《code2vec: Learning Distributed Representations of Code》阅读报告

研究现状

word2vec，doc2vec的出现

研究目标

主要: 学习code embedding

具体来说，将函数编码，预测其函数名（跟代码变摘要有点像），有次学习得"副产品"code embedding

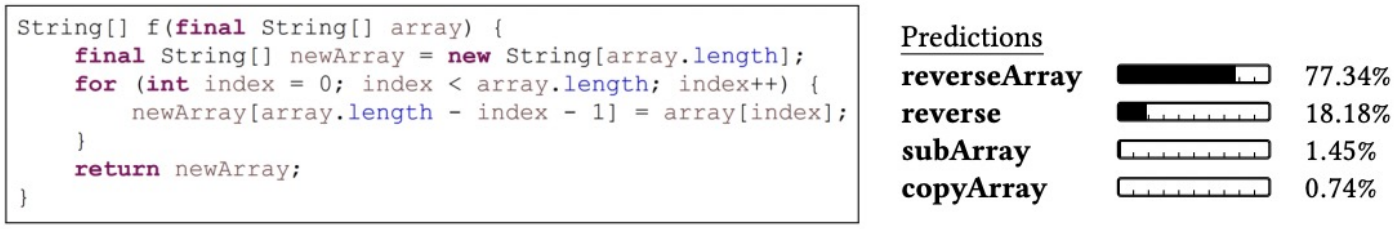


Fig. 1. A code snippet and its predicted labels as computed by our model.

研究方法

数据的生成

1. 生成AST
2. 函数体内的语句根据句法分析，生成一个个叶节点与叶节点之间的路径

如有：

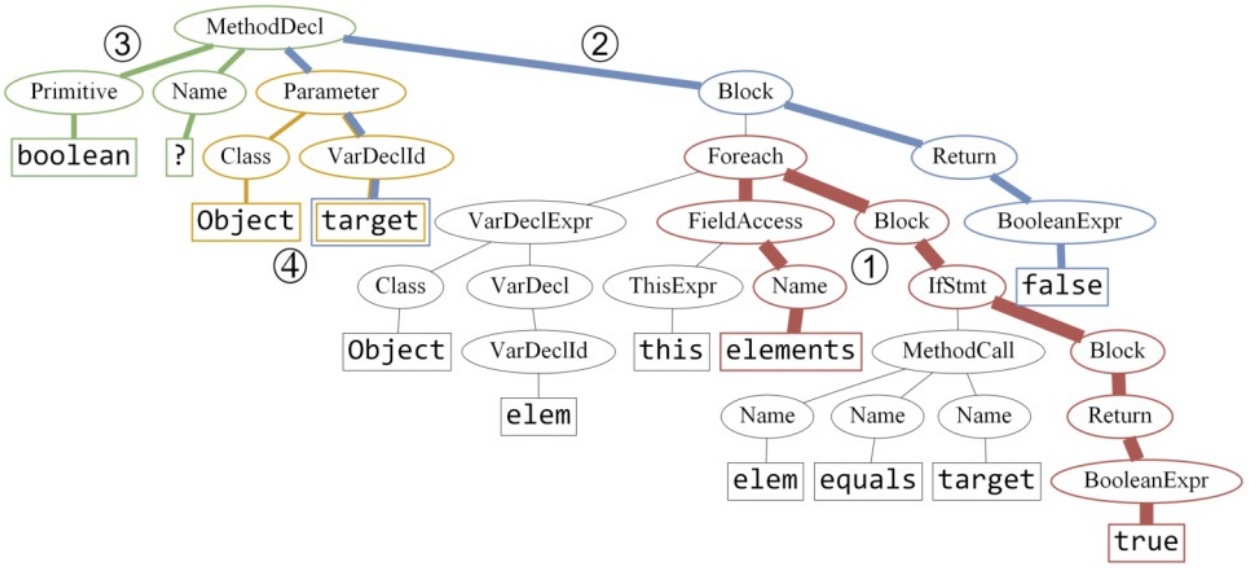


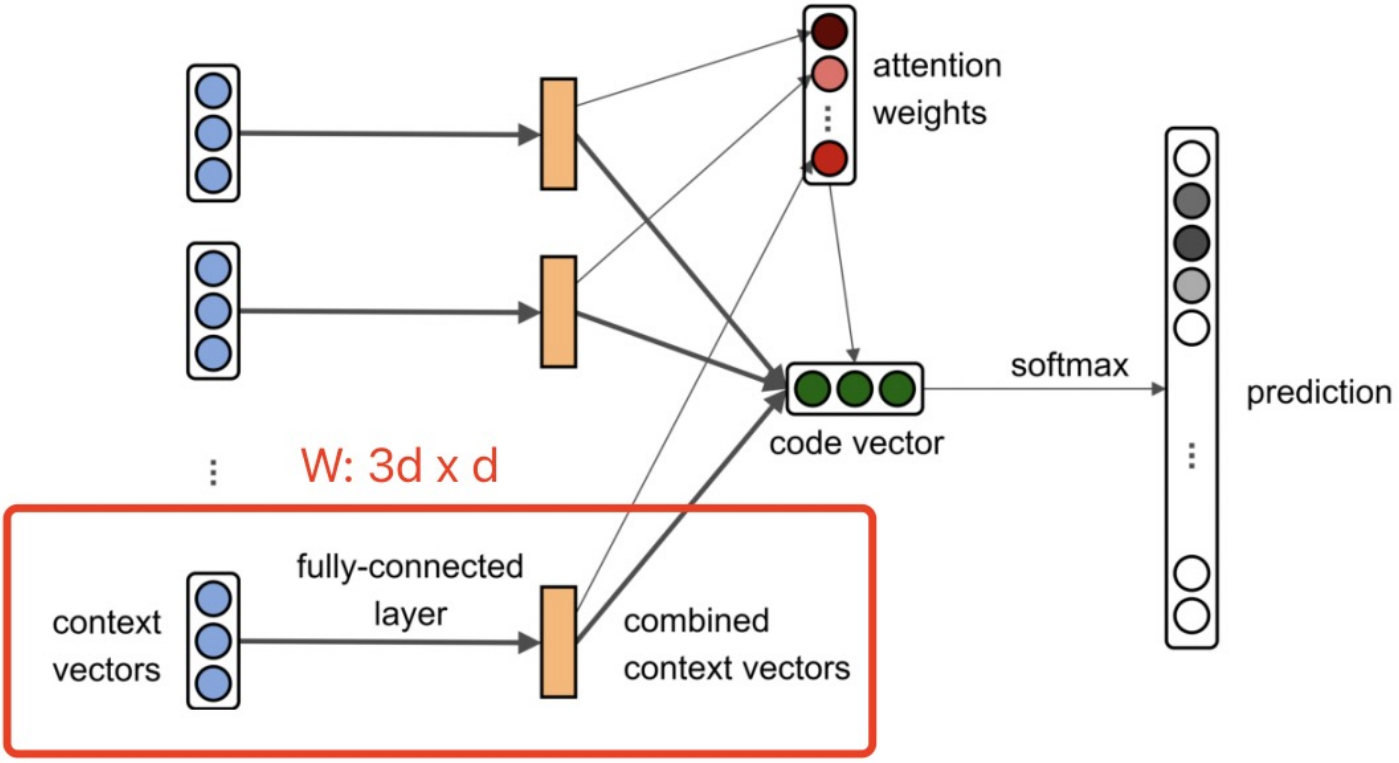
Fig. 3. The top-4 attended paths of Figure 2a, as were learned by the model, shown on the AST of the same snippet. The width of each colored path is proportional to the attention it was given (**red** ①: **0.23**, **blue** ②: **0.14**, **green** ③: **0.09**, **orange** ④: **0.07**).

形成一个path-contexts集：

$$(\texttt{elements}, \text{Name}\uparrow\text{FieldAccess}\uparrow\text{Foreach}\downarrow\text{Block}\downarrow\text{IfStmt}\downarrow\text{Block}\downarrow\text{Return}\downarrow\text{BooleanExpr}, \texttt{true})$$

- ↑表示结点沿AST路径向上， ↓则沿路径向下
- 为了限制训练数据的规模（不要过大）和减少数据的稀疏性，引入两个超参：path的最大长度和具有相同父结点的两个孩子结点序号的最大差值（path两端点的“宽度”）
- path-context其实由三个表示为一个三元组（x_s, p_j, x_t）

模型架构



1.将（x_s, p_j, x_t）三个向量（每个context vector, 大小1xd）拼接为一个一个（1x3d）的向量，输入全连接层，将其信息“压缩”为1xd向量(combined context vector)。

$$\tilde{c}_i = tanh(W \cdot c_i)$$

“This combination allows the model the expressivity of giving a certain path more attention when observed with certain values and less attention when the exact same path is observed with other value。”

2. 将多个path-context的组合向量聚合为一个1xd向量（code vector），以表示这段代码。

利用Attention机制：声明全局的注意力向量a，将每个path-context组合后的向量与a点积，赋予每个path-context对于这段代码特定的关注度（权重），该全局注意力向量也一起参与学习。

$$\text{attention weight } \alpha_i = \frac{\exp(\tilde{c}_i^{\,T} \cdot a)}{\sum_{j=1}^{n} \exp(\tilde{c}_j^{\,T} \cdot a)}$$

$$\text{code vector } \boldsymbol{v} = \sum_{i=1}^{n} \alpha_i \cdot \tilde{\boldsymbol{c}}_i$$

3. tag名（标注，函数名）也作embedding（y vector）,code vector与每个y vector做softmax，计算概率大小：

$$for\ y_i \in Y:\ q(y_i) = \frac{\exp(\boldsymbol{v}^T \cdot tags\_vocab_i)}{\sum_{y_j \in Y} \exp(\boldsymbol{v}^T \cdot tags\_vocab_j)}$$

4. 损失函数：交叉熵

$$\mathcal{H}(p||q) = -\sum_{y \in Y} p(y) \log q(y) = -\log q(y_{true})$$

结论

1.

Table 3. Evaluation comparison between our model and previous works.

Model	Sampled Test Set Precision	Recall	F1	Full Test Set Precision	Recall	F1	prediction rate (examples / sec)
CNN+Attention [Allamanis et al. 2016]	47.3	29.4	33.9	-	-	-	0.1
LSTM+Attention [Iyer et al. 2016]	27.5	21.5	24.1	33.7	22.0	26.6	5
Paths+CRFs [Alon et al. 2018]	-	-	-	53.6	46.6	49.9	10
PathAttention (this work)	**63.3**	**56.2**	**59.5**	**63.1**	**54.4**	**58.4**	**1000**

2.

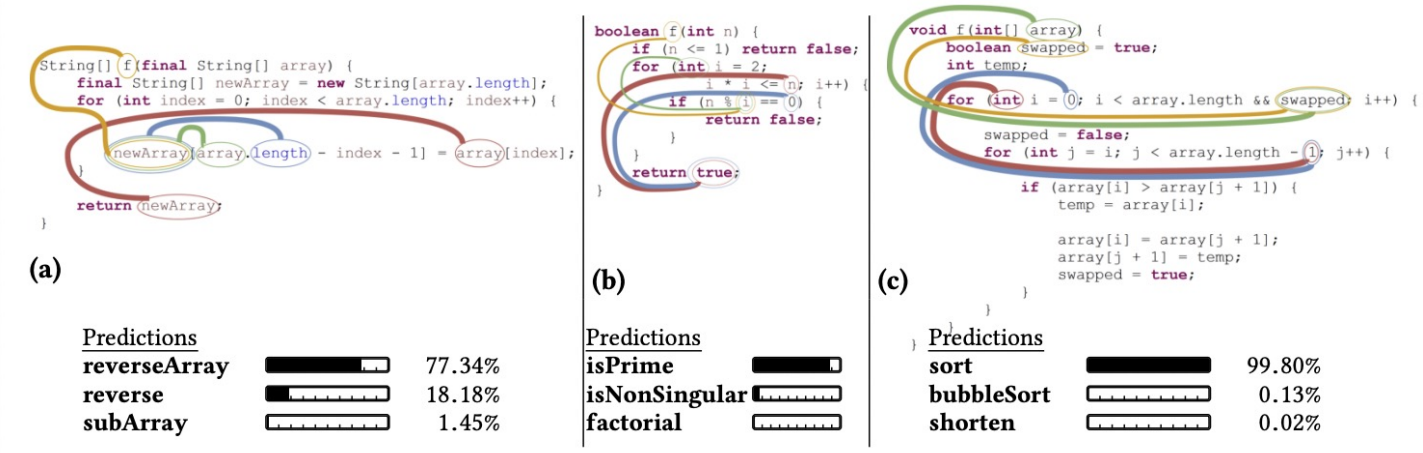
Table 4. Comparison of model designs.

Model Design	Precision	Recall	F1
No-attention	54.4	45.3	49.4
Hard attention	42.1	35.4	38.5
Train-soft, predict-hard	52.7	45.9	49.1
Soft attention	**63.1**	**54.4**	**58.4**
Element-wise soft attention	**63.7**	**55.4**	**59.3**

- Hard attention: 在与注意力向量点积后，只选取权值最高的combined context vector作为最终的code vector
- Element-wise soft attention: 使用d个全局注意力向量（d个数跟combined context vector维度相同）（感觉这里有点像transformer多头注意力的操作）

$$\alpha_{i_j} = \frac{\exp(\tilde{c_i}^T \cdot a_j)}{\sum_{k=1}^{n} \exp(\tilde{c_k}^T \cdot a_j)}$$

3. 注意力的可解析性（path-context的粗细）



4. code embedding的可解析性

- 语义的相似性

Table 1. Semantic similarities between method names.

A	≈B	A	≈B
size	getSize, length, getCount, getLength	executeQuery	executeSql, runQuery, getResultSet
active	isActive, setActive, getIsActive, enabled	actionPerformed	itemStateChanged, mouseClicked, keyPressed
done	end, stop, terminate	toString	getName, getDescription, getDisplayName
toJson	serialize, toJsonString, getJson, asJson,	equal	eq, notEqual, greaterOrEqual, lessOrEqual
run	execute, call, init, start	error	fatalError, warning, warn

- 语义的可组合性

Table 6. Semantic combinations of method names.

A	+B	≈C
get	value	getValue
get	instance	getInstance
getRequest	addBody	postRequest
setHeaders	setRequestBody	createHttpPost
remove	add	update
decode	fromBytes	deserialize
encode	toBytes	serialize
equals	toLowerCase	equalsIgnoreCase

- 语义的可类比性

Table 7. Semantic analogies between method names.

A :	B	C :	D
open :	connect	close :	disconnect
key :	keys	value :	values
lower :	toLowerCase	upper :	toUpperCase
down :	onMouseDown	up :	onMouseUp
warning :	getWarningCount	error :	getErrorCount
value :	containsValue	key :	containsKey
start :	activate	end :	deactivate
receive :	download	send :	upload

研究不足

1. 只能预测出在训练时已有的label
2. 数据稀疏性（数据饥饿），到时训练代码高

- 终止节点（叶节点的value）语义相近，却表示为不同的向量
- 只有单个不同的节点的path，却表示为不同向量
- target节点也有类似问题（语义相近，却向量不同）

3. 对变量名的依赖：用语义明确的变量名训练，预测准确性较高。用不太明确语义的变量名，预测的效果较差。（数据的偏见）

启发

1. 多个维度的“压缩”信息
2. 利用注意力生成代码向量
3. 注意力的可解析性（path-context的粗细）

附：

- 文献链接：https://dl.acm.org/doi/pdf/10.1145/3290353
- 代码：https://github.com/tech-srl/code2vec
- 相关工具：https://code2vec.org/