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.
A Spectrum of Problems for Machine Learning

Classification
- Supervised

Clustering
- Unsupervised
A Spectrum of Problems for Machine Learning

Joint Classification

Learning Features
Natural Language Processing with Machine Learning

Resolve language ambiguities with principled probabilistic models of language.

Learn model parameters from annotated corpora.
Natural Language Processing (NLP)

Some **Knowledge** of Linguistics

**Data:** Corpora of Text, Speech etc

Use Machine Learning to **model** aspects of a natural language.
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
Language Models for Source Code

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

Probabilities learned from training corpus

```java
for (int i = 0; i < nProperties; i++) {
    final List<TreeNode<TSGNode>> children = node.getChildrenByProperty().get(i);
    final int nChildren = children.size();
    ruleConsequent.nodes.add(Lists.<Integer> newArrayListWithCapacity(nChildren));
    for (int j = 0; j < nChildren; j++) {
        final int childNode = node.getChild(j, i).getData().nodeKey;
        ruleConsequent.nodes.get(i).add(childNode);
    }
}
return new CFGRule(rootId, ruleConsequent);
}

public BlockedPosteriorComputer getPosteriorComputer() {
    return samplePosteriorComputer;
}

public final CFGPrior getPrior() {
    return prior;
}

public void lockSamplerData() {
    prior.lockPrior();
    burninPosteriorComputer.getPrior().cfg = samplePosteriorComputer.getPrior().cfg;
}
```
Language Models of Source Code – Design Choices

Token-level Models

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

Syntactic Models
N-gram Language Models

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

Parameters of ML Model

\[ P(t_m | t_{m-1} \ldots t_{m-n+1}) \]

e.g. \( P(\theta | \text{"for (int i ="}) \)
How n-gram models see code?

```java
package org.cfeclipse.cfml.snippets;

import org.rioproject.examples.logicdesigner.model.getState ( ) {
    cdl.Choreography;
}

import org.apache.thrift.protocol.TProtocolUtil.skip(iprot);

if ( event.newLineCount == 3 )
    case '|':
        if ( rule.FireAllRulesCommand;
            import org.apache.hadoop.conf.get(0, 0, newByteBuffer, 0,
            count);
        }

    switch ( classifierID ) {
        pd.getName() {
            cBondNeighborsB.get(MODULE).declaringType
                = (DEREnumerated)
```
**Machine Learning**

**Learn** the parameters of the model from data.

**Handle uncertainty** and **noise**.

**Machine Learning Model**
- Designed by humans

**Model Parameters**
- Learned from data
Learning Model Parameters

- **Optimize** objective function in training set
- Use **computational methods** of optimization
Finding a good model

High Bias  
Underfitting

High Variance  
Overfitting

image from http://antianti.org/?p=175
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
Source Code and Machine Learning

Coding Patterns
Mine & exploit common patterns

Formal Methods
Probabilities over Search Space (e.g. Synthesis)
[Ellis et al. 2015]

Code & Text
Code search, NL to Code

Probabilistic Static Analyses
Probability Distribution of (Formal) Properties
[Raychev et al. 2015, Mangal et al. 2015]

Runtime Traces
Infer Program Properties from Traces
[Brockschmidt et al. 2014, Yujia Li et al. 2015]
Outline

Learning Naming Conventions
  › Lexical Patterns

Learning to Map Natural Language to Source Code
  › Syntactic Patterns
“Programs must be written for people to read, and only incidentally for machines to execute.”

- Abelson & Sussman, SICP, preface to the first edition

Learning Naming Conventions
A coding convention is a syntactic constraint beyond those imposed by the language grammar.

Allamanis et. al, FSE 2014, FSE 2015
ACM Distinguished Paper Award
The Importance of Coding Conventions

Based on 169 code reviews with 1,093 discussion threads in Microsoft.

<table>
<thead>
<tr>
<th>Code Review Discussions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventions</td>
<td>38%</td>
</tr>
<tr>
<td>Naming</td>
<td>24%</td>
</tr>
<tr>
<td>Formatting</td>
<td>9%</td>
</tr>
</tbody>
</table>

[Allamanis et al. FSE 2014]
## The Importance of Coding Conventions

<table>
<thead>
<tr>
<th>Coding Conventions</th>
<th>Code Review Discussions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventions</td>
<td>38%</td>
</tr>
<tr>
<td><strong>Naming</strong></td>
<td>24%</td>
</tr>
<tr>
<td>Formatting</td>
<td>9%</td>
</tr>
</tbody>
</table>

Based on 169 code reviews with 1,093 discussion threads in Microsoft.

[Allamanis et al. FSE 2014]
Is recommending identifier renamings useful?

![Graph showing percentage of developers' responses to the question: Definitely YES 31%, Probably YES 37%, Probably NO 17%, Definitely NO 1%]
A Machine Learning Perspective

A name reflects important aspects of code functionality.

Learning to name source code elements is a first step in understanding code through machine learning.
public class TextRunnerTest extends TestCase {
    void execTest(String testClass, boolean success) throws Exception {
        ...
        InputStream i = p.getInputStream();
        while ((i.read()) != -1);
        ...
    }
    ...
}
public class TextRunnerTest extends TestCase {
    void execTest(String testClass, boolean success) throws Exception {
        ...
        InputStream i = p.getInputStream();
        while ((i.read()) != -1);
        ...
    }
    ...
}
Suggestions for junit/src/test/java/junit/tests/runner/TextRunnerTest.java

public class TextRunnerTest extends TestCase {
    void execTest(String testClass, boolean success) throws Exception {
        ...
        InputStream i = p.getInputStream();
        while ((i.read()) != -1);
        ...
    }
    ...
}

1. 'i' (18.07%) -> {input(81.93%), }
Suggesting Names to Developers: The Naturalize Framework

[Allamanis et al. FSE 2014, FSE 2015]
Naturalize Tools - devstyle

devstyle suggests identifier renamings
18 patches for 5 well known open source projects:
14 accepted, 4 ignored
Method Naming Problem

```java
private void () {
    String vertexShader = "literal_1";
    String fragmentShader = "literal_2";
    shader = new ShaderProgram(vertexShader,
                                fragmentShader);
    if(shader.isCompiled() == false)
        throw new IllegalArgumentException(
            "literal_3" + shader.getLog());
}
Method Naming Problem

```java
private void () {
    String vertexShader = "literal_1";
    String fragmentShader = "literal_2";
    shader = new ShaderProgram(vertexShader, fragmentShader);
    if (shader.isCompiled() == false)
        throw new IllegalArgumentException("literal_3" + shader.getLog());
}
```

Names describe what it does not what it is.
Models need to be “non-local”
private void createShader() {
    String vertexShader = "literal_1";
    String fragmentShader = "literal_2";
    shader = new ShaderProgram(vertexShader, fragmentShader);
    if (shader.isCompiled() == false)
        throw new IllegalArgumentException("literal_3" + shader.getLog());
}
Method Naming Problem

```java
private void createDefaultShader () {
  String vertexShader = "literal_1";
  String fragmentShader = "literal_2";
  Shader shader = new ShaderProgram(vertexShader, fragmentShader);
  if (shader.isCompiled() == false)
    throw new IllegalArgumentException("literal_3" + shader.getLog());
}
```

Suggestions:
- create
- create?UNK?
- init
- createShader
A Machine Learning Model of Names

\[ P(t|c) = \frac{\exp(s_\theta(t, c))}{\sum_{t'} \exp(s_\theta(t', c))} \]

Embedding Identifiers

$q_t \in \mathbb{R}^D$ are “embeddings” ::: model parameters

Embedding Identifiers

\[ s_\theta(t, c) = \hat{r}_{context}^\top q_t + b_t \]

\[ P(t|c) = \frac{\exp(s_\theta(t, c))}{\sum_{t'} \exp(s_\theta(t', c))} \]

Neural Context Models of Source Code

Variable: \texttt{isDone}

\[ s_{\theta}(.) = \hat{r}_{\text{context}}^T q_{\text{isDone}} + b_{\text{isDone}} \]
Neural Context Models of Source Code

Variable: $\text{isDone}$

$$s_\theta(.) = \hat{r}_{\text{context}} \cdot q_{\text{isDone}} + b_{\text{isDone}}$$
Neural Context Models of Source Code

Variable: \texttt{isDone}

Features:
- boolean, \texttt{in:MethodBody}, \texttt{final}

\[ s_\theta(.) = \hat{r}_{context}^T q_{isDone} + b_{isDone} \]
Neural Context Models of Source Code

Variable:  \textit{isDone}

Features:
- boolean, in:MethodBody, final

Contexts:
- while(!isDone) 

\[ s_{\theta}(.) = \hat{r}_{context}^{T} q_{isDone} + b_{isDone} \]
Neural Context Models of Source Code

Variable: \( \text{isDone} \)

Features:
- boolean, in:MethodBody, final

\[ r_{\text{boolean}} + r_{\text{in:MethodBody}} + r_{\text{final}} \]

Contexts:
while ( !isDone ) {
  ... 
}

\[ C_{-2}r + C_{-1}r + C_{1}r + C_{2}r \]

\[ s_\theta(.) = \hat{r}_{\text{context}}^T q_{\text{isDone}} + b_{\text{isDone}} \]
Neural Context Models of Source Code

Variable: \texttt{isDone}

Features:
- boolean, in:MethodBody, final

Contexts:
- while(\(!\texttt{isDone}\)) {
  \(-2r_{-2} + (-1)r_{-1} + r_1 + r_2\)

Neural Context Models of Source Code

\[ s_\theta(\cdot) = \hat{r}_{\text{context}}^T q_{\text{isDone}} + b_{\text{isDone}} \]

\[ \hat{r}_{\text{context}} = \sum_{f \in F_{tc}} r_f + \sum_{\forall k: K \geq |k| > 0} C_k r_{t+k} \]

Variable: \text{isDone}

Features:
- boolean
- in:MethodBody
- final

Contexts:
\[ \text{while} \ (\! \text{isDone} \ ) \{ \]
- \( C_{-2} r \)
- \( C_{-1} r \)
- \( C_1 r \)
- \( C_2 r \)
Neural Context Models of Source Code

Variable: $\text{isDone}$

Features:
- $\text{boolean}$
- $\text{in:MethodBody}$
- $\text{final}$

$\hat{r}_{\text{context}} = \sum_{f \in F_{ic}} r_f + \sum_{\forall k: |k| > 0} C_k r_{t_{i+k}}$

Global Information
Neural Context Models of Source Code

Variable: \( \text{isDone} \)

Features:
- boolean
- in:MethodBody
- final

\[
\hat{r}_{\text{context}} = \sum_{f \in F_{tc}} r_f + \sum_{\forall k: K \geq |k| > 0} C_k r_{t_i+k}
\]

Contexts:
- while (!isDone) {

Local Information
Neural Context Models of Source Code

Variable: \texttt{isDone}

Features:
- boolean
- \texttt{in:MethodBody}
- final

\[
\hat{r}_{\text{context}} = \sum_{f \in F_{tc}} r_f + \sum_{\forall k: K \geq |k| > 0} C_k r_{t_i+k}
\]

\[
s_{\theta}(.) = \hat{r}_{\text{context}}^T q_{\text{isDone}} + b_{\text{isDone}}
\]

Contexts:
- while ( ! \texttt{isDone} )
- \texttt{C_2r} + \texttt{C_1r} + \texttt{C_1r} + \texttt{C_2r}
Embedding Identifiers

\[ P(t|c) = \frac{\exp(s_\theta(t, c))}{\sum_{t'} \exp(s_\theta(t', c))} \]

[Please refer to the sources for detailed explanations.]
Neologisms

neologism

[ nɪˈɒlədʒɪz(ə)m ]

NOUN

a newly coined word or expression.
synonyms: new word · new expression · new term · new phrase · coinage · More

Powered by OxfordDictionaries · © Oxford University Press
Subtoken Context Models of Code

Sequentially predict each subtoken given the context and the previous subtokens

\[ P(t_i | t_{i-1}, t_{i-2}, context) \]
Training Data (project) → Train Neural Network → Embeddings → Suggest Names on Test Data
Evaluation Methodology

Test File

ForkJoinTask<? super T> task;
if (task instanceof ForkJoinTask<?>) // avoid re-wrap
    task = (ForkJoinTask<?>) task;
else
    task = new ForkJoinTask.AdaptedRunnableAction(task);
    externalPush(task);

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

Evaluation on top 10 Java GitHub projects.

Perturb existing code and retrieve ground truth.
Evaluation Methodology

Test File
ForkJoinTask<?> job;
if (task instanceof ForkJoinTask<?>) // avoid re-wrap
    job = (ForkJoinTask<?>) task;
else
    job = new ForkJoinTask.AdaptedRunnableAction(task);
externalPush(job);

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

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

compare with ground truth

Evaluation on top 10 Java GitHub projects.
Perturb existing code and retrieve ground truth.
Suggesting Variable Names

![Graph showing the comparison of different suggestion methods: ngram, context model, and subtoken. The x-axis represents the suggestion frequency, and the y-axis represents the F1 score. The graph shows the performance of each method across various suggestion frequencies.](image)
Suggesting Variable Names

The graph illustrates the F1 score (Suggestion F1) as a function of suggestion frequency. The graph compares different methods for naming suggestions:

- **ngram**
- **context model**
- **subtoken**

The x-axis represents the suggestion frequency, ranging from 0.0 to 0.7. The y-axis shows the Suggestion F1 score, ranging from 0.0 to 1.0. The graph shows the performance of each method across different suggestion frequencies, with each method represented by a distinct line color.
Suggesting Method Names

The graph above illustrates the performance of different methods in suggesting method names based on suggestion frequency. The x-axis represents the suggestion frequency, while the y-axis shows the F1 score. The three methods compared are:

- **ngram**: represented by a green dotted line.
- **context model**: represented by a black dashed line.
- **subtoken**: represented by a red solid line.

The ngram method shows a significant drop in F1 score as the suggestion frequency increases, indicating a decline in performance. The context model maintains a relatively high F1 score across the frequency range but shows a slight decrease. The subtoken method exhibits a more stable F1 score, indicating consistent performance across different suggestion frequencies.
Suggesting Method Names

![Graph showing the performance of different methods for suggesting method names.](image)

- **ngram**: A solid red line.
- **context model**: A dotted black line.
- **subtoken**: A dashed green line.

The graph compares the performance of these methods based on suggestion F1 scores against suggestion frequency.
Embedding Visualization

http://groups.inf.ed.ac.uk/cup/naturalize
Embedding Visualization

http://groups.inf.ed.ac.uk/cup/naturalize
Embedding Visualization

http://groups.inf.ed.ac.uk/cup/naturalize
Learning to map natural language to source code

Work done in Microsoft Research - Cambridge

Joint work with Danny Tarlow, Yi Wei, Andy Gordon
Applications of Joint Models of Code & NL

Code Retrieval

NL Retrieval

for Source Code

and eventually code synthesis...
A Conditional Generative Model

“get the first letter of each word in string and uppercase”

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);
}
Syntactic model of source code, i.e. model how AST is generated
Tree Generation Model: Context Free Grammars (CFG)

\[ E \rightarrow E + E \]
\[ E \rightarrow T \]
\[ F \rightarrow (E) \]
\[ T \rightarrow F \ast F \]
\[ T \rightarrow F \]
\[ F \rightarrow id \]
Tree Generation Model: Probabilistic Context Free Grammars (PCFG)

\[
E \rightarrow E + E \quad \text{(prob 0.7)} \\
E \rightarrow T \quad \text{(prob 0.3)} \\
F \rightarrow (E) \quad \text{(prob 0.1)} \\
T \rightarrow F \ast F \quad \text{(prob 0.6)} \\
T \rightarrow F \quad \text{(prob 0.4)} \\
F \rightarrow id \quad \text{(prob 0.9)}
\]
Generating from a PCFG
Generating from a PCFG

ForStatement
  Initialization
  Expression
  Expression
  Body
Generating from a PCFG

ForStatement

- Initialization
- Expression
- Expression
- Body

Single Variable Declaration
Generating from a PCFG

ForStatement

- Initialization
- Expression
- Expression
- Body

Single Variable Declaration

- Type
- Name
- Initializer
Generating from a PCFG

ForStatement

Initialization

Expression

Expression

Body

Single Variable Declaration

Type

Name

Initializer

int
Generating from a PCFG

ForStatement

Initialization

Single Variable Declaration

Type

Name

Initializer

Expression

Expression

Body

int

i

0

<

10
Conditional Generative Model of Source Code

Given natural language, get a model that can generate (probabilistically) source code, i.e.

\[ P(\text{code} | \text{natural language}) \]
A Neural Log-Bilinear Bimodal Model of Code

Kiros, Ryan, Ruslan Salakhutdinov, and Rich Zemel. "Multimodal neural language models."

Maddison, Chris and Daniel Tarlow. "Structured generative models of natural source code."
A Neural Log-Bilinear Bimodal Model of Code

$\mathcal{L}$

$\mathbf{s}_\theta(\mathbf{v}, \mathcal{L}, \mathcal{C}_{\leq n}) = (\mathbf{l} \odot \mathbf{c})^T \mathbf{r} + b_{n \rightarrow v}$

Kiros, Ryan, Ruslan Salakhutdinov, and Rich Zemel. "Multimodal neural language models."
Maddison, Chris and Daniel Tarlow. "Structured generative models of natural source code."
A Neural Log-Bilinear Bimodal Model of Code

\[ P(v | \mathcal{L}, \mathcal{C}_{\leq n}) \propto \exp s_\theta(v, \mathcal{L}, \mathcal{C}_{\leq n}) \]

\[ s_\theta(v, \mathcal{L}, \mathcal{C}_{\leq n}) = (l \diamond c)^T r + b_{n \rightarrow v} \]

Kiros, Ryan, Ruslan Salakhutdinov, and Rich Zemel. "Multimodal neural language models."
Maddison, Chris and Daniel Tarlow. "Structured generative models of natural source code."
Natural Language “Query” — Make first letter of a string upper case

Code Snippets

```csharp
public static string FirstCharToUpper(string input)
{
    if (String.IsNullOrEmpty(input))
        throw new ArgumentException("ARG!");
    return input.First().ToString().ToUpper() + String.Join("", input.Skip(1));
}
```

EDIT: This version is shorter. For a faster solution take a look at Equino's answer.
How do I enumerate an enum

```csharp
foreach (Suit suit in (Suit[]) Enum.GetValues(typeof(Suit)))
{
}
```

C# Configuration Manager. ConnectionStrings

Check your `machine.config`. If you only want your entry, you can add a `<clear />` element to the `<connectionStrings>` element like so...

```xml
<connectionStrings>
  <clear />
  <add name="Target" connectionString="server=MYSERVER; Database=MYDB; Integrated Security=SSPI;" />
</connectionStrings>
```
How do I enumerate an enum

```csharp
foreach (Suit suit in (Suit[]) Enum.GetValues(typeof(Suit)))
{
}
```

http://stackoverflow.com/questions/105372/how-do-i-enumerate-an-enum
40,092 C# Snippets
6,355,393 Natural Language Queries
Performance Metric: Mean Reciprocal Rank

\[
MRR = \frac{1}{|Q|} \sum_{i=1}^{\left|Q\right|} \frac{1}{\text{RANK}(i)}
\]

**Measures** how well we rank the *correct* answer
# Retrieval Evaluation - MRR Performance

<table>
<thead>
<tr>
<th>Code Retrieval</th>
<th>Model</th>
<th>StackOverflow Test 1</th>
<th>StackOverflow Test 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NL+Code</td>
<td>0.18</td>
<td>0.17</td>
</tr>
<tr>
<td></td>
<td>NL only</td>
<td>0.12</td>
<td>0.13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query Retrieval</th>
<th>Model</th>
<th>StackOverflow Test 1</th>
<th>StackOverflow Test 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Multiplicative</td>
<td>0.43</td>
<td>0.41</td>
</tr>
<tr>
<td></td>
<td>NL only</td>
<td>0.25</td>
<td>0.26</td>
</tr>
</tbody>
</table>

**Test 1**: Code snippets from training set with new natural language queries.

**Test 2**: New code snippets and new natural language queries.
Synthesis Samples

> `timespan day the week`
```csharp
DateTime DateTime = DateTime.Now(0);
```

> `file exists on directory`
```csharp
var path = new File(directory)
```
Synthesis Samples

> timespan day the week
DateTime DateTime=DateTime.Now(0);

    foreach(string s in Days(new DateTime(2010, 1, 1), new DateTime(2010, 2, 1))) {
        Console.WriteLine(s);
    }

> file exists on directory
var path = new File(directory)

    System.IO.File.Exists(path)
Retrieval Sample

```csharp
path = Path.GetFullPathInternal(path);
new FileIOPermission(
    FileIOPermissionAccess.Read,
    new string[] { path },
    false, false).Demand();
flag = InternalExists(path);
```

1. wpf get directory name from path
2. determine a file exist on shared folder
3. open file dialog class
4. create directory pathname
5. load binary file to variable
Challenges

❯ Code Representations in Machine Learning
❯ Define Representative Evaluation Metrics for Software Engineering Tasks
❯ Create Useful and Efficient Software Engineering Tools
Understanding Source Code through Machine Learning to Create Smart Software Engineering Tools
Learning Naming Conventions
Allamanis et al, 2014; 2015

Learning to name source code

n-gram LMs for Code
Allamanis et al, 2013

Mining Source Code Idioms
Allamanis and Sutton, 2014
The Sympathetic Uniqueness Principle

**Rare names often usefully signify unusual functionality, and need to be preserved.**

- Prune rare words
- Repurpose special UNK token
- Allows Naturalize to decide when it should *not* suggest

```java
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);
}
```
Idioms vs. the Rest

**Code Clones** copy-paste code fragments

**API Patterns** usage patterns of methods
- J. Wang et al. Mining succinct and high-coverage API usage patterns from source code. MSR 2013.

**Idioms** syntactic code fragments
The Distributional Hypothesis

“You shall know a word by the company it keeps”.

John Rupert Firth, 1957
The Distributional Hypothesis

The ???????????? is walking

“You shall know a word by the company it keeps”.

John Rupert Firth, 1957
For IWESEP, in a way I'm inviting you as a ML/NLP person that can teach software engineering people the cool things you can do with ML/NLP techniques. I'm not sure if you see yourself this way, but I think a quick "intro to ML/NLP" for the first 1/3 or so, then "look at all the cool things you can do" for the second 2/3 would be one potential way to give the presentation.
Sanity Check:
String Manipulation Synthetic Data

```csharp
var result = input_string.Split(' ').Select((string x) =>
    Double.parseDouble(x)).Average();
```

each element parse double separated by a space and get mean
each element parse double separated by a space and get average
each element convert to double separated by a space and get mean
each element convert to double separated by a space and get average
each element parse to double separated by a space and get mean
A Neural Log-Bilinear Bimodal Model of Code

\[ c = \sum_{j=1}^{J} H_j c_{\phi_j} \]

\[ l = \frac{1}{|\mathcal{L}|} \sum_{w \in \mathcal{L}} l_w \]

\[ s_\theta(v, \mathcal{L}, \mathcal{C}_{\leq n}) = (l \odot c) r^T + b_{n \rightarrow v} \]

Kiros, Ryan, Ruslan Salakhutdinov, and Rich Zemel. "Multimodal neural language models."
Maddison, Chris and Daniel Tarlow. "Structured generative models of natural source code."