\pi | f(\mathbb{c}))$$

- Structured Prediction
- Distributed Representations

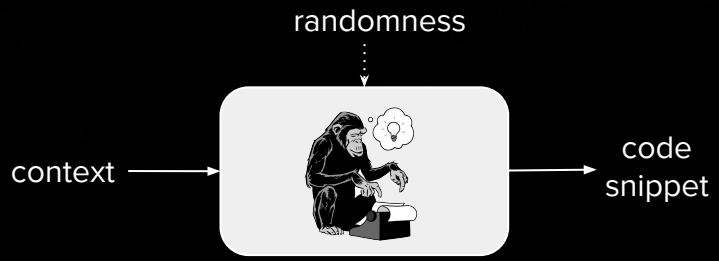
Pattern Mining Models



$$P_{\mathcal{D}}(f(\mathbb{c})) = \sum_{\mathbf{l}} P_{\mathcal{D}}(f(\mathbb{c}) | \mathbf{l}) P(\mathbf{l})$$

- Latent Variable Models

Code Generating Models



$$P_{\mathcal{D}}(\mathbb{c}|C(\mathbb{c}))$$

code snippet

context

Simplifying assumptions about *how* code is generated

Language Models

Assign a non-zero probability to every piece of valid code

$$P(\mathbb{C})$$

n-gram Language Models

```
public void execute(Runnable task) {
        if (task == null)
            throw new NullPointerException();
        ForkJoinTask<?> job;
        if (task instanceof ForkJoinTask<?>) // avoid re-wrap
            job = (ForkJoinTask<?>) task;
        else
            job = new ForkJoinTask.AdaptedRunnableAction(task);
        externalPush(job);
    }
```

n-gram Language Models

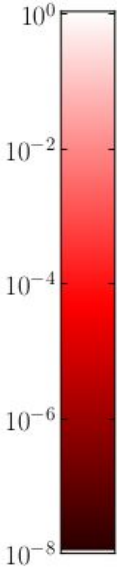
```
public void execute(          task) {
        if (task == null)
            throw new NullPointerException();
        ForkJoinTask<?> job;
        if (task instanceof ForkJoinTask<?>) // avoid re-wrap
            job = (ForkJoinTask<?>) task;
        else
            job = new ForkJoinTask.AdaptedRunnableAction(task);
        externalPush(job);
    }
```

n-gram Language Models

```
public void execute(                    task) {
        if (task == null)
            throw new NullPointerException();
        ForkJoinTask<?> job;
        if (task instanceof ForkJoinTask<?>) // avoid re-wrap
            job = (ForkJoinTask<?>) task;
        else
            job = new ForkJoinTask.AdaptedRunnableAction(task);
        externalPush(job);
    }
```

$$P(t_0 \dots t_M) = \prod_{m=0}^{M} P(t_m | t_{m-1} \dots t_{m-n+1})$$

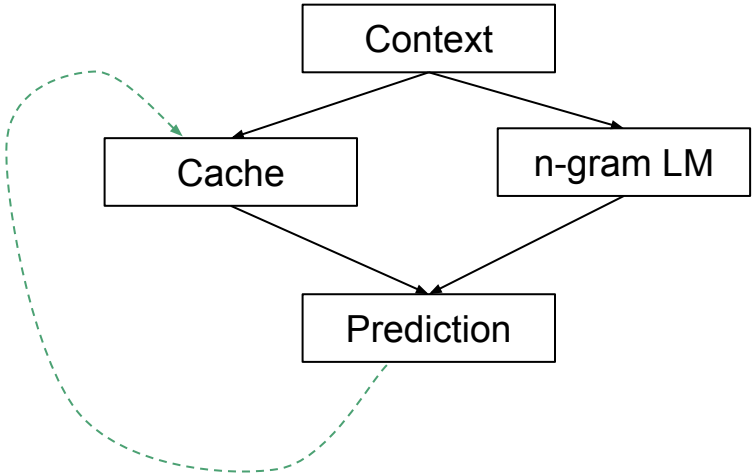
n-gram Language Models




```
1  public void execute(Runnable task) {
2      if (task == null)
3          throw new NullPointerException();
4      ForkJoinTask<?> job;
5      if (task instanceof ForkJoinTask<?>) // avoid re-wrap
6          job = (ForkJoinTask<?>) task;
7      else
8          job = new ForkJoinTask.AdaptedRunnableAction(task);
9      doSubmit(job);
10 }
```

Image from Allamanis & Sutton, MSR 2013

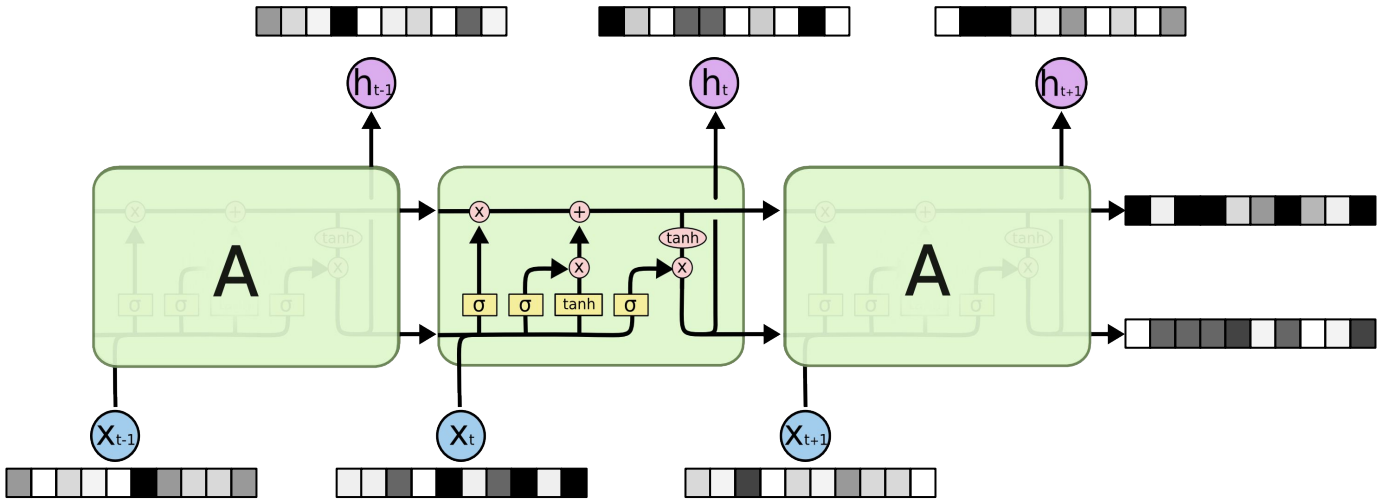
Cache *n*-gram Language Models



Tu *et al.* 2014

$$P(t_i|h, cache) = \lambda \cdot P_{n\text{-gram}}(t_i|h) + (1 - \lambda) \cdot P_{cache}(t_i|h)$$

Neural Language Models

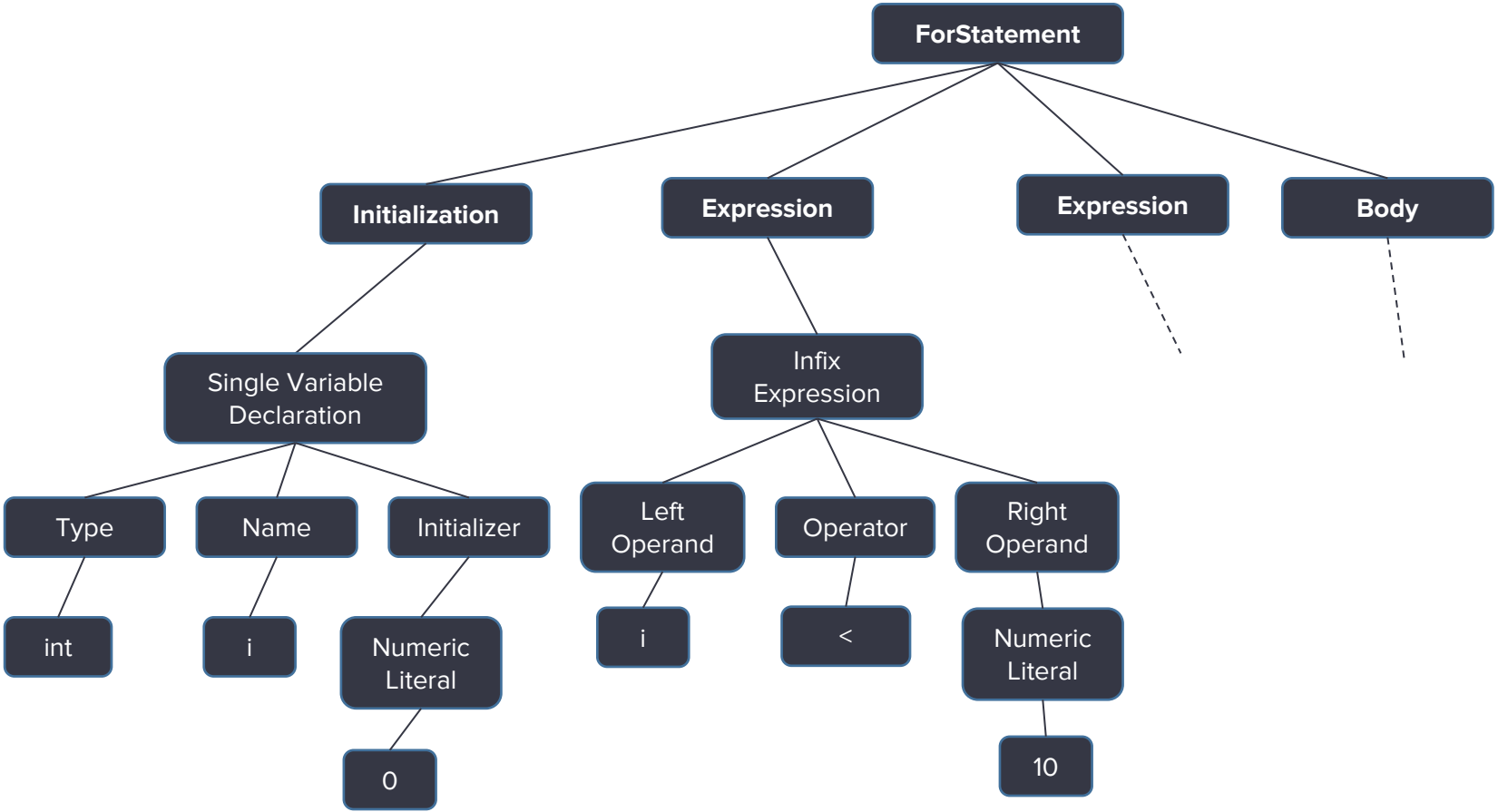



```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
  int i;
  if (classes[class]) {
    for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
      if (mask[i] & classes[class][i])
        return 0;
  }
  return 1;
}
```

Image from colah.github.io, Karpathy Andrej

Karpathy *et al.* 2013

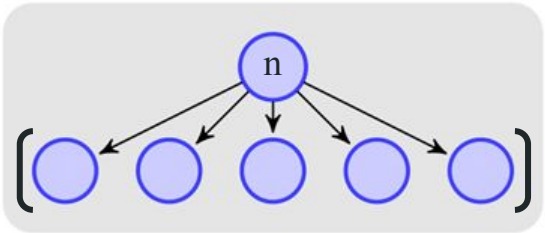
White *et al.* 2015



Syntactic model of source code, *i.e.* model how AST is generated

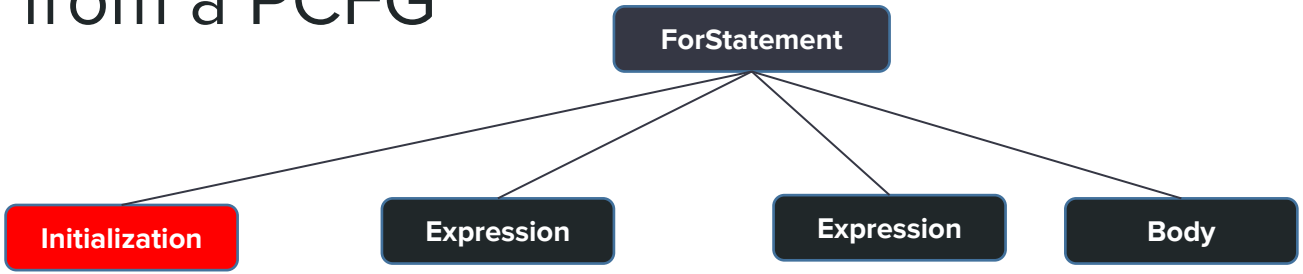
Tree Generation Model:
Probabilistic Context Free Grammars (PCFG)

$E \to E + E$	(prob 0.7)	$T \to F * F$	(prob 0.6)
$E \to T$	(prob 0.3)	$T \to F$	(prob 0.4)
$F \to (E)$	(prob 0.1)	$F \to id$	(prob 0.9)

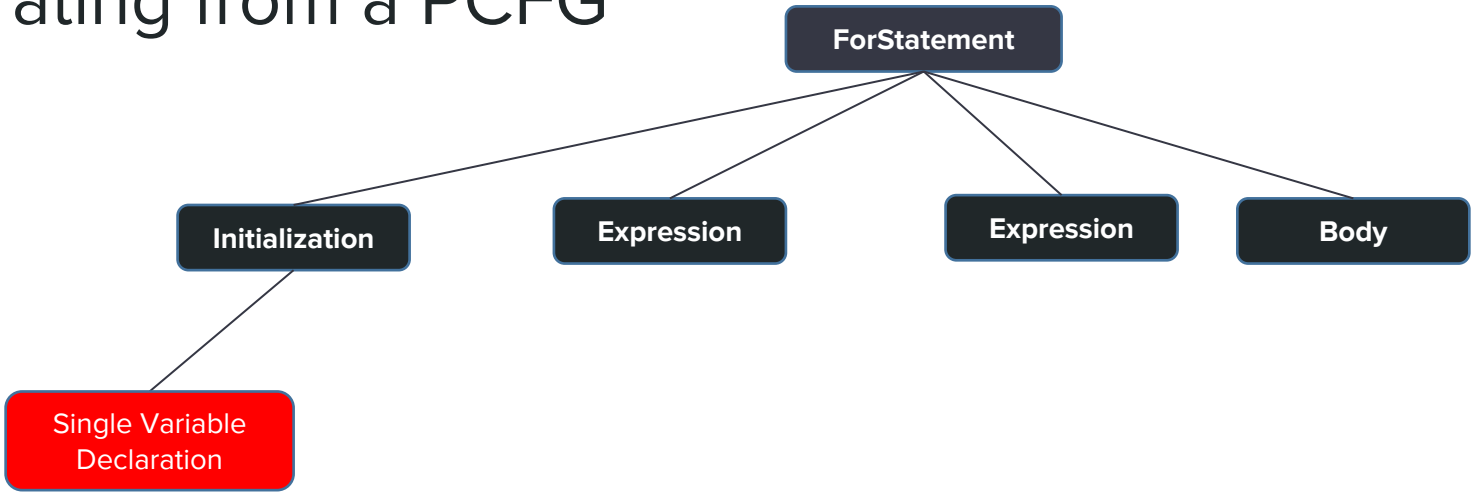


Generating from a PCFG

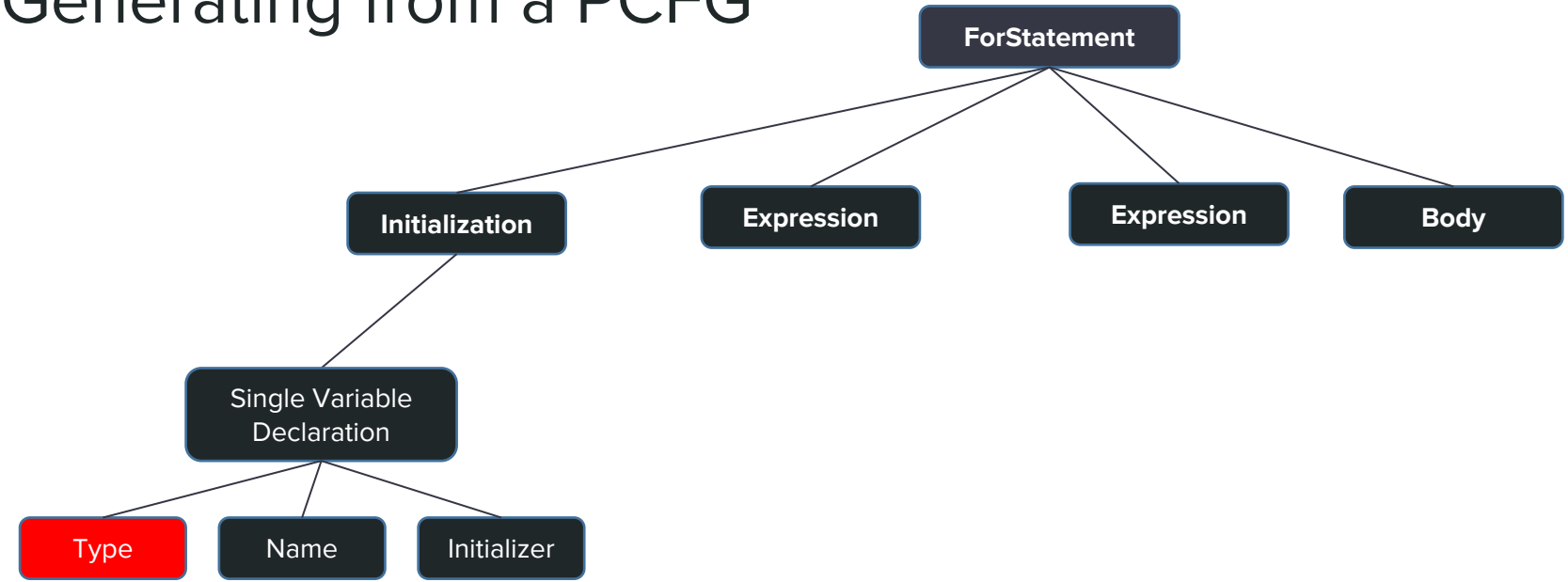
Generating from a PCFG



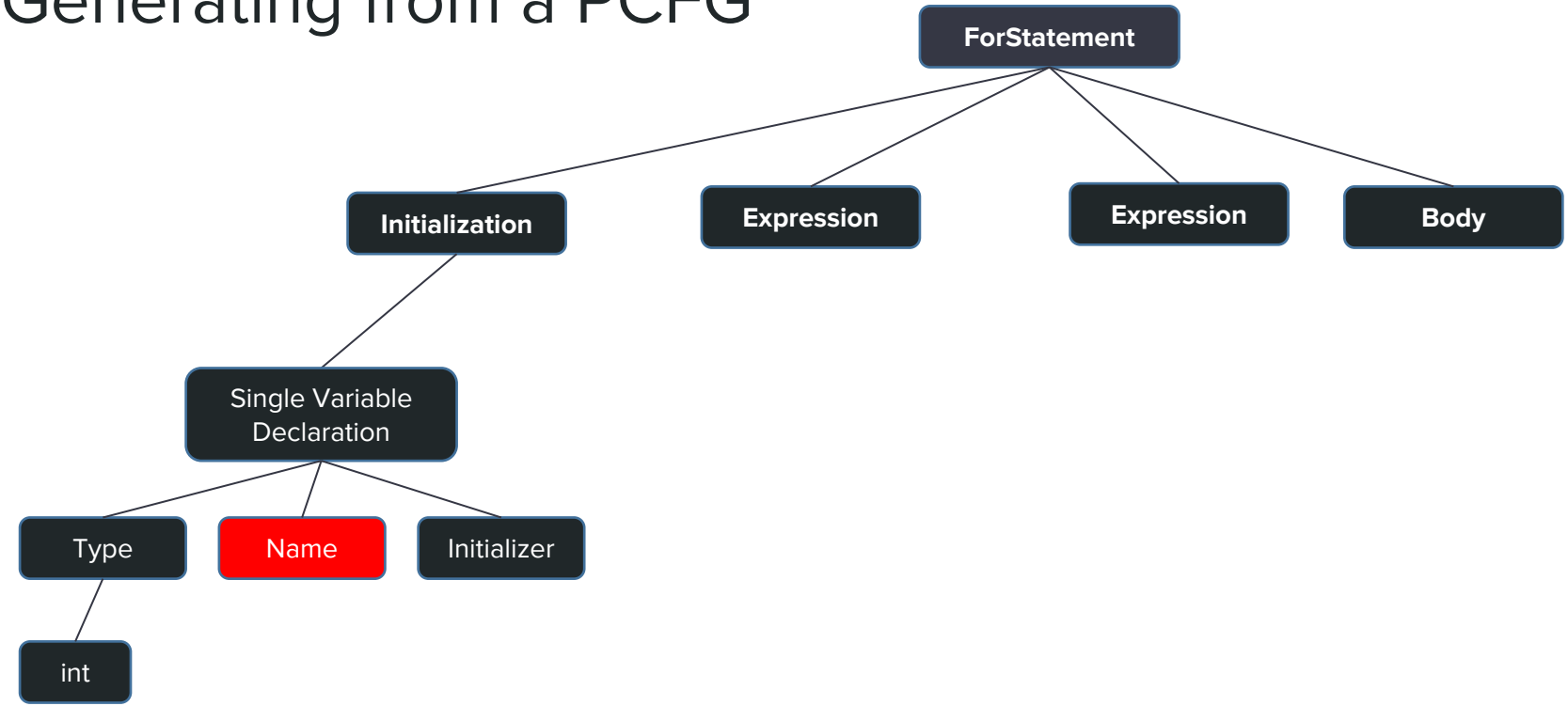
Generating from a PCFG



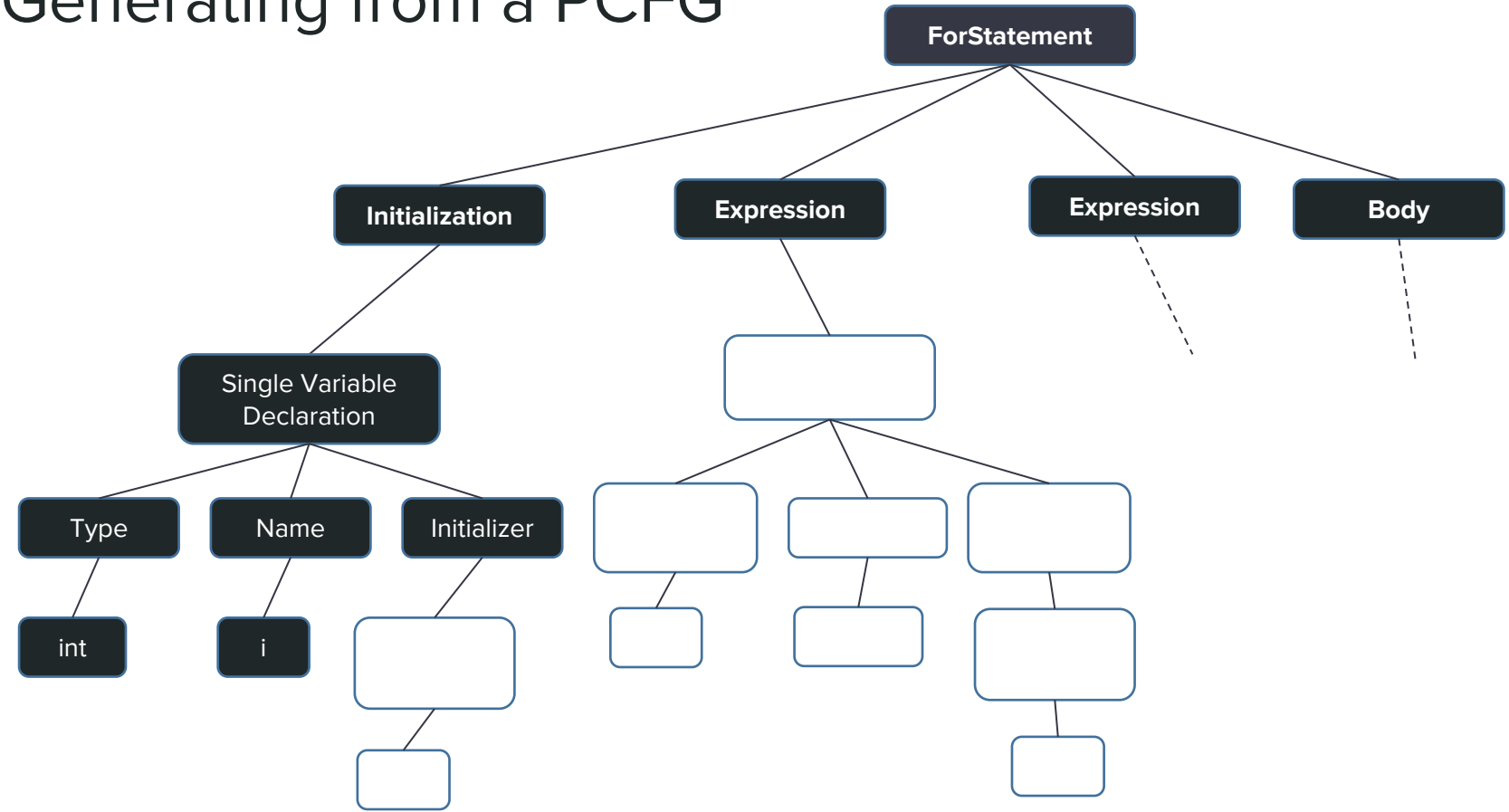
Generating from a PCFG

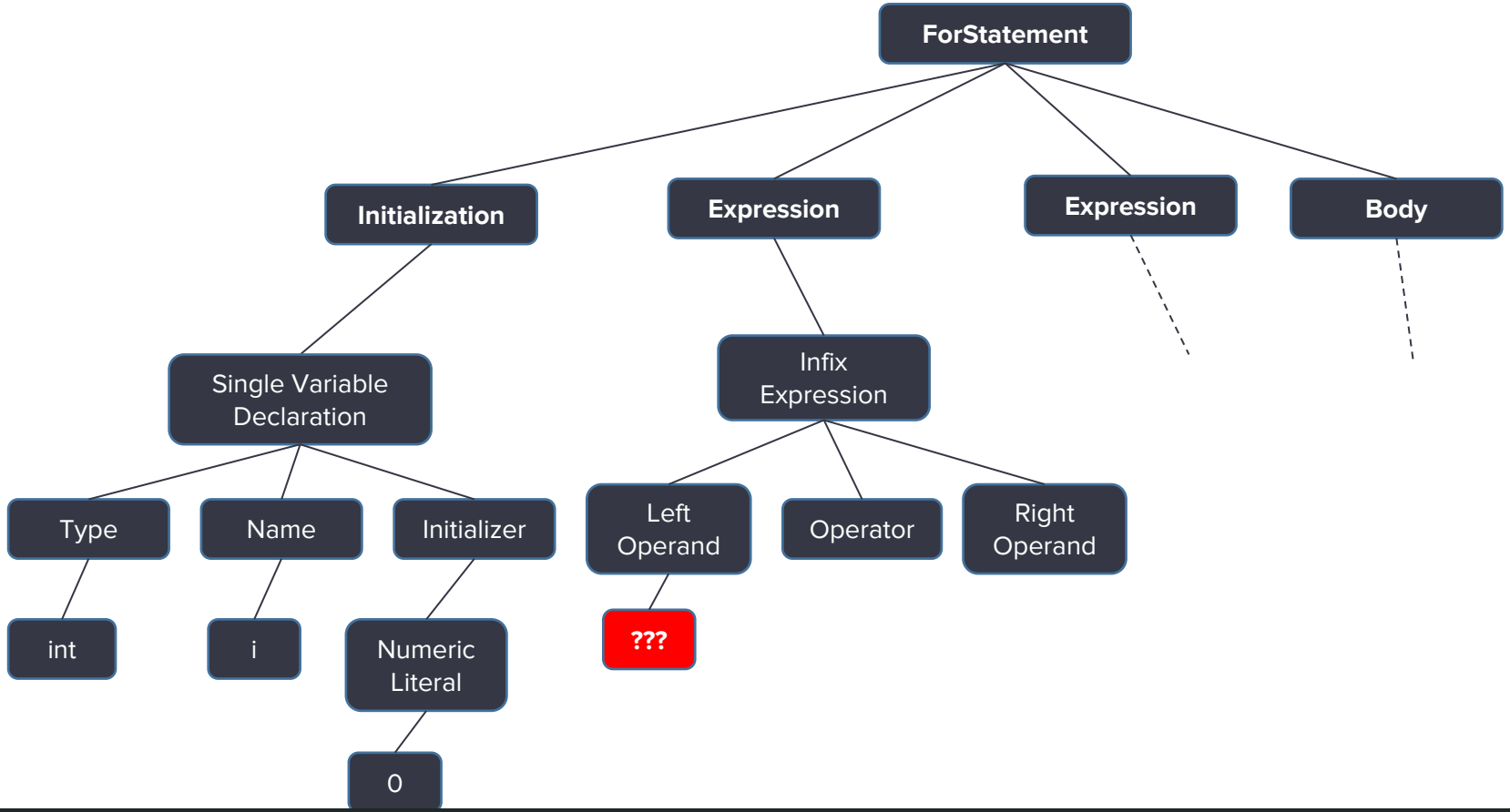


Generating from a PCFG

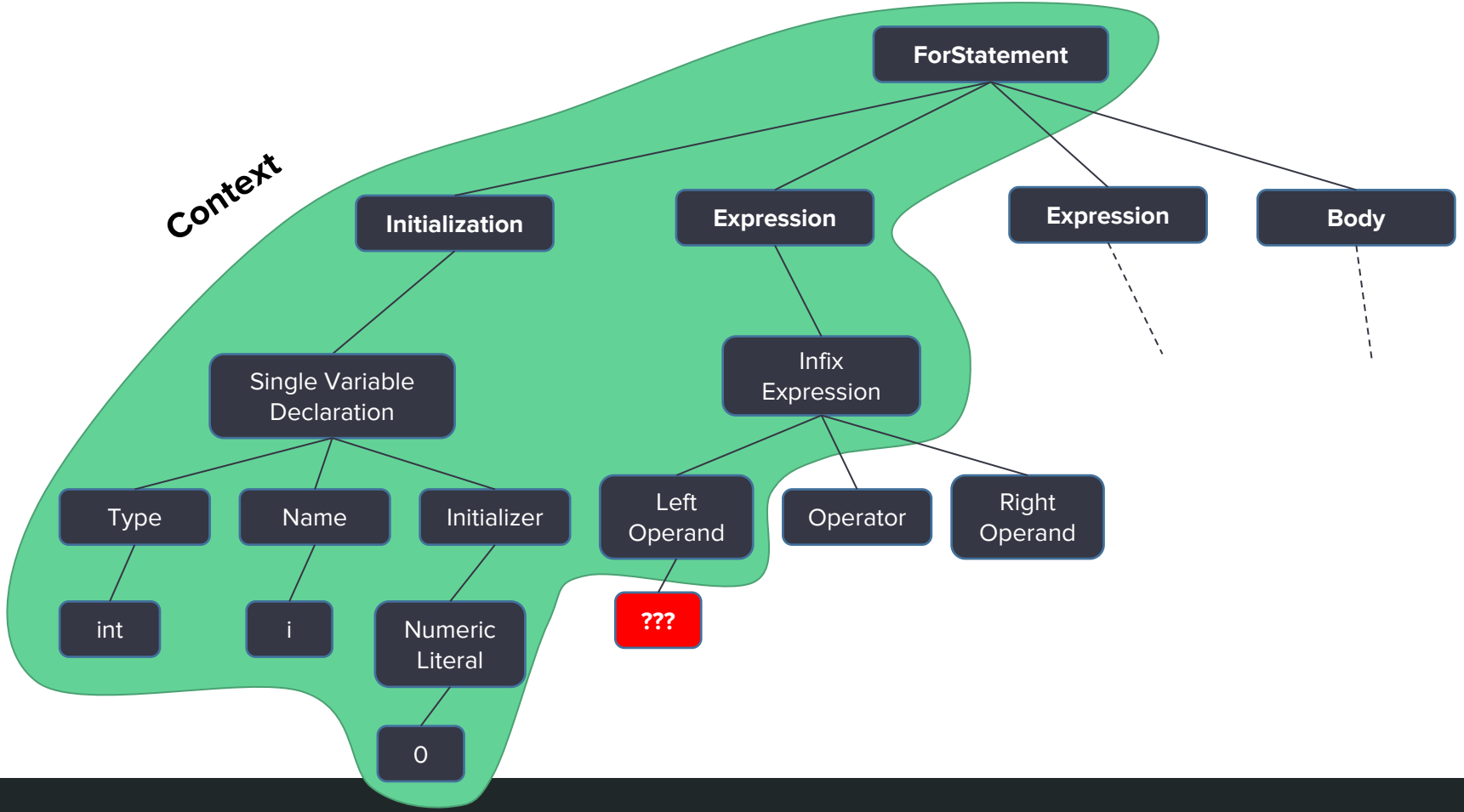


Generating from a PCFG



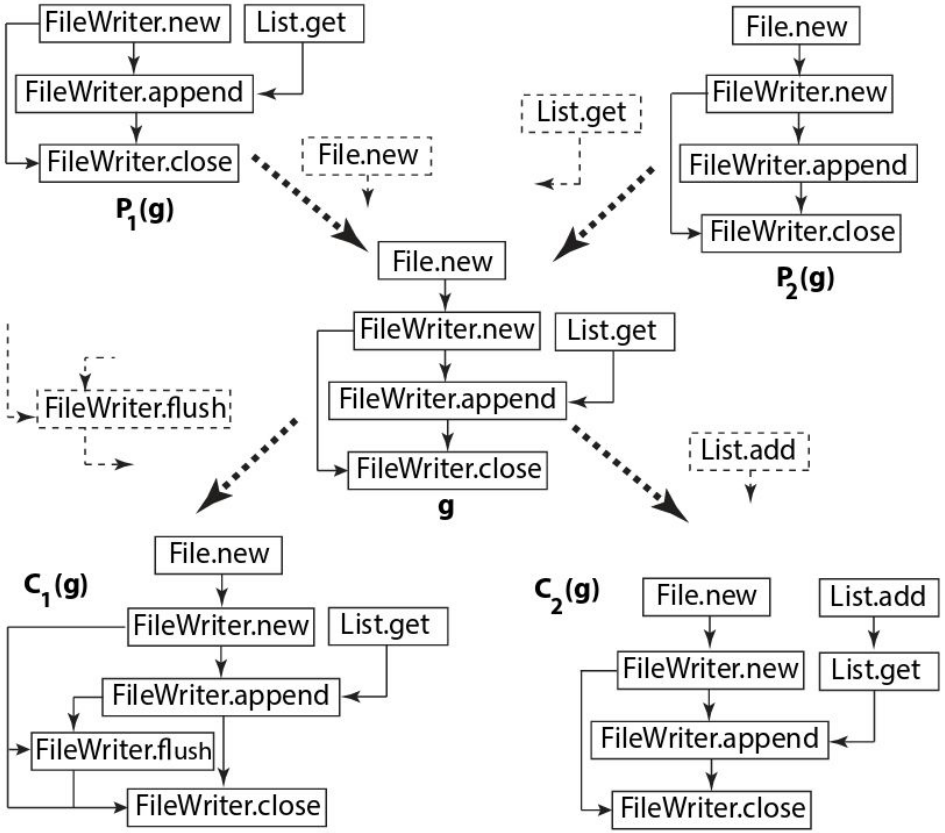


Maddison & Tarlow, ICML 2014 Raychev *et al.* OOPSLA 2016, POPL 2016



Maddison & Tarlow, ICML 2014 Raychev *et al.* OOPSLA 2016, POPL 2016

Graph Language Models



Nguyen & Nguyen, ICSE 2015

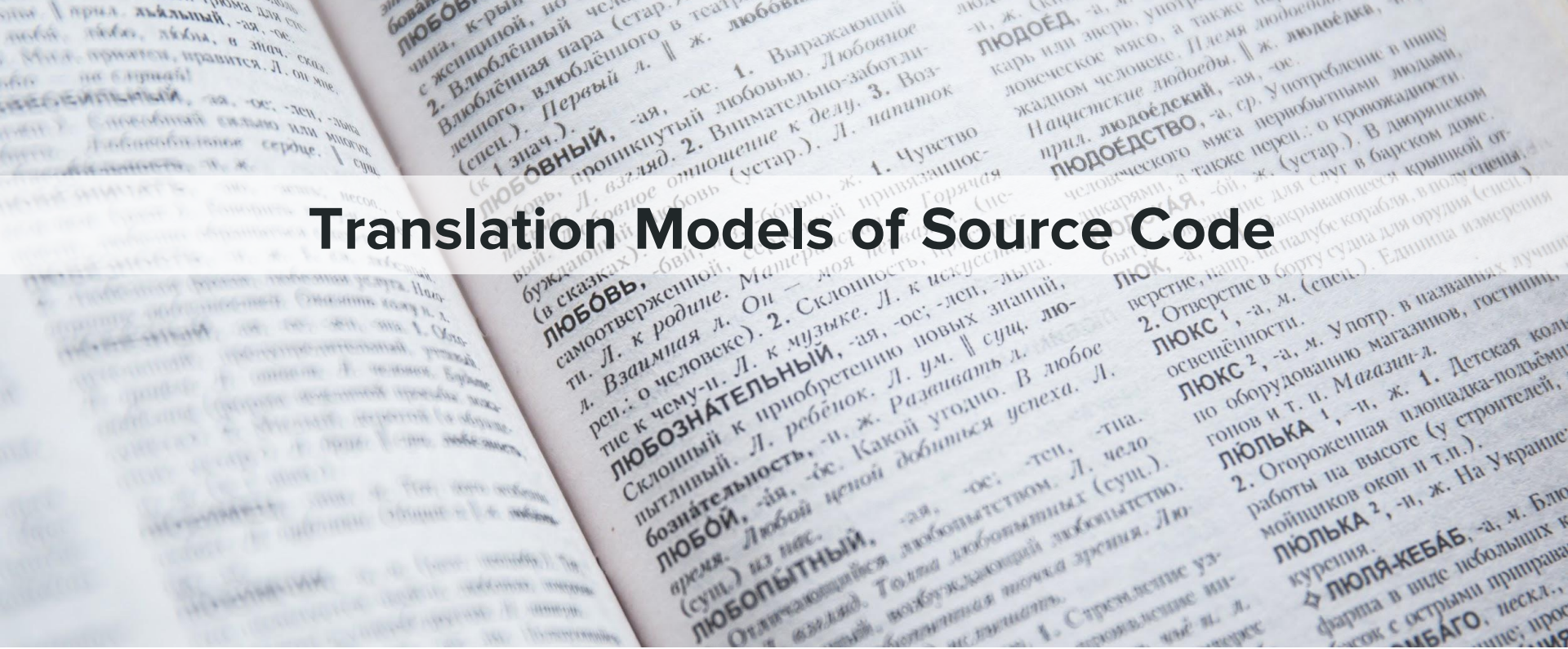
Evaluation Metrics for Language Models

Log Probability

$$Q(\mathbb{c}, P_{\mathcal{D}}) = -\log_2 \left(P_{\mathcal{D}}(\mathbb{c})\right)$$

Cross Entropy

$$H(\mathbb{c}, P_{\mathcal{D}}) = -\frac{1}{M} \log_2 P_{\mathcal{D}}(\mathbb{c})$$

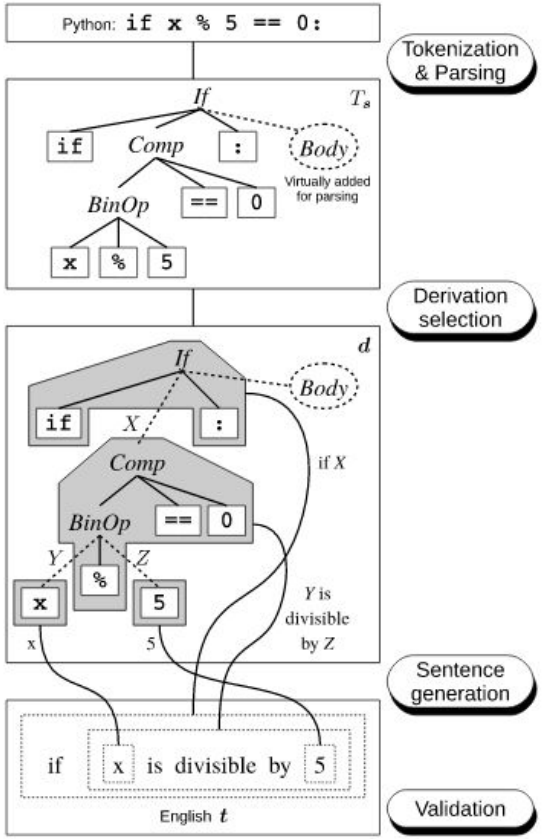


Translation Models of Source Code

$$\mathbb{t}^* = \arg\max P_{\mathcal{D}}(\mathbb{t}|\mathbb{s}) = \arg\max P_{\mathcal{D}}(\mathbb{s}|\mathbb{t}) P_{\mathcal{D}}(\mathbb{t})$$

Language Model

Subphase	Output	Example output
Parallel data collection	Method pairs	**C#:** `Console . WriteLine ( "Hello World!" ) ;` **Java:** `System . out . println ( "Hello World!" ) ;`
Word alignment	Aligned method pairs	**C#:** `Console . WriteLine ( "Hello World!" ) ;` **Java:** `System . out . println ( "Hello World!" ) ;`
Phrase table construction	Phrase table	**C# phrase** / **Java phrase** / **Score** i.e. Pr(**C# phrase** | **Java phrase**) `Console` / `System . out` / 0.8 `WriteLine` / `println` / 0.5 `.` / `.` / 0.7 `(` / `(` / 0.9 ... / ... / ...



Raychev *et al.* 2014 Nguyen *et al.* 2013,2015 Oda *et al.* ASE 2015

Multimodal Models of Source Code

"*get the first letter of each word in string and uppercase*"

$$P_{\mathcal{D}}(\mathbb{C}|\mathbf{m})$$

```
string s;
string[] words = s.ToUpper().split(‘ ‘);
string[] firstLetters = new string[words.Length];
for (int i=0; i < words.Length; i++) {
       firstLetters[i] = words.Substring(0,1);
}
```

Non-Code Modality

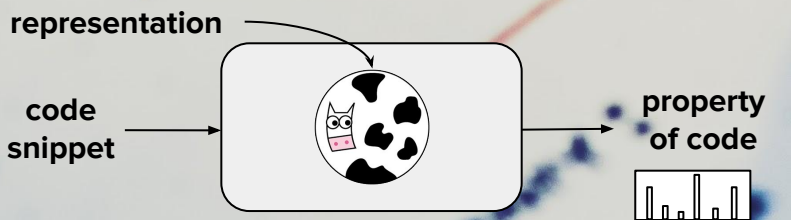
Synthesize/Score Code Snippet

- all elements to small letters for each line and sort
- all elements to small letters for each line and order
- all elements to lowercase for each line and sort
- all elements to lowercase for each line and order
- all elements lowercase for each line and sort
- all elements lowercase for each line and order
- all elements to lower case for each line and sort
- all elements to lower case for each line and order
- all elements to small letters for each new line and sort

```
string result = String.Join("\\n",input_string.Split(\'\\n\')
        .Select((string x) => x.ToLower()).OrderBy(x => x));
```

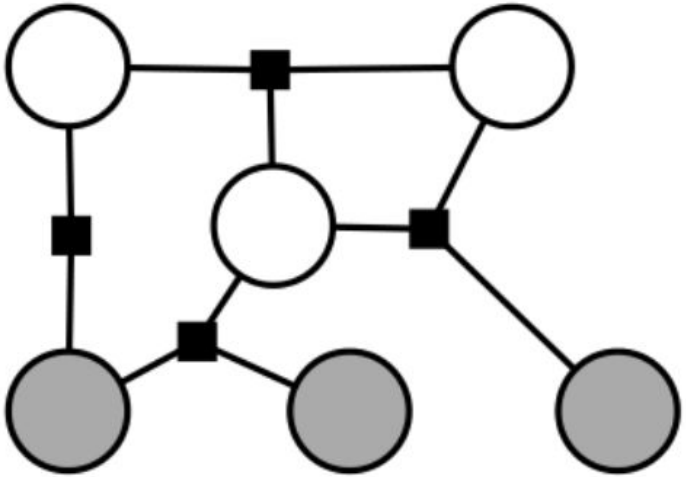
Allamanis *et al.* “Bimodal modelling of source code and natural language” ICML 2015

Representational Models of Code



$$P_{\mathcal{D}}(\pi|f(\mathbb{c}))$$

Structured Prediction



$$P_{\mathcal{D}}(\pi | f(\mathbb{c}))$$

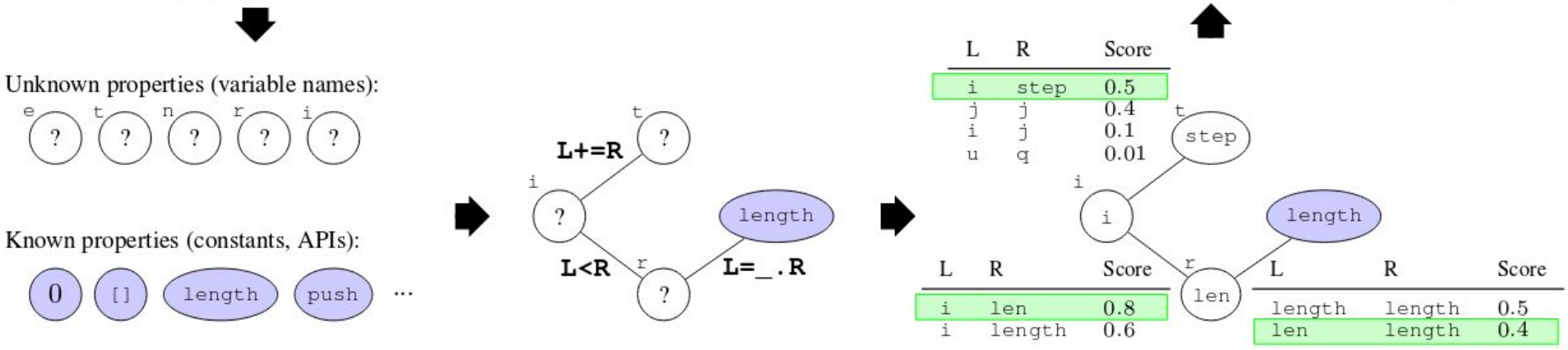
- Output variables are mutually dependent or constrained.

```
function chunkData(e, t) {
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t) {
    if (i + t < r) {
      n.push(e.substring(i, i + t));
    } else {
      n.push(e.substring(i, r));
    }
  }
  return n;
}
```

(a) JavaScript program with minified identifier names

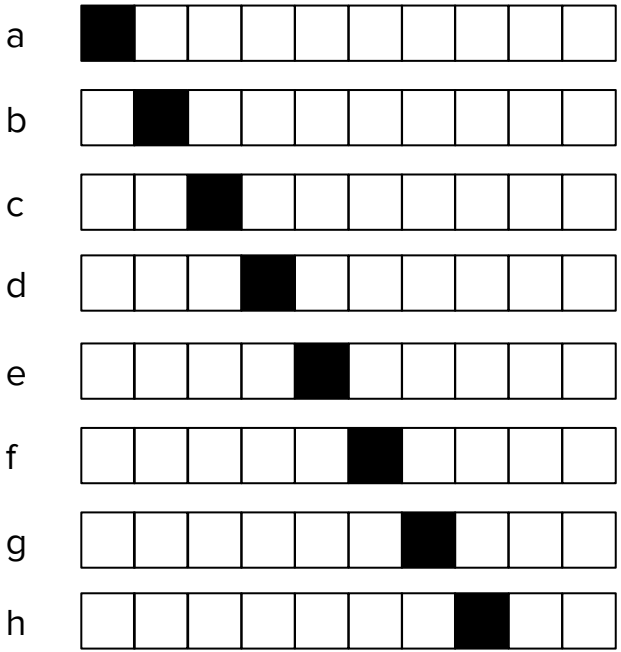
```
/* str: string, step: number, return: Array */
function chunkData(str, step) {
  var colNames = []; /* colNames: Array */
  var len = str.length;
  var i = 0; /* i: number */
  for (; i < len; i += step) {
    if (i + step < len) {
      colNames.push(str.substring(i, i + step));
    } else {
      colNames.push(str.substring(i, len));
    }
  }
  return colNames;
}
```

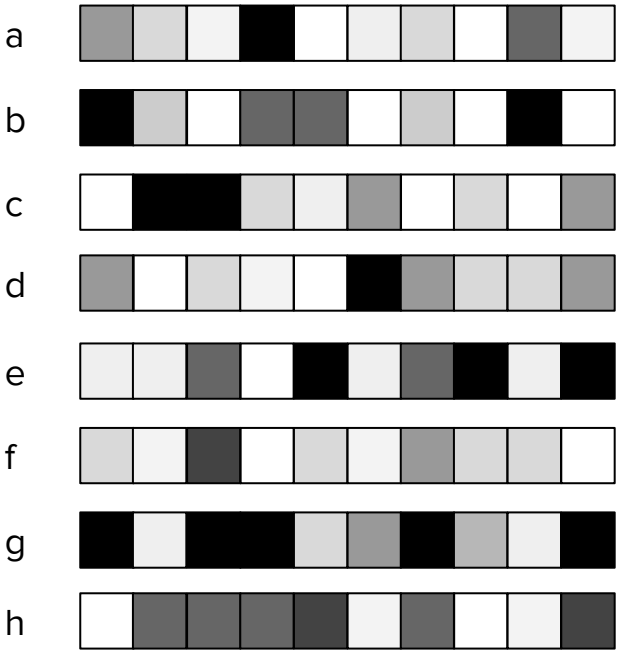
(e) JavaScript program with new identifier names and types



Raychev *et al.* “Predicting Program Properties from Big Code”, 2015

Localized *vs* Distributed (Vector) Representations



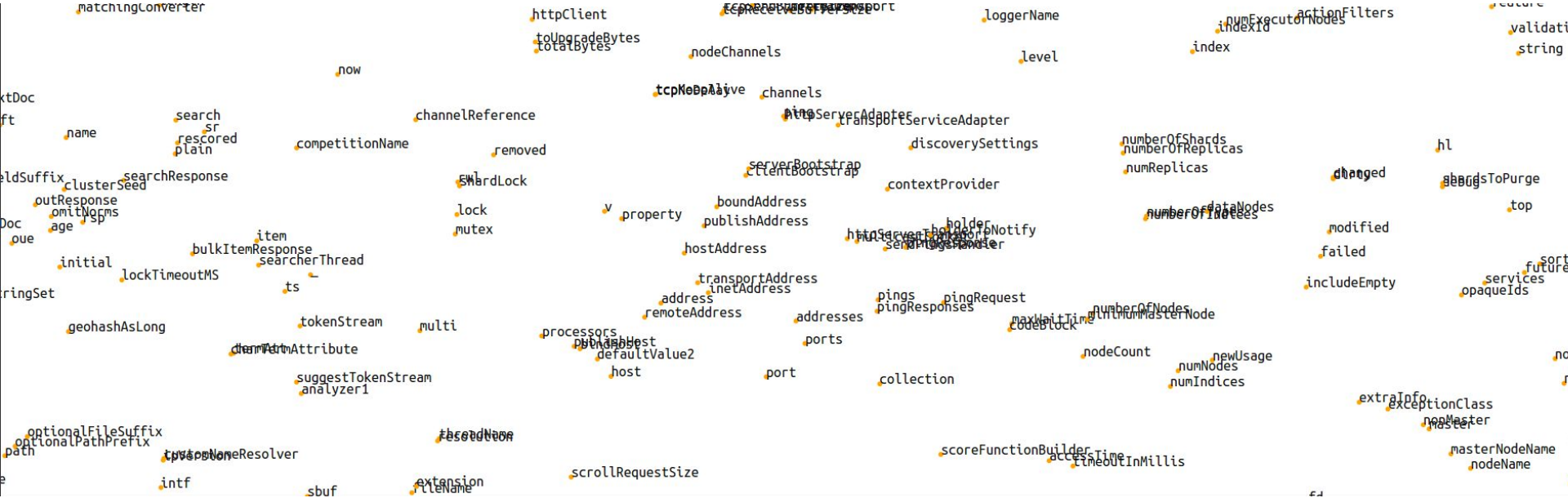


$$P_{\mathcal{D}}(\pi|f(\mathbb{c}))$$

$$\downarrow$$

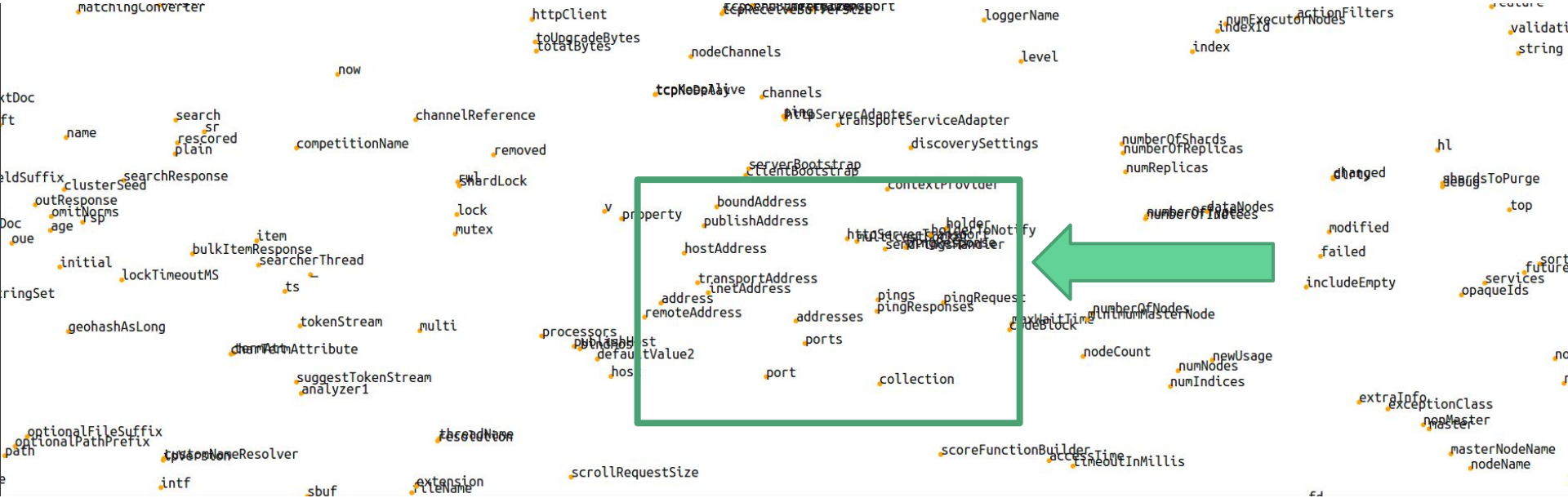
$$\mathbb{R}^D$$

Distributed Representations



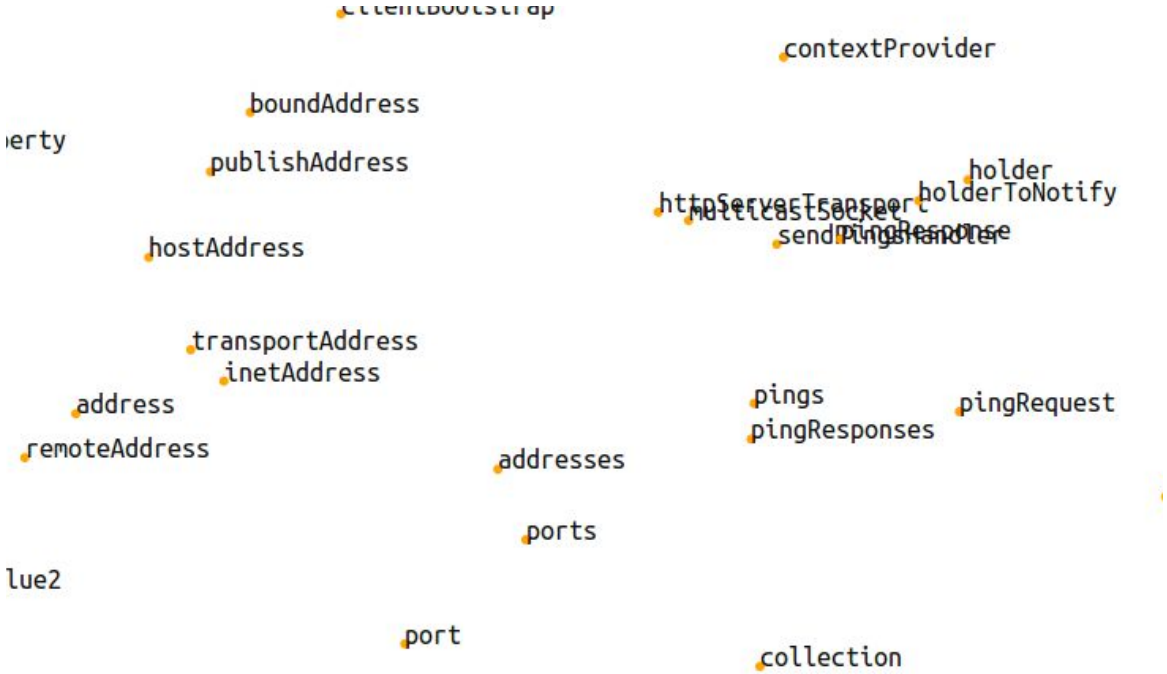
Allamanis *et al.*, “Suggesting accurate method and class names”, 2015

Distributed Representations



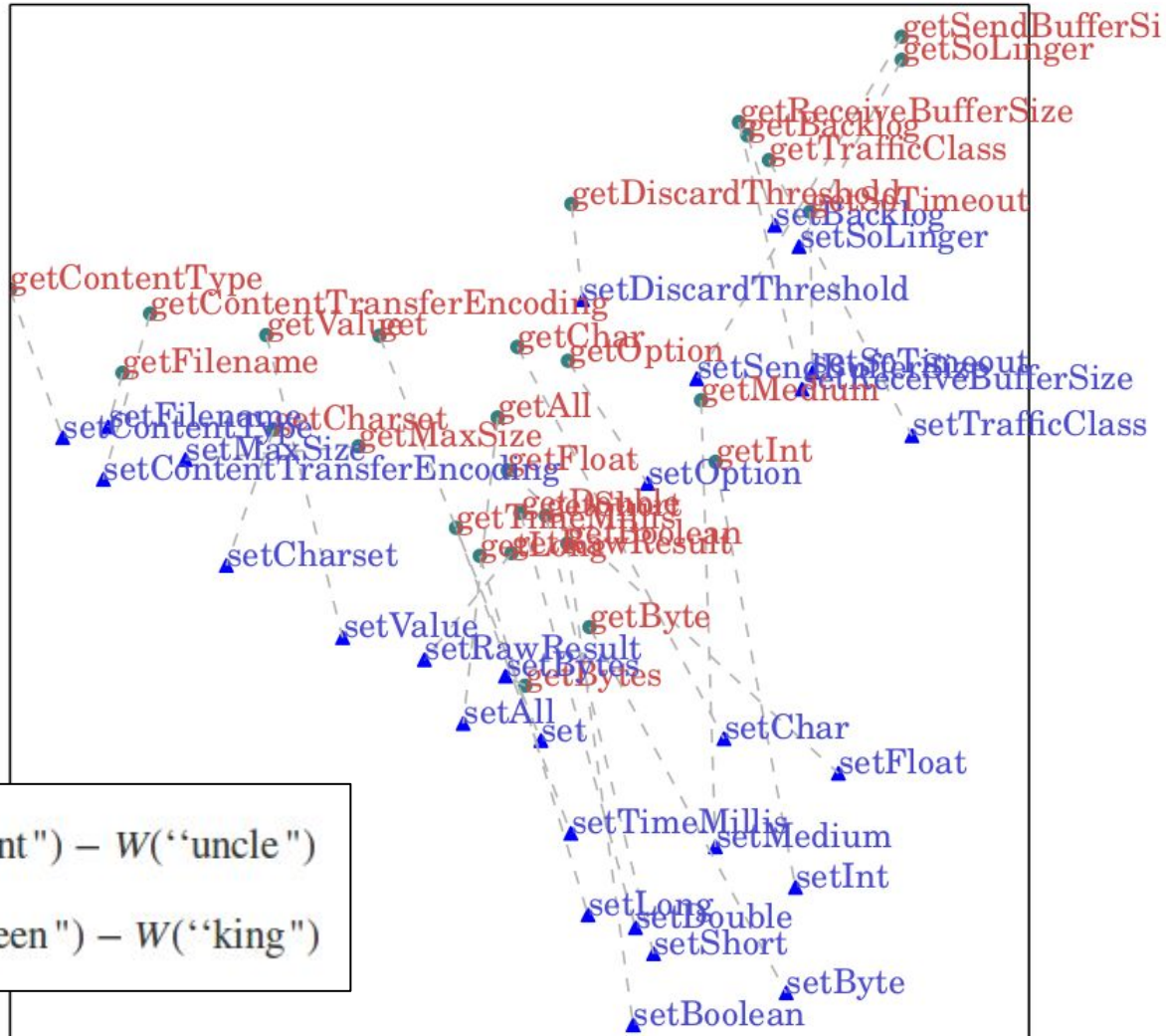
Allamanis *et al.*, “Suggesting accurate method and class names”, 2015

Distributed Representations



Allamanis *et al.*, “Suggesting accurate method and class names”, 2015

Setters *vs*
Getters
in
bigbluebutton



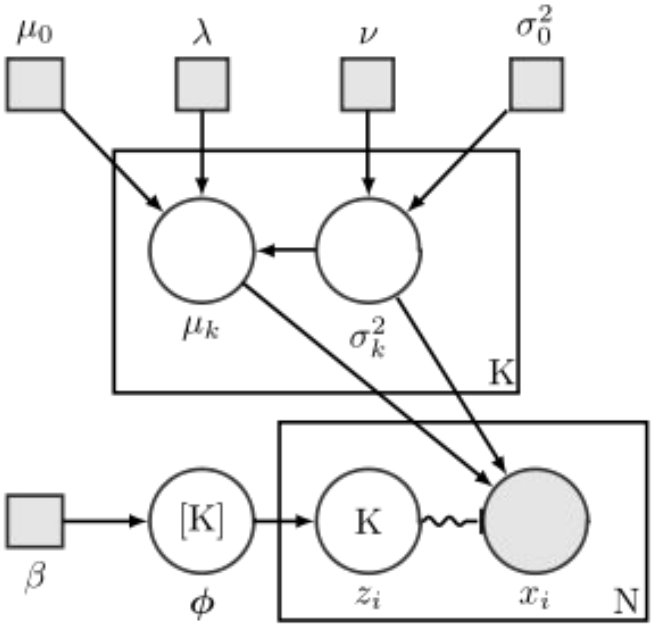
$$W(\text{“woman”}) - W(\text{“man”}) \simeq W(\text{“aunt”}) - W(\text{“uncle”})$$

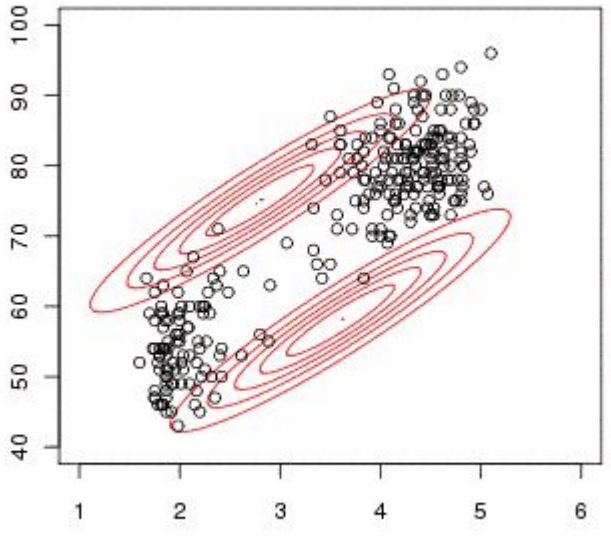
$$W(\text{“woman”}) - W(\text{“man”}) \simeq W(\text{“queen”}) - W(\text{“king”})$$

Pattern Mining Models of Code

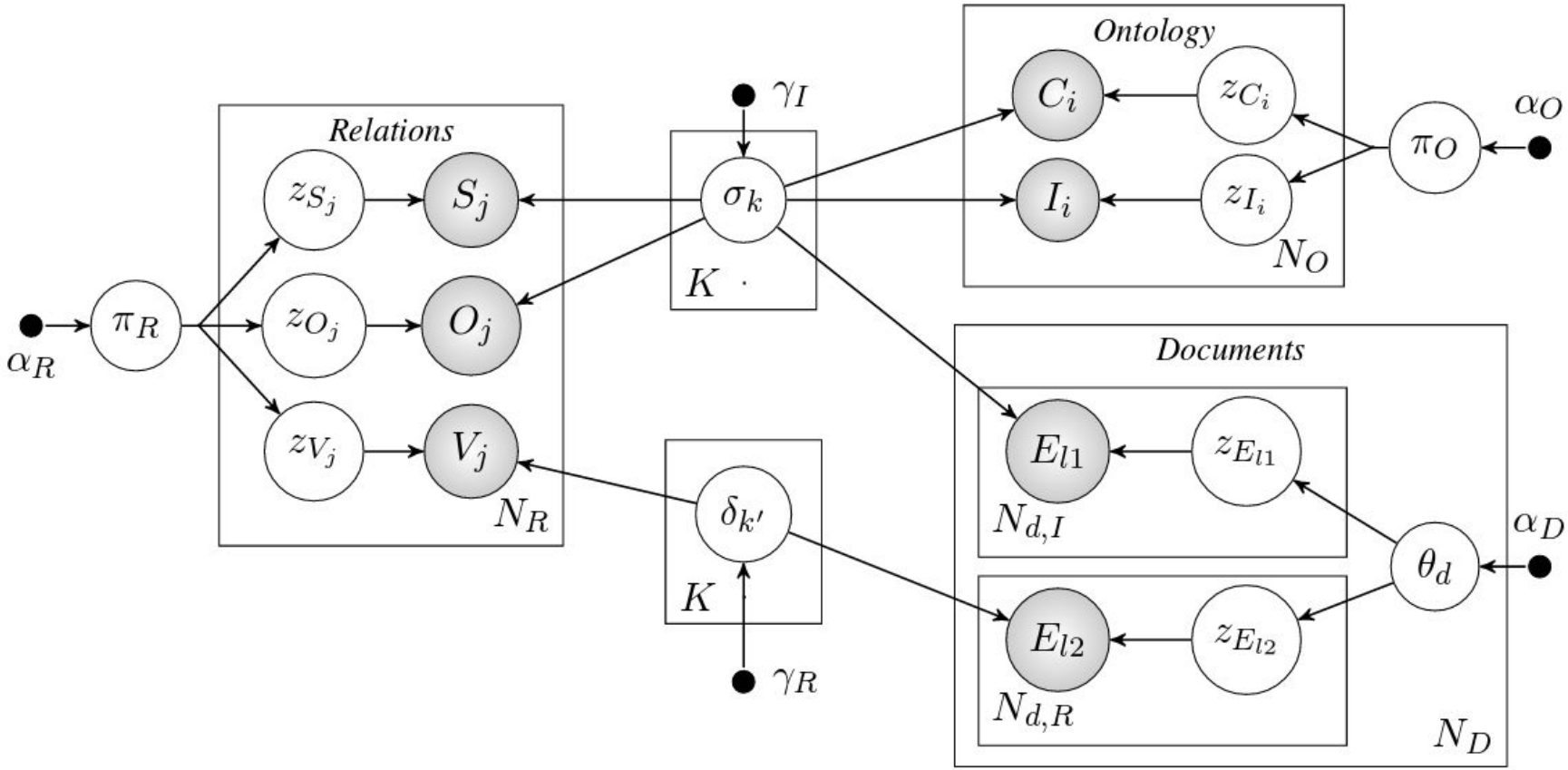
$$P_{\mathcal{D}}(f(\mathbb{c})) = \sum_{\mathbb{l}} P_{\mathcal{D}}(f(\mathbb{c})|\mathbb{l})P(\mathbb{l})$$

Latent Variable Models

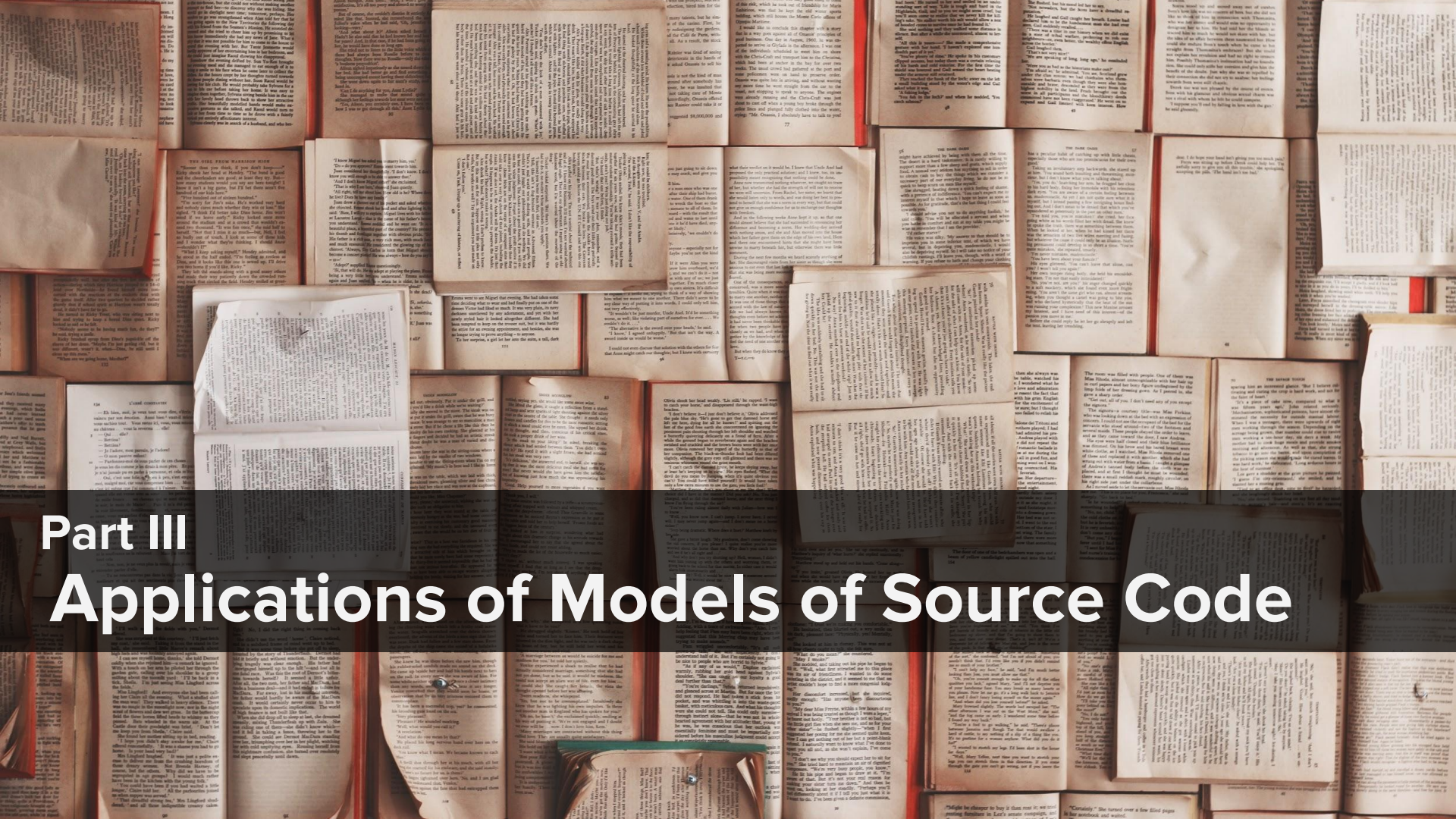




Images from Wikipedia

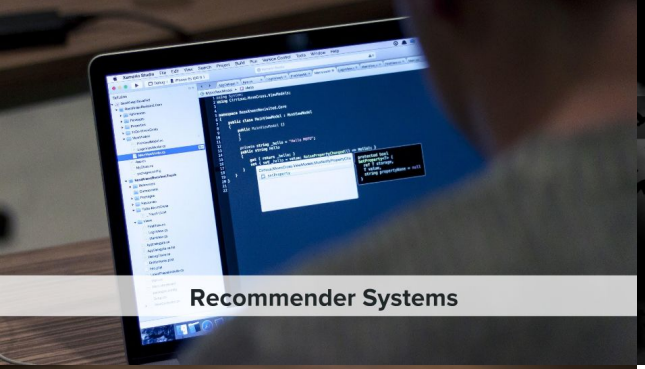


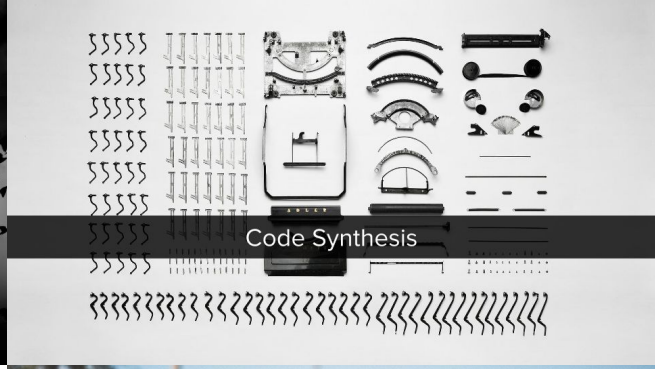
Movshovitz-Attias & Cohen, “KB-LDA: Jointly learning a knowledge base of hierarchy, relations, and facts”, 2015

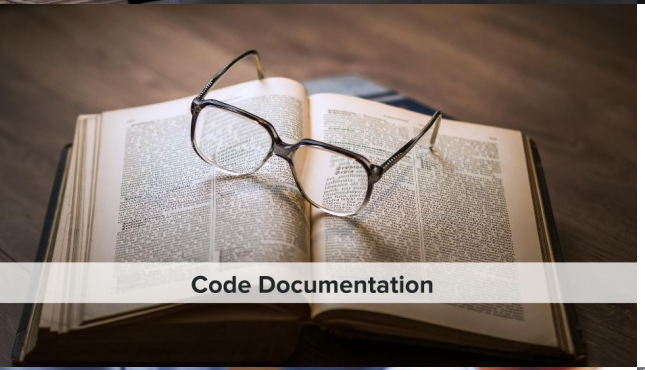


Part III
Applications of Models of Source Code

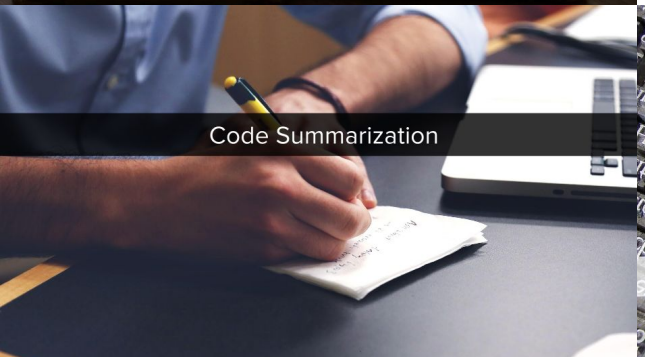
Machine learning helps each application area by **learning** to resolve (some) **ambiguities** from **data**.

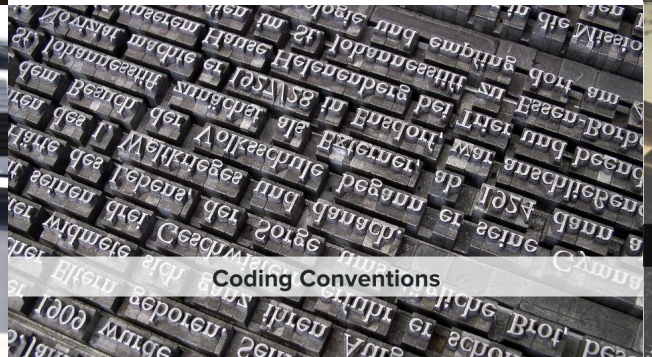


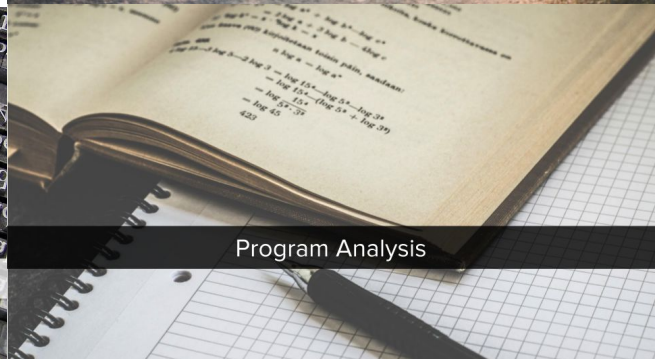




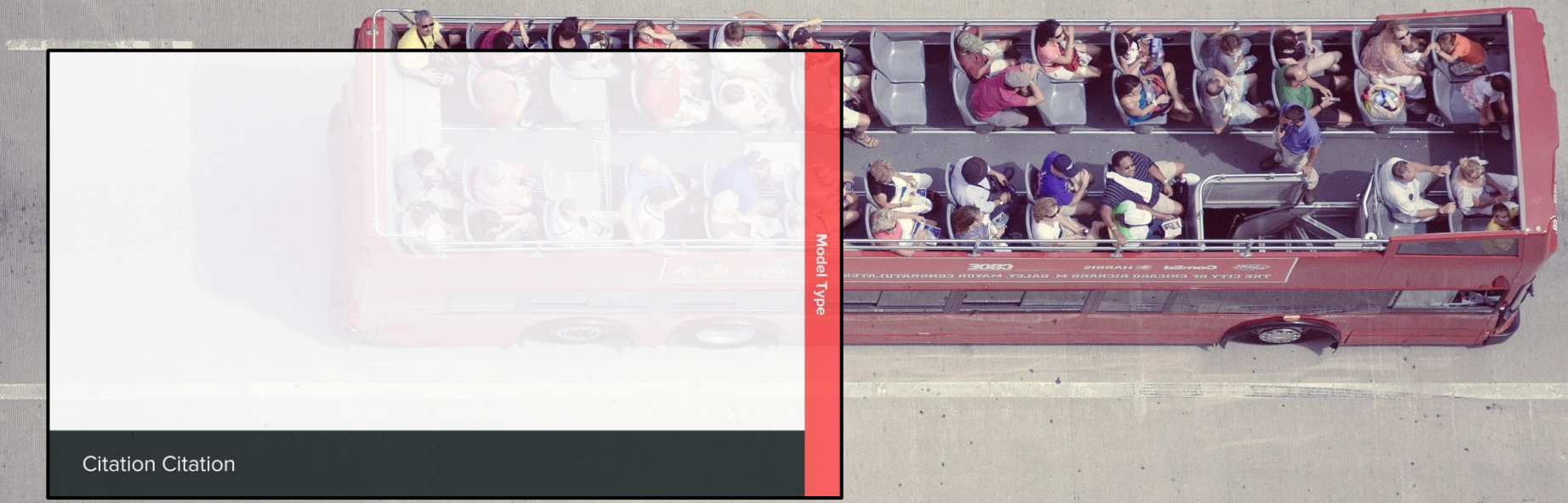
Machine Learning Models





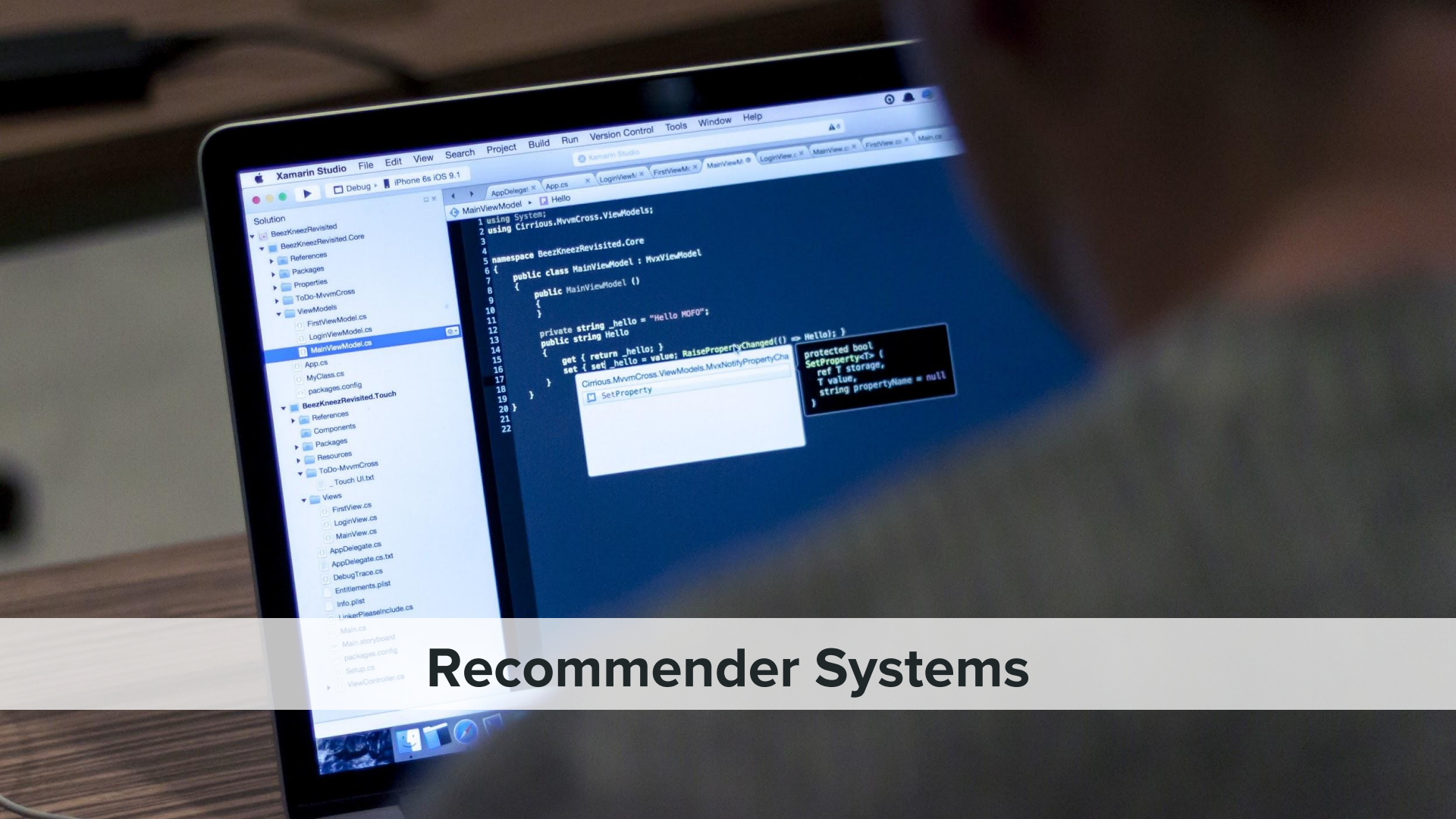


Applications: A (Very) Brief Tour

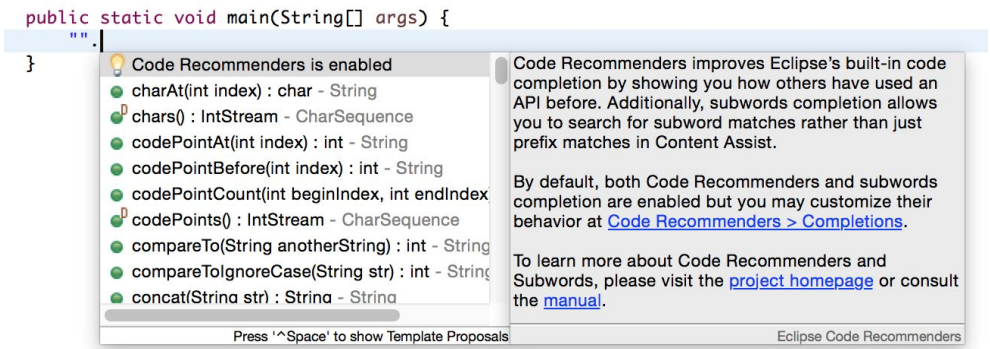


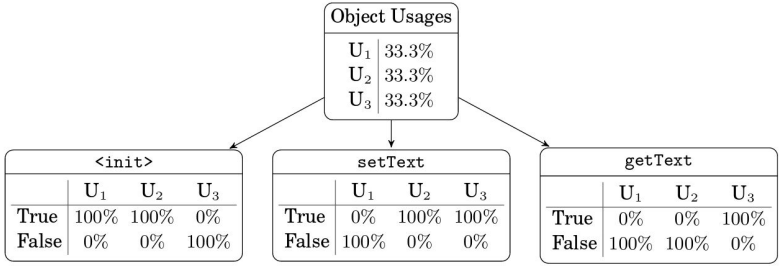
Model Type

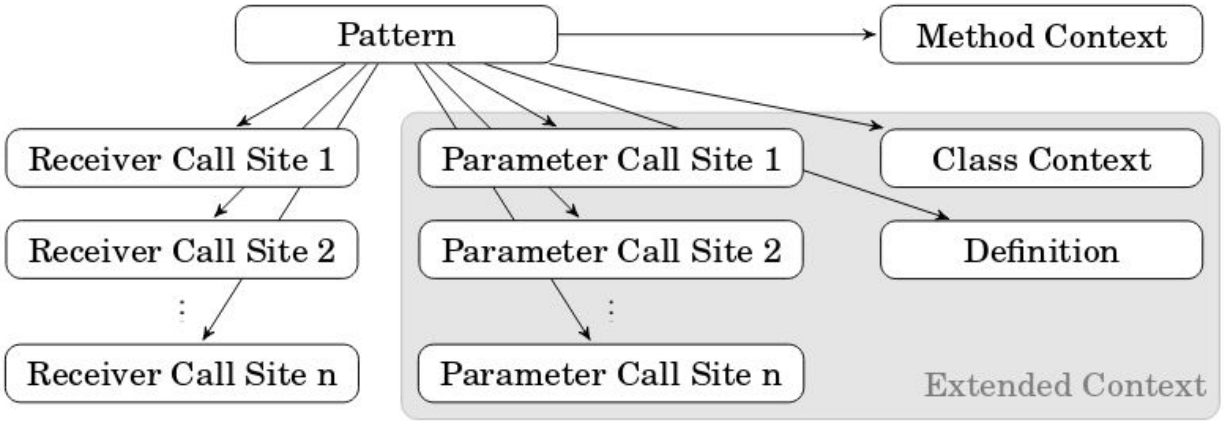
Citation Citation



Recommender Systems







Proksch *et al.* 2015 Bruch *et al.* 2009

Structured Prediction Models

Token Role	Construction Rule
Data type T	TYPE[T]
Variable x	VAR[typeof(x)]
Literal v	LIT[typeof(v)]
Function decl m	FUNC[type(m),lexeme(m),paralist(m),rettype(m)]
Function call m	CALL[type(m),lexeme(m),paracount(m),rettype(m)]
Parameter x	PARA[typeof(x)]
Field f	FIELD[type(f), lexeme(f)]
Operator o	OP[name(o)]
Cast (T)	CAST[T]
Keyword	*To corresponding reserved token*
Block open & close	*To corresponding reserved token*
Special literal	*To corresponding reserved token*
Unknown	*To special lexical token* LEX

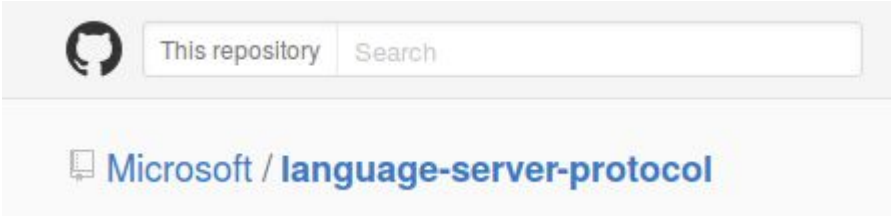
```
offset -= replacementLength;
char c = ctx.getDocument().getChar(offset);
boolean prevCharWasWhitespace = false;
while (numberOfSeparators < 10 && offset >= 0){
    if (!(Character.isWhitespace(ctx.getDocument().get
        prev
        b.ap
    }
    else {
        if(p
        prev
    }
    if(c ==
        numb
    }
    offset--
    if (offs
        brea
    }
    c = ctx.
}
```

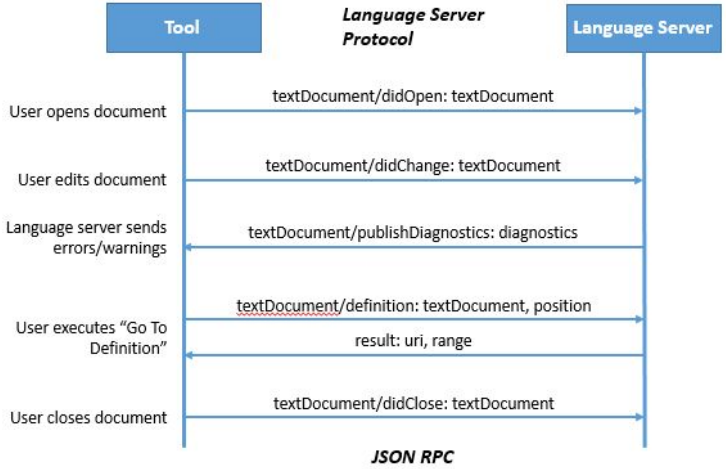
getDocument() : IDocument - ContentAssistInvocationCc
lookup
computeIdentifierPrefix() : CharSequence - ContentAssist
equals(Object obj) : boolean - ContentAssistInvocationCc
getClass() : Class<?> - Object
getCompilationUnit() : ICompilationUnit - JavaContentAs
getCoreContext() : CompletionContext - JavaContentAssi
getExpectedType() : IType - JavaContentAssistInvocation(
getHistoryRelevance(String qualifiedTypeName) : float - J
getInvocationOffset() : int - ContentAssistInvocationCont
getKeywordProposals() : IJavaCompletionProposal[] - Jav
Press 'Ctrl+Space' to show CACHECA Proposa

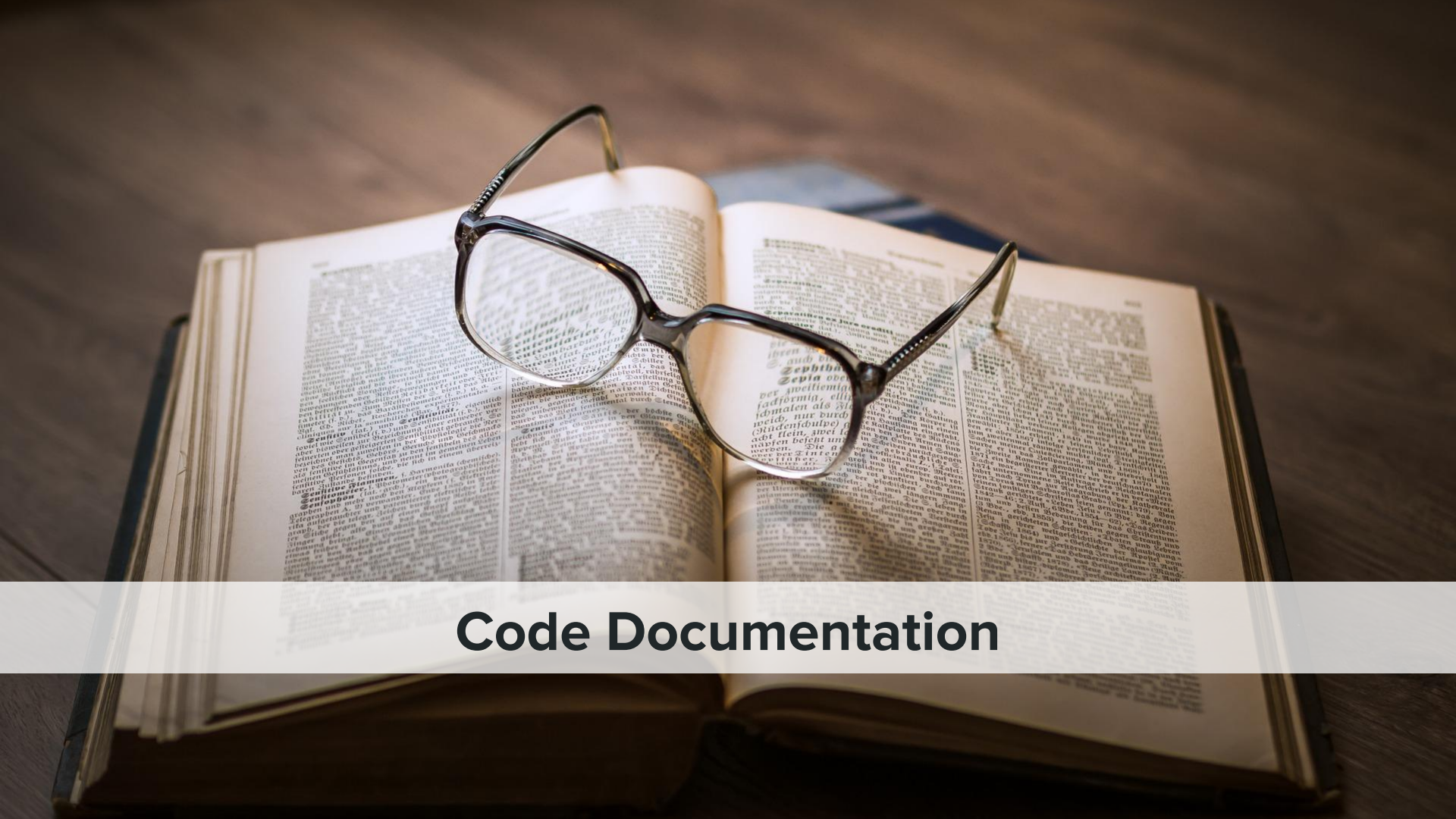
Language Model

Hindle *et al.* 2012 Nguyen *et al.* 2013 Franks *et al.* 2015

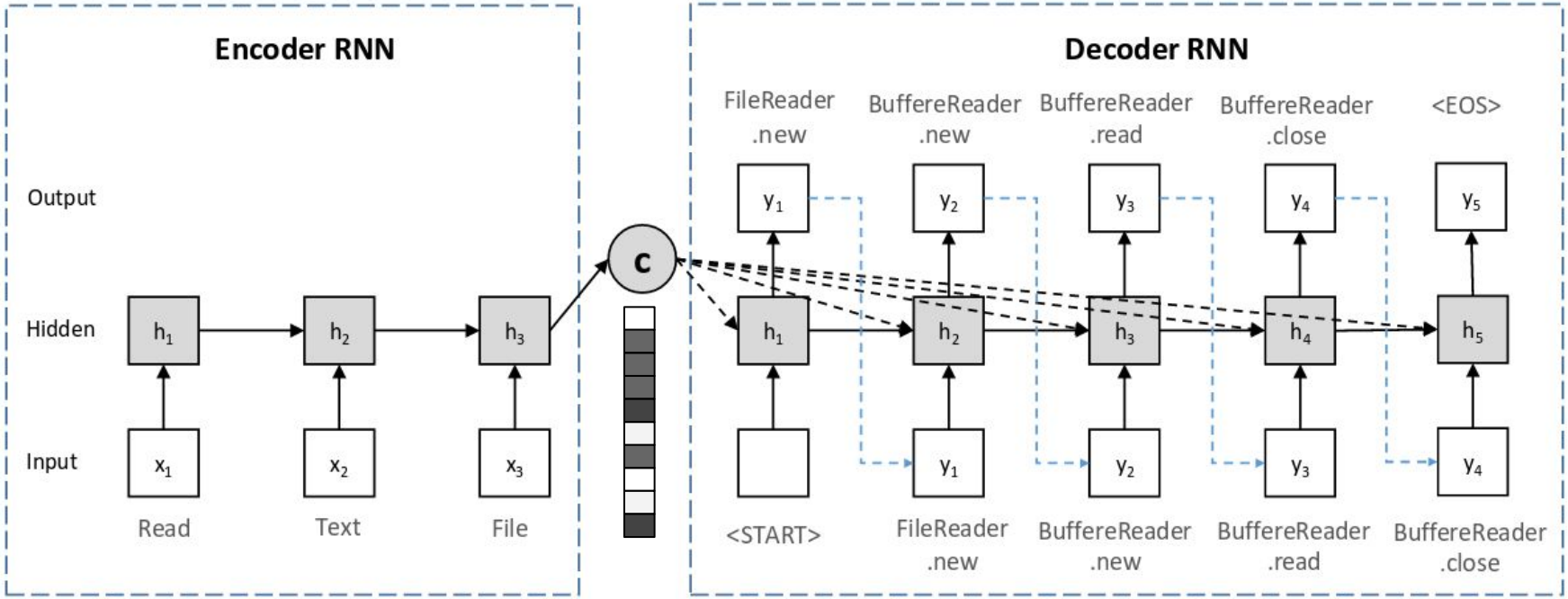
Language Server Protocol



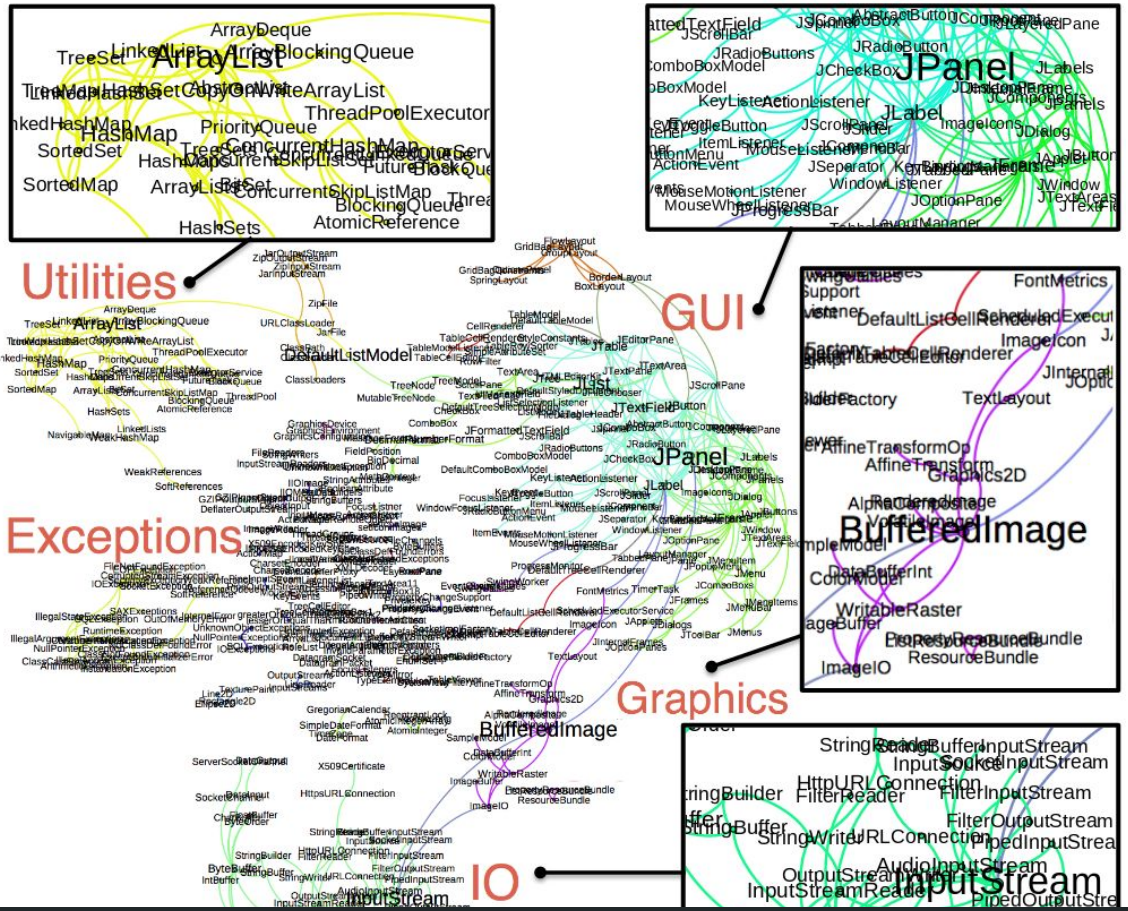




Code Documentation



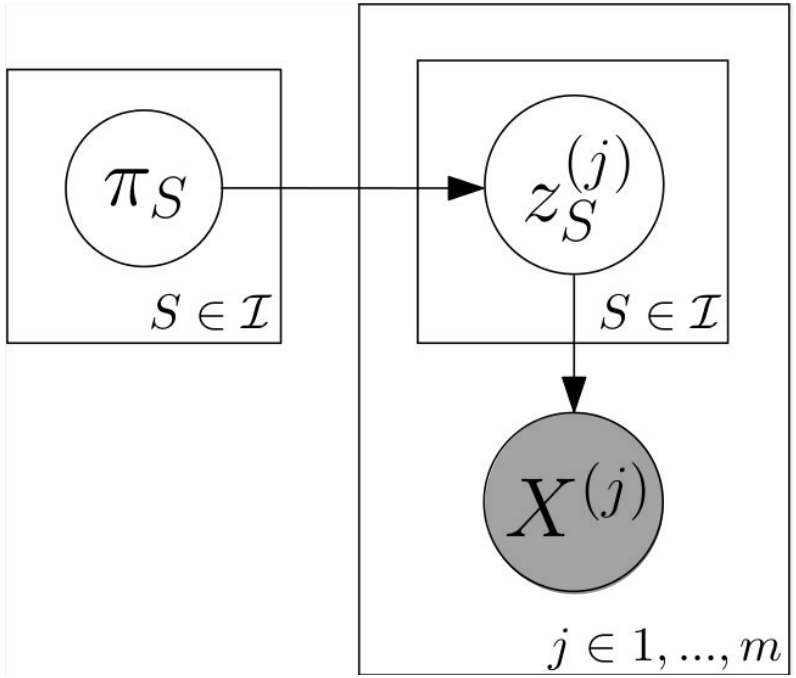
Gu *et al.* “Deep API Learning” 2016



Movshovitz-Attias & Cohen, “Grounded Discovery of Coordinate Term Relationships between Software Entities”, 2015

Latent Variable Model

```
TwitterFactory.<init>
TwitterFactory.getInstance

Status.getUser
Status.getText

ConfigurationBuilder.<init>
ConfigurationBuilder.build

ConfigurationBuilder.<init>
TwitterFactory.<init>

ConfigurationBuilder.<init>
ConfigurationBuilder.setOAuthConsumerKey

ConfigurationBuilder.build
TwitterFactory.<init>

ConfigurationBuilder.<init>
ConfigurationBuilder.build
TwitterFactory.<init>
```



Latent Variable Model

Fowkes & Sutton, “Parameter-Free Probabilistic API Mining across GitHub”, 2016


Statistical Code Migration

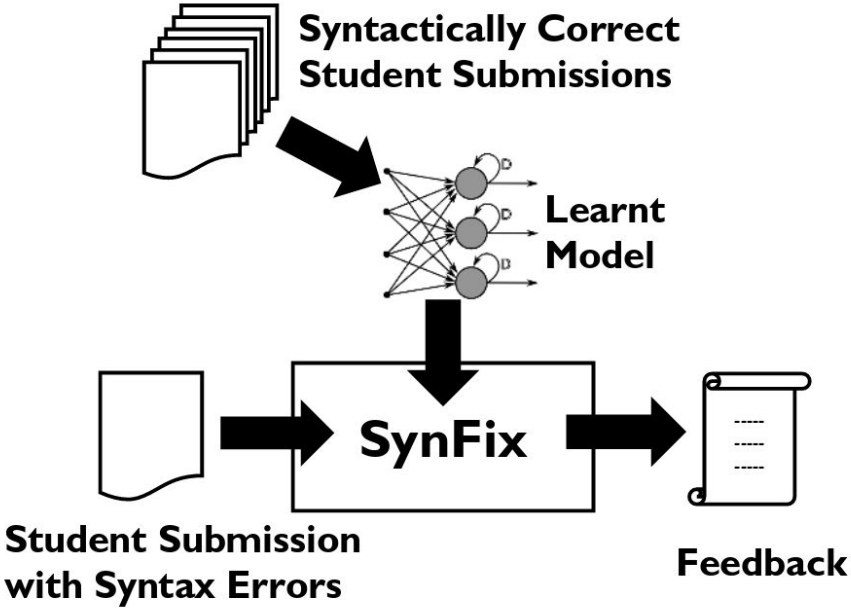
Subphase	Output	Example output
Parallel data collection	Method pairs	**C#:** `Console . WriteLine ( "Hello World!" ) ;` **Java:** `System . out . println ( "Hello World!" ) ;`
Word alignment	Aligned method pairs	**C#:** `Console . WriteLine ( "Hello World!" ) ;` **Java:** `System . out . println ( "Hello World!" ) ;`
Phrase table construction	Phrase table	**C# phrase** | **Java phrase** | **Score** i.e. $Pr($**C# phrase** $\mid$ **Java phrase**$)$ `Console` | `System . out` | 0.8 `WriteLine` | `println` | 0.5 `.` | `.` | 0.7 `(` | `(` | 0.9 ... | ... | ...

$$\mathbb{t}^* = \arg\max P_{\mathcal{D}}(\mathbb{t}|\mathbb{s}) = \arg\max P_{\mathcal{D}}(\mathbb{s}|\mathbb{t}) P_{\mathcal{D}}(\mathbb{t})$$

Raychev *et al.* Onwards 2014

Nguyen *et al.* FSE 2013, ASE 2015

Code Defects

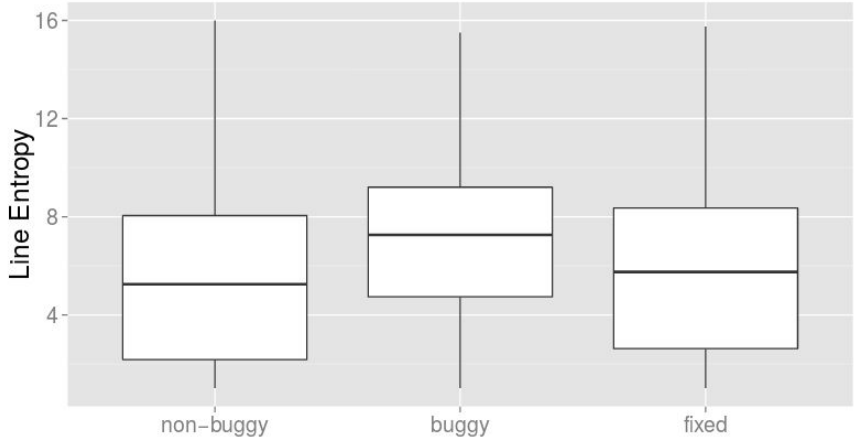



```
1 for (int i = 0; i < scorers.length; i++) {
2     if (scorers[i].nextDoc() == NO_MORE_DOCS)
5         lastDoc = NO_MORE_DOCS;
6         return;
7     }
8 }
```

Language Model

Campbell *et al.* MSR 2014

Bhatia & Singh, 2016



Netty (2013-08-20)
File: `ThreadPerChannelEventLoopGroup.java`

Entropy dropped after bugfix : **4.6257**

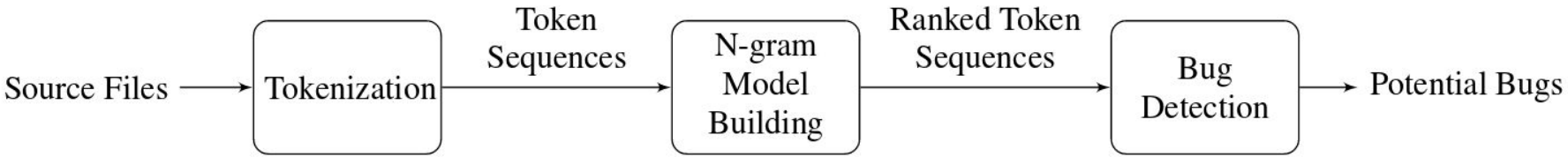
```
  if (isTerminated()) {
    // Before (entropy = 5.96485):
-   terminationFuture.setSuccess(null);
    // After (entropy = 1.33915):
+   terminationFuture.trySuccess(null);
```

Ray *et al.* “On the Naturalness of Buggy Code”, 2016

(Generative) Language Model

```
6
7  # CodexLint Example:
8  "a string".split("\n").to_s
9
10
```

Function to_s has appeared 12 times and split has appeared 29 times, and they've appeared 0 times together.

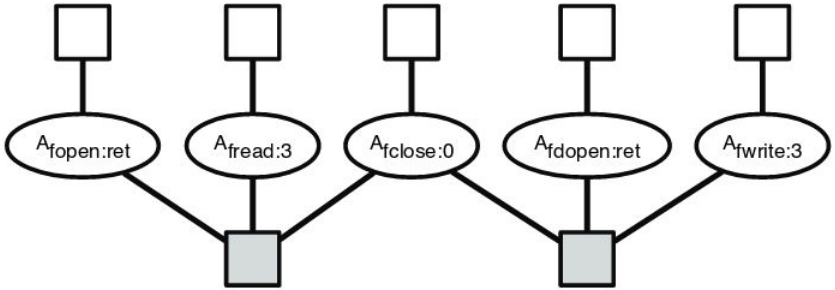


Fast *et al.* CHI 2014

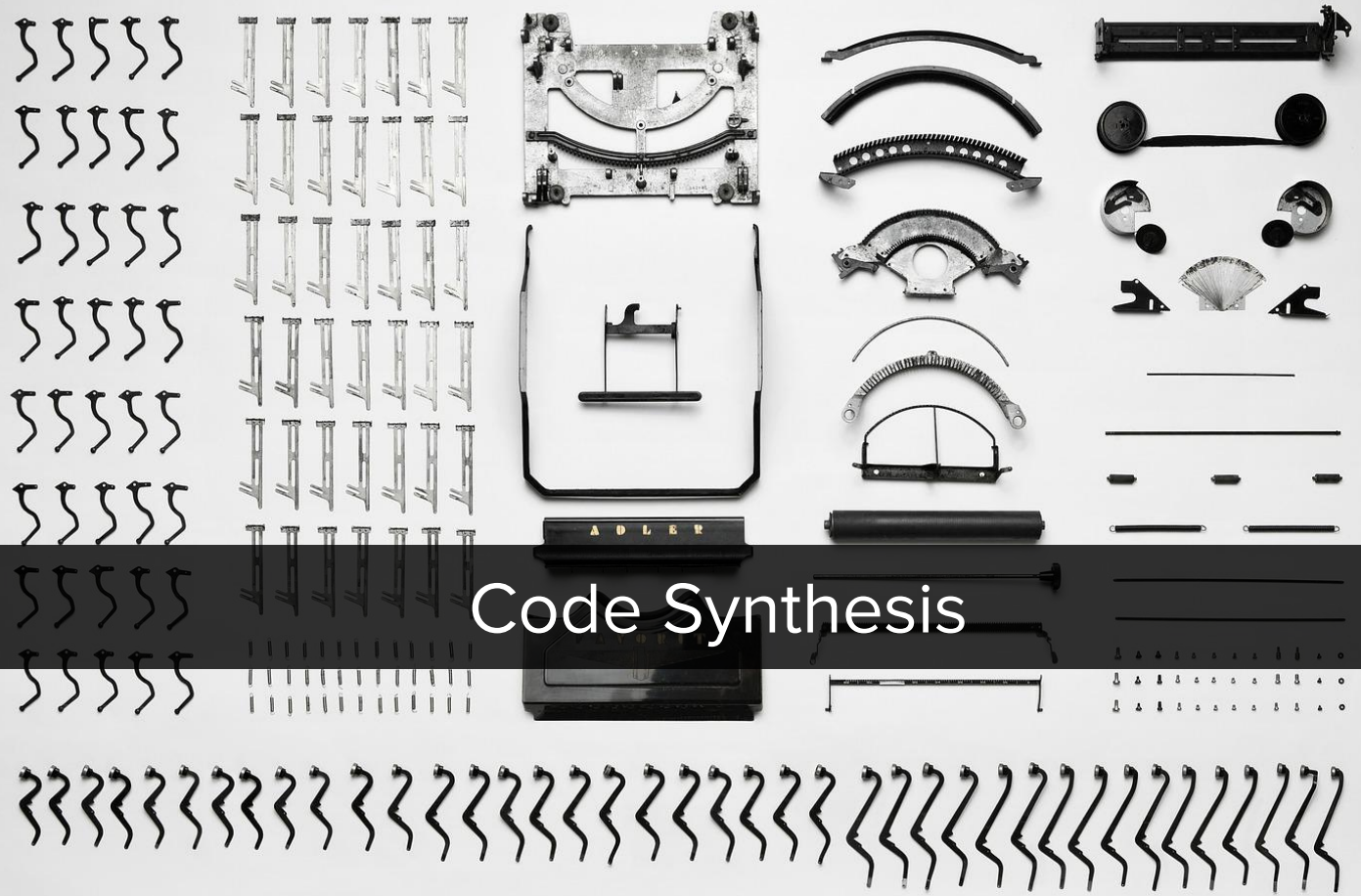
Wang *et al.* ASE 2016

Language Model

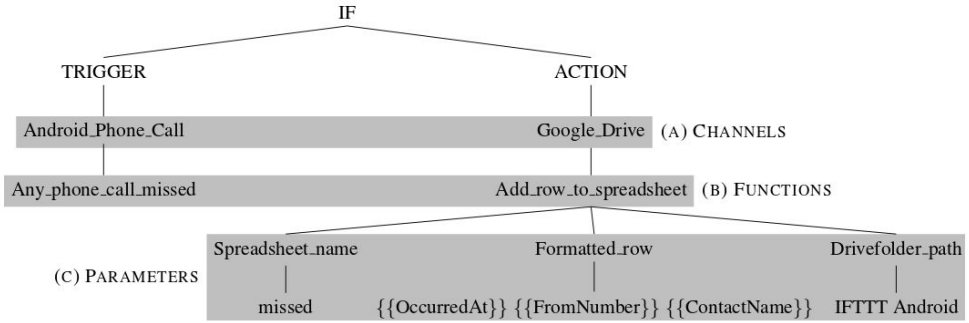
```
1.  FILE * fp1 = fopen( "myfile.txt", "r" );
2.  FILE * fp2 = fdopen( fd, "w" );
3.  fread( buffer, n, 1, fp1 );
4.  fwrite( buffer, n, 1, fp2 );
5.  fclose( fp1 );
6.  fclose( fp2 );
```



Kremenek *et al.* “A Factor Graph Model for Software Bug Finding”, 2007



Code Synthesis



Archive your missed calls from Android to Google Drive

sum the totalpay for the capitol hill baristas

sum the totalpay for the capitol hill baristas ≈ `SUMIFS(WeeklyHours[totalpay], WeeklyHours[location], "capitol hi`
sum the totalpay for the ~~capitol hill~~ baristas ≈ `SUMIF(WeeklyHours[title], "barista", WeeklyHours[totalpay])`
sum the totalpay for the capitol hill ~~baristas~~ ≈ `SUMIF(WeeklyHours[location], "capitol hill", WeeklyHours[totalpay`

I2 | =SUMIFS(WeeklyHours[totalpay], WeeklyHours[location], "capitol hill", WeeklyHours[title], "barista")

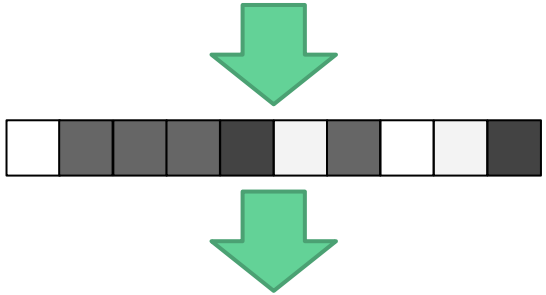
	A	B	C	D	E	F	G	H	I
1	location	name	title	hours	othours	basepay	otpay	totalpay	
2	capitol hill	aaron	chef	18	0	$243.00	$0.00	$243.00	$888.80
3	capitol hill	blanca	cashier	25	0	$193.75	$0.00	$193.75	
4	capitol hill	chris	manager	40	10	$990.00	$371.25	$1,361.25	
5	capitol hill	deeraj	barista	40	0	$352.00	$0.00	$352.00	
6	capitol hill	grace	barista	21	0	$184.80	$0.00	$184.80	
7	capitol hill	hannah	cashier	40	3	$310.00	$34.88	$344.88	
8	capitol hill	irene	barista	40	0	$352.00	$0.00	$352.00	
9	queen anne	tom	chef	16	0	$216.00	$0.00	$216.00	
10	queen anne	susan	barista	26	0	$228.80	$0.00	$228.80	
11	queen anne	steve	cashier	22	0	$170.50	$0.00	$170.50	

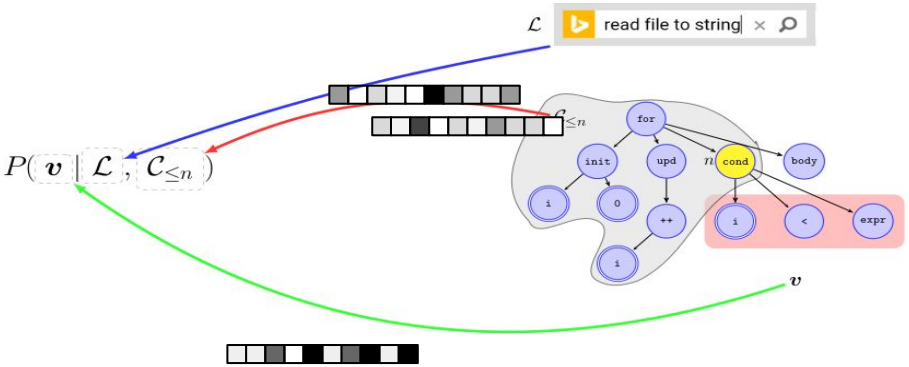
Quirk *et al.* ACL 2015, 2016

Gulwani & Marron, ICMD 2014

Bimodal Generative Models

- all elements to small letters for each line and sort
- all elements to small letters for each line and order
- all elements to lowercase for each line and sort
- all elements to lowercase for each line and order
- all elements lowercase for each line and sort
- all elements lowercase for each line and order
- all elements to lower case for each line and sort
- all elements to lower case for each line and order
- all elements to small letters for each new line and sort

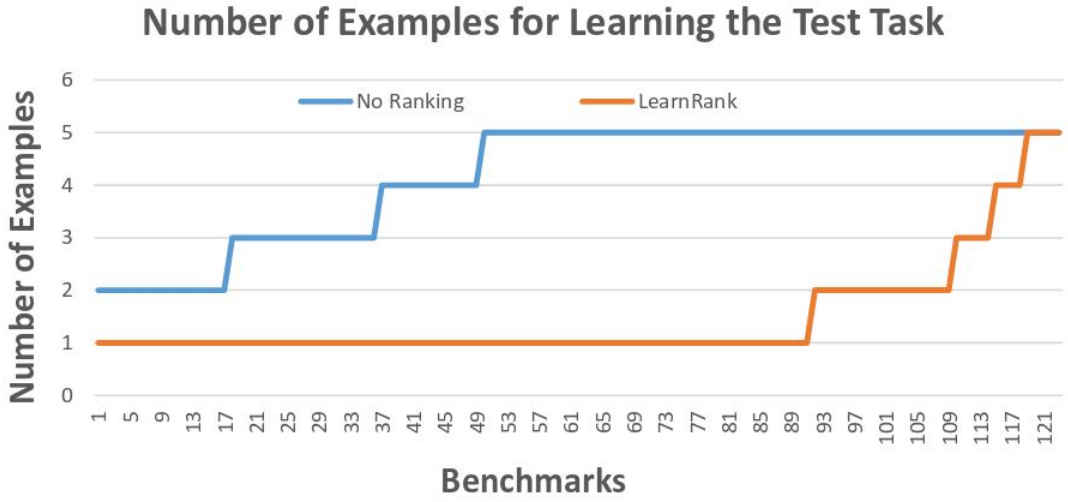





```
string result = String.Join("\\n",input_string.Split(\'\\n\')
        .Select((string x) => x.ToLower()).OrderBy(x => x));
```

Allamanis *et al.* “Bimodal modelling of source code and natural language” ICML 2015

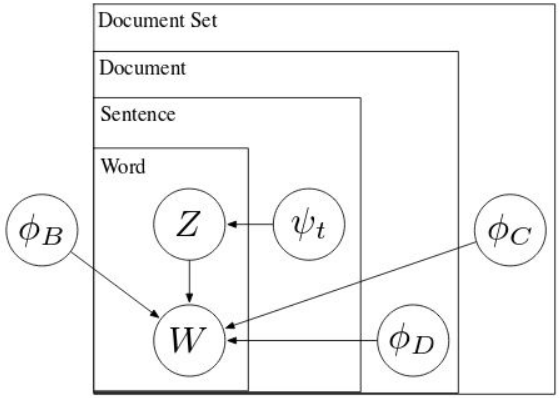
Bimodal Generative + Distributed Representation Model



Singh & Gulwani “Predicting a Correct Program in Programming By Example” 2015


Code Summarization

```
{
    protected int code;
    protected String reason;

    /** Construct StatusLine  */
    public StatusLine(int c, String r)
    {  code=c;
       reason=r;
    }

    /** Create a new copy of the request-line*/
    public Object clone()
    {...}

    /** Indicates whether some other Object is "equal to" this StatusLine
    public boolean equals(Object obj)
    {...}

    public String toString()
    {...}

    public int getCode()
    {...}

    public String getReason()
    {...}
}
```

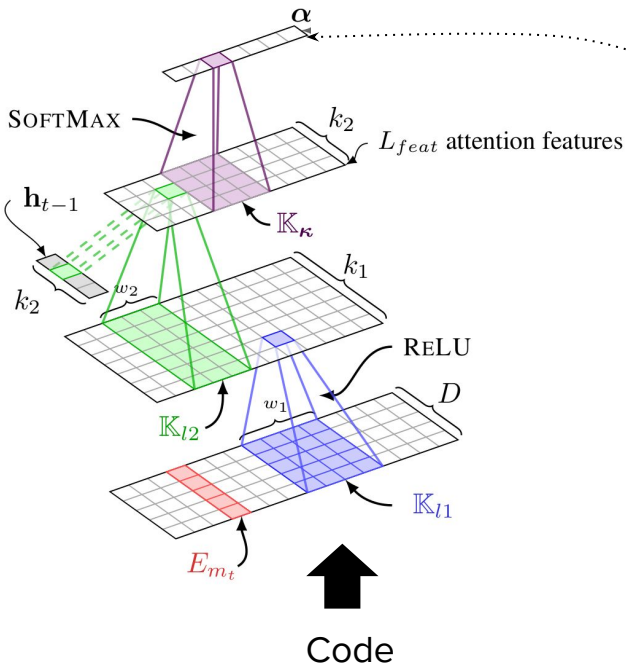


Haghighi & Vanderwende, 2009

Latent Variable Model

Fowkes *et al.* “Autofolding for Source Code Summarization” 2016

Summarization via Method Name Prediction



conv_attention (code **c**, previous state $\mathbf{h}_{t-1}$)

$\boldsymbol{\alpha} \leftarrow$ **attention_weights** $(L_{feat}, \mathbb{K}_{att})$

Allamanis *et al*, “A Convolutional Attention Network for Extreme Summarization of Source Code”, ICML 2016

Distributed Representation Model

	Target		Attention Vectors	λ
m_1	set	$\boldsymbol{\alpha}$ =	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.012
		$\boldsymbol{\kappa}$ =	<s> { this . use Browser Cache = use Browser Cache ; } </s>	
m_2	use	$\boldsymbol{\alpha}$ =	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.974
		$\boldsymbol{\kappa}$ =	<s> { this . use Browser Cache = use Browser Cache ; } </s>	
m_3	browser	$\boldsymbol{\alpha}$ =	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.969
		$\boldsymbol{\kappa}$ =	<s> { this . use Browser Cache = use Browser Cache ; } </s>	
m_4	cache	$\boldsymbol{\alpha}$ =	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.583
		$\boldsymbol{\kappa}$ =	<s> { this . use Browser Cache = use Browser Cache ; } </s>	
m_5	END	$\boldsymbol{\alpha}$ =	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.066
		$\boldsymbol{\kappa}$ =	<s> { this . use Browser Cache = use Browser Cache ; } </s>	

Attention Visualization

Allamanis *et al*, “A Convolutional Attention Network for Extreme Summarization of Source Code”, ICML 2016

Code Summarization to Natural Language

1. Source Code (C#):

```
public int TextWidth(string text) {
  TextBlock t = new TextBlock();
  t.Text = text;
  return
    (int)Math.Ceiling(t.ActualWidth);
}
```

Descriptions:

a. Get rendered width of string rounded up to the nearest integer

b. Compute the actual textwidth inside a textblock

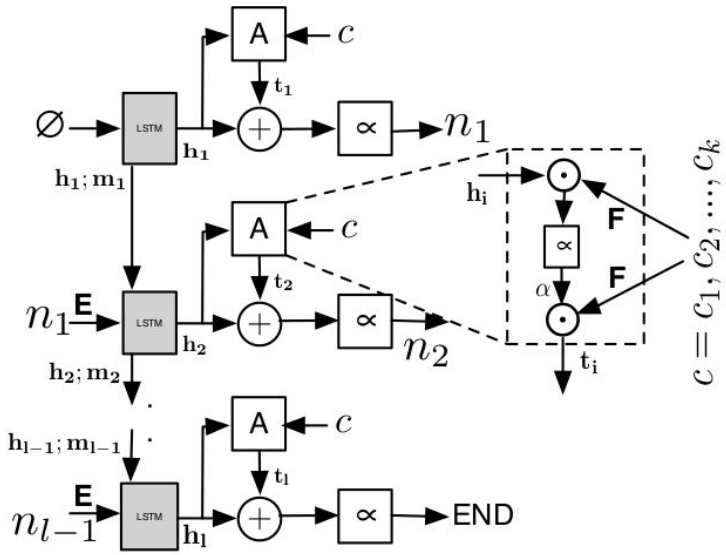
2. Source Code (C#):

```
  var input = "Hello";
  var regEx = new Regex("World");
  return !regEx.IsMatch(input);
```

Descriptions:

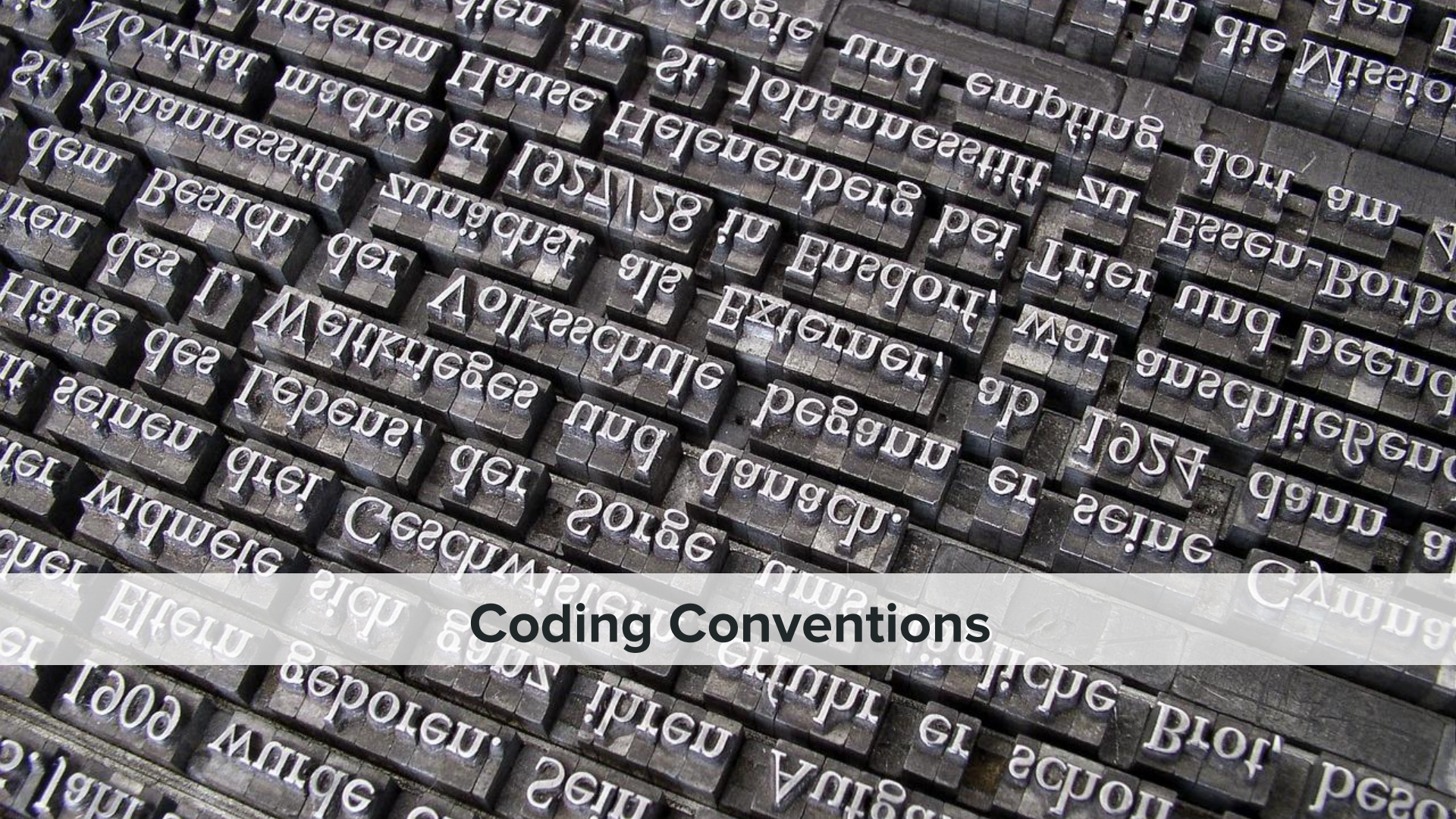
a. Return if the input doesn't contain a particular word in it

b. Lookup a substring in a string using regex



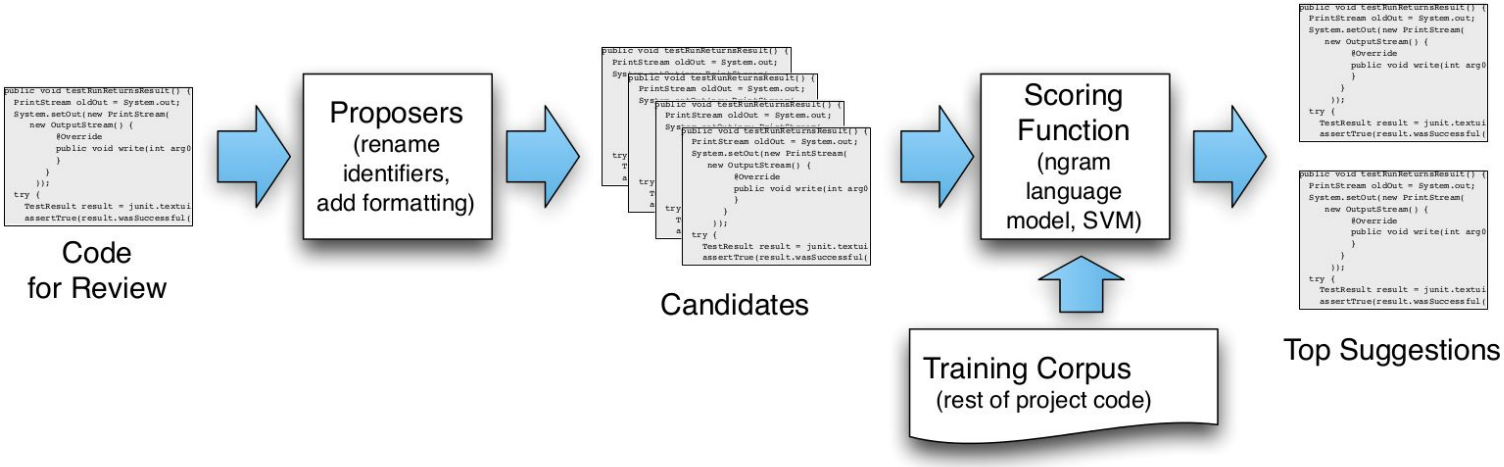
Iyer *et al.* "Summarizing Source Code using a Neural Attention Model" 2016

Distributed Representation Model



Coding Conventions

Naming Conventions



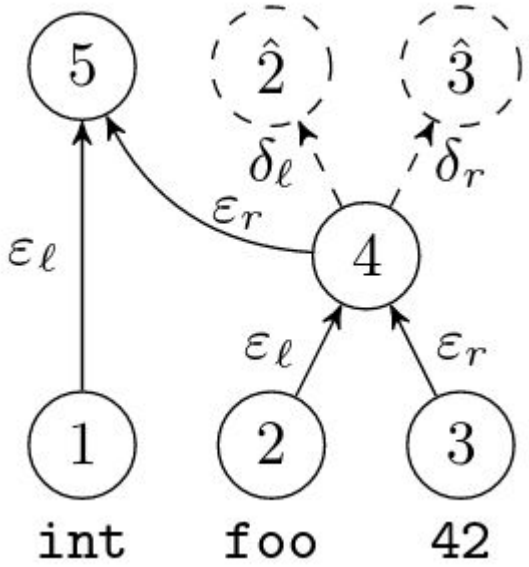

```
ForkJoinTask<?> [redacted];
if (task instanceof ForkJoinTask<?>) // avoid re-wrap
     [redacted] = (ForkJoinTask<?>) task;
else
     [redacted] = new
ForkJoinTask.AdaptedRunnableAction(task);
externalPush([redacted]);
```

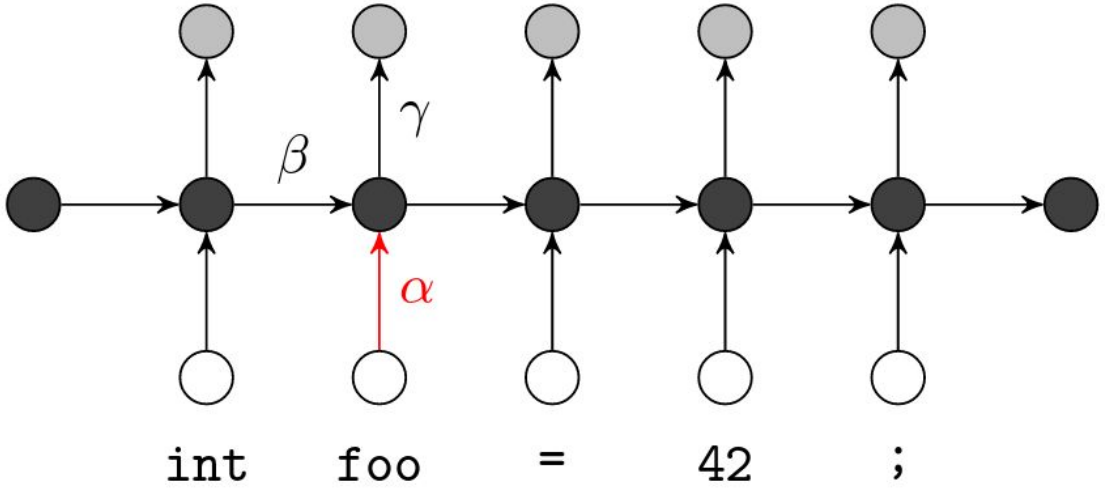
Suggestions

1. job (30%)
2. task (20%)
3. tsk (15%)

Allamanis *et al.* FSE 2014, 2015

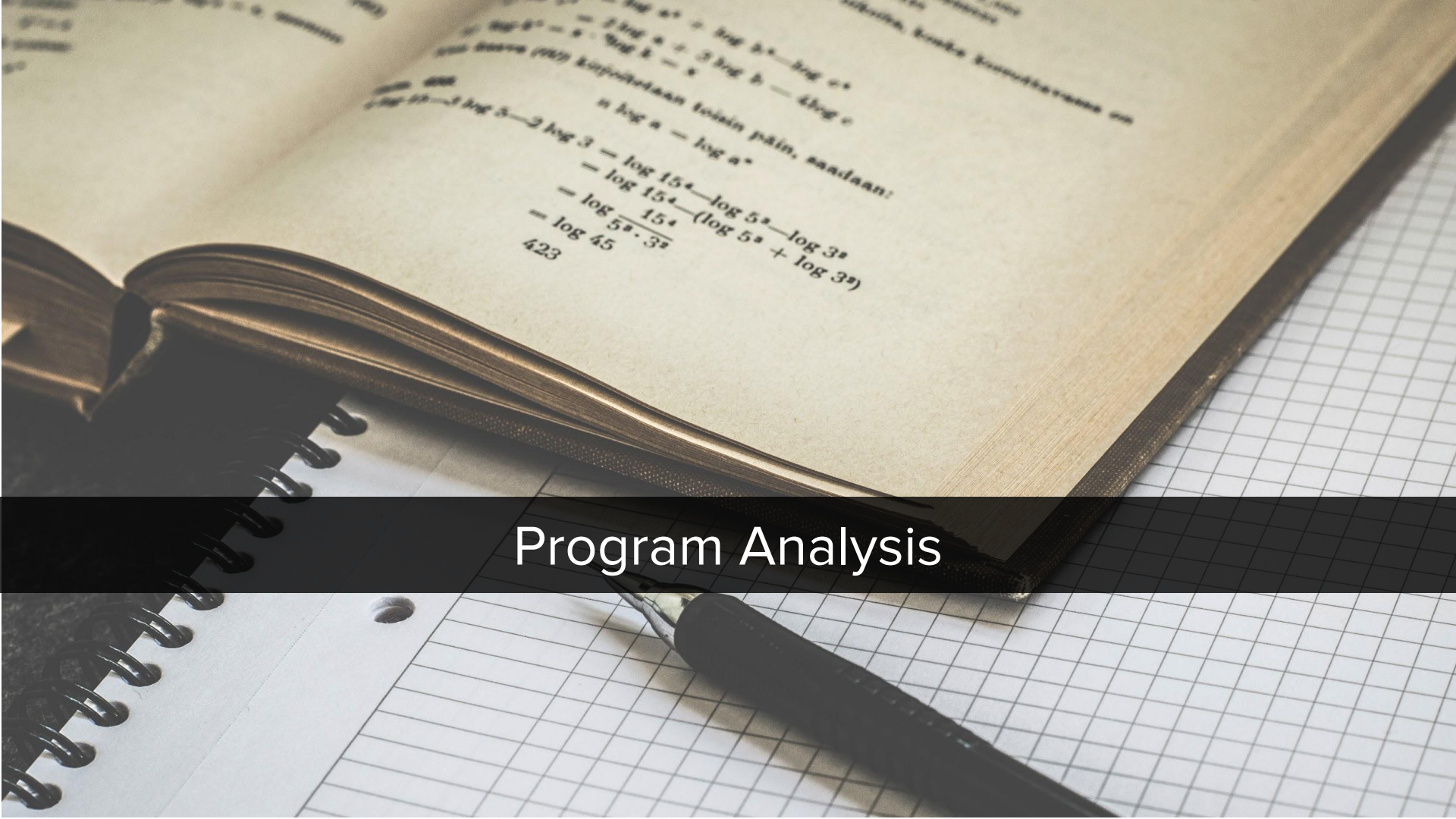
Distributed Representation / Language Model





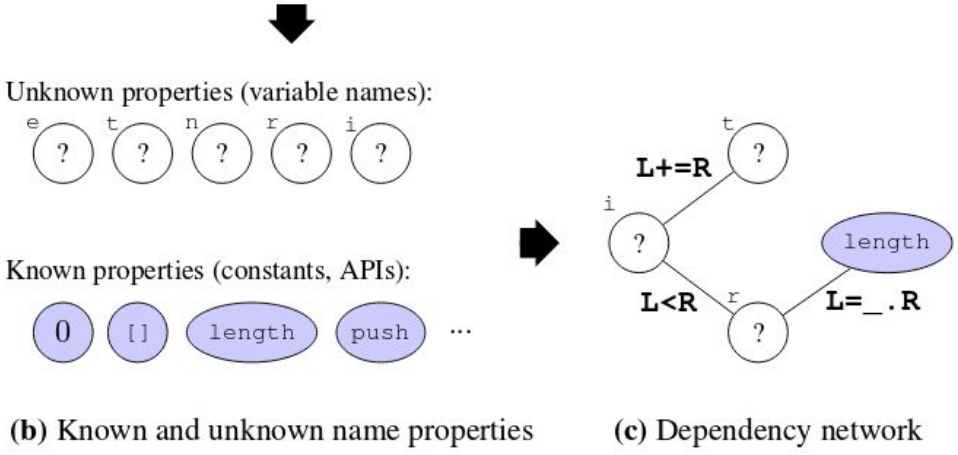
Distributed Representation Model

White *et al.* “Deep Learning Code Fragments for Code Clone Detection” 2016


Program Analysis

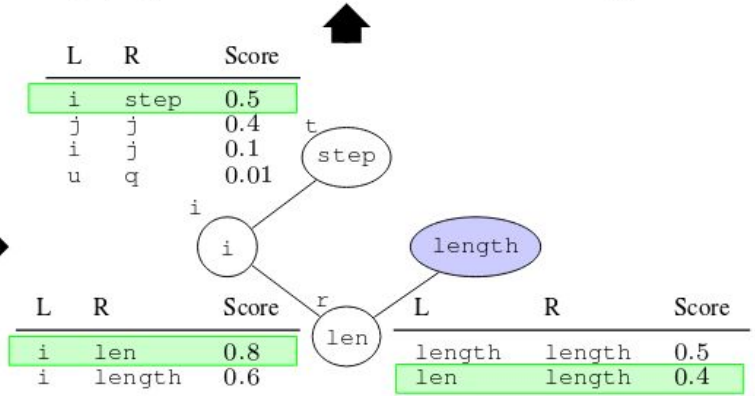
```
function chunkData(e, t) {
  var n = [];
  var r = e.length;
  var i = 0;
  for (; i < r; i += t) {
    if (i + t < r) {
      n.push(e.substring(i, i + t));
    } else {
      n.push(e.substring(i, r));
    }
  }
  return n;
}
```

(a) JavaScript program with minified identifier names



(b) Known and unknown name properties

(c) Dependency network



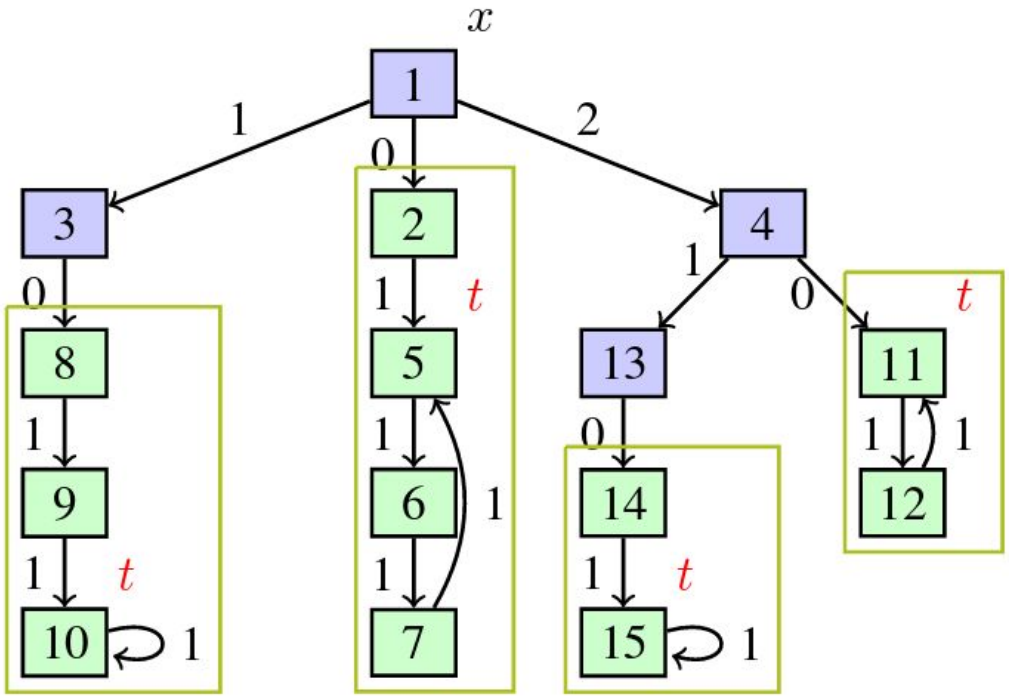
(d) Result of MAP inference

```
/* str: string, step: number, return: Array */
function chunkData(str, step) {
  var colNames = []; /* colNames: Array */
  var len = str.length;
  var i = 0; /* i: number */
  for (; i < len; i += step) {
    if (i + step < len) {
      colNames.push(str.substring(i, i + step));
    } else {
      colNames.push(str.substring(i, len));
    }
  }
  return colNames;
}
```

(e) JavaScript program with new identifier names and types

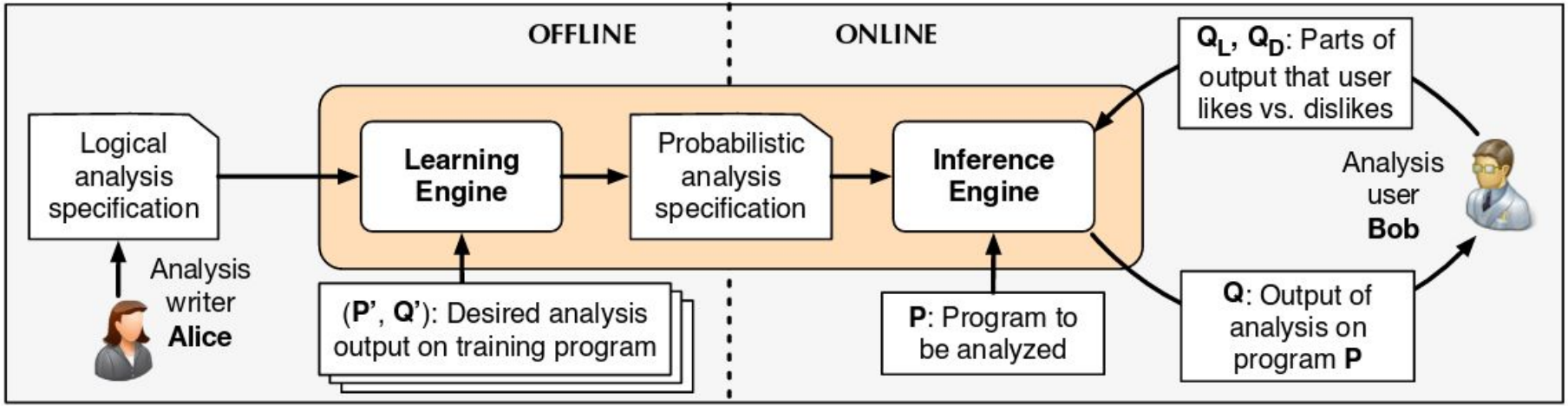
Structured Prediction Model

Raychev *et al.* POPL 2015



$$\psi = \mathsf{tree}(x, \lambda i_1, i_2, i_3, i_4 \rightarrow \exists t.\mathsf{ls}(i_2, t, \lambda i_5, i_6, i_7, i_8 \rightarrow \top) * \mathsf{ls}(t, t, \lambda i_9, i_{10}, i_{11}, i_{12} \rightarrow \top)).$$

Li *et al.* ICLR 2016 Brockschmidt *et al.* CML 2015



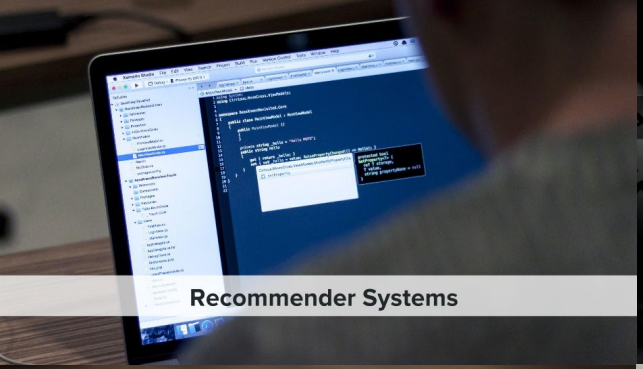
Mangal *et al.* FSE 2015

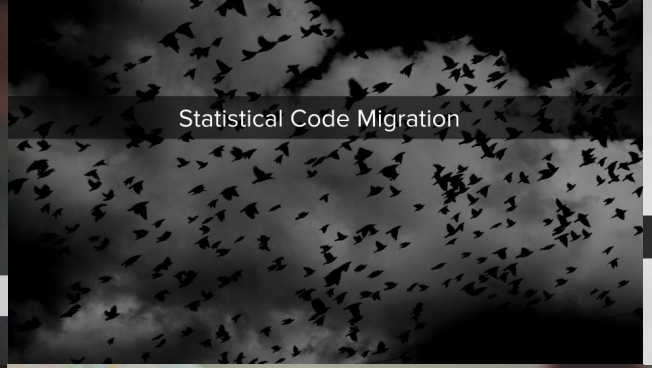
Probabilistic Models of Source Code

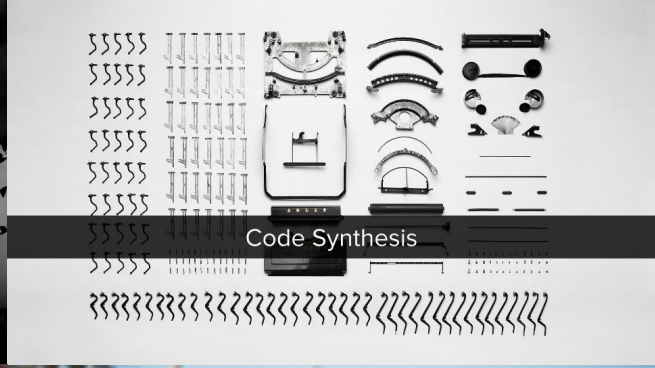
What was *not* discussed

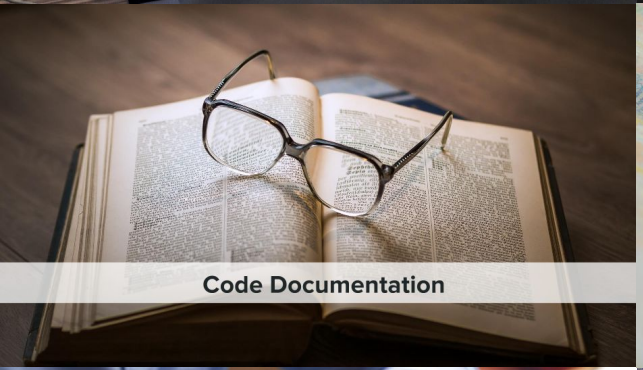
- Non-*Probabilistic* Models
- Models with *no* Learning Component
- Pure Information Retrieval Models
- Off-the-shelf models: topic models, *etc.* with hand-crafted features
- Bag-of-Word Representations
- Things that don't Involve Code

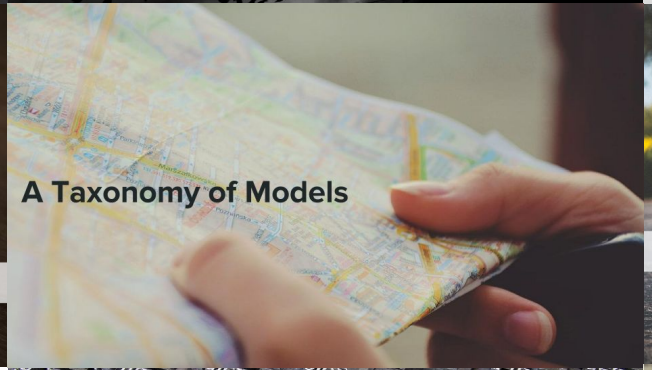

Challenges Ahead...


Recommender Systems

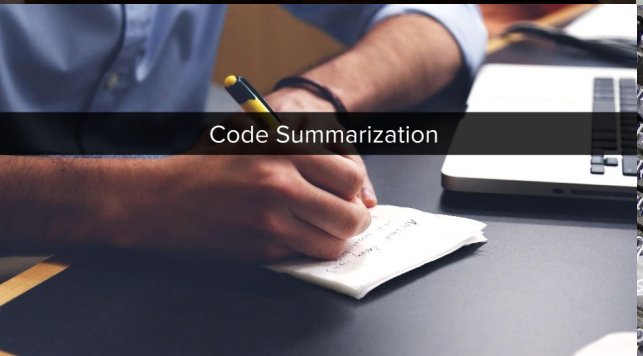

Statistical Code Migration

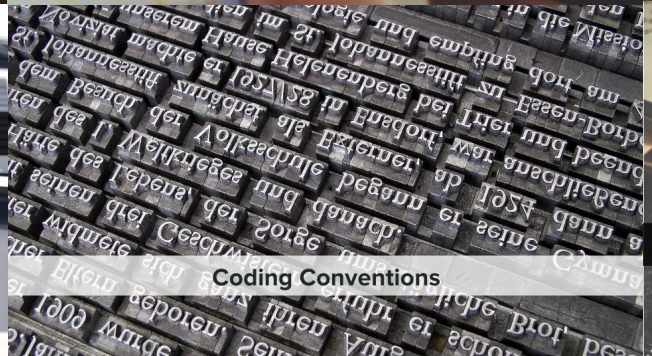

Code Synthesis

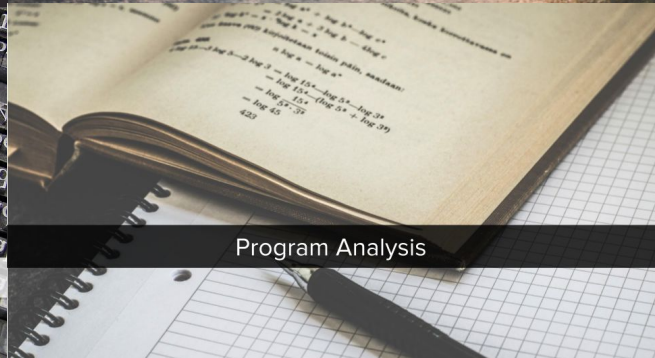

Code Documentation


A Taxonomy of Models


Code Defects


Code Summarization


Coding Conventions


Program Analysis


(the end)