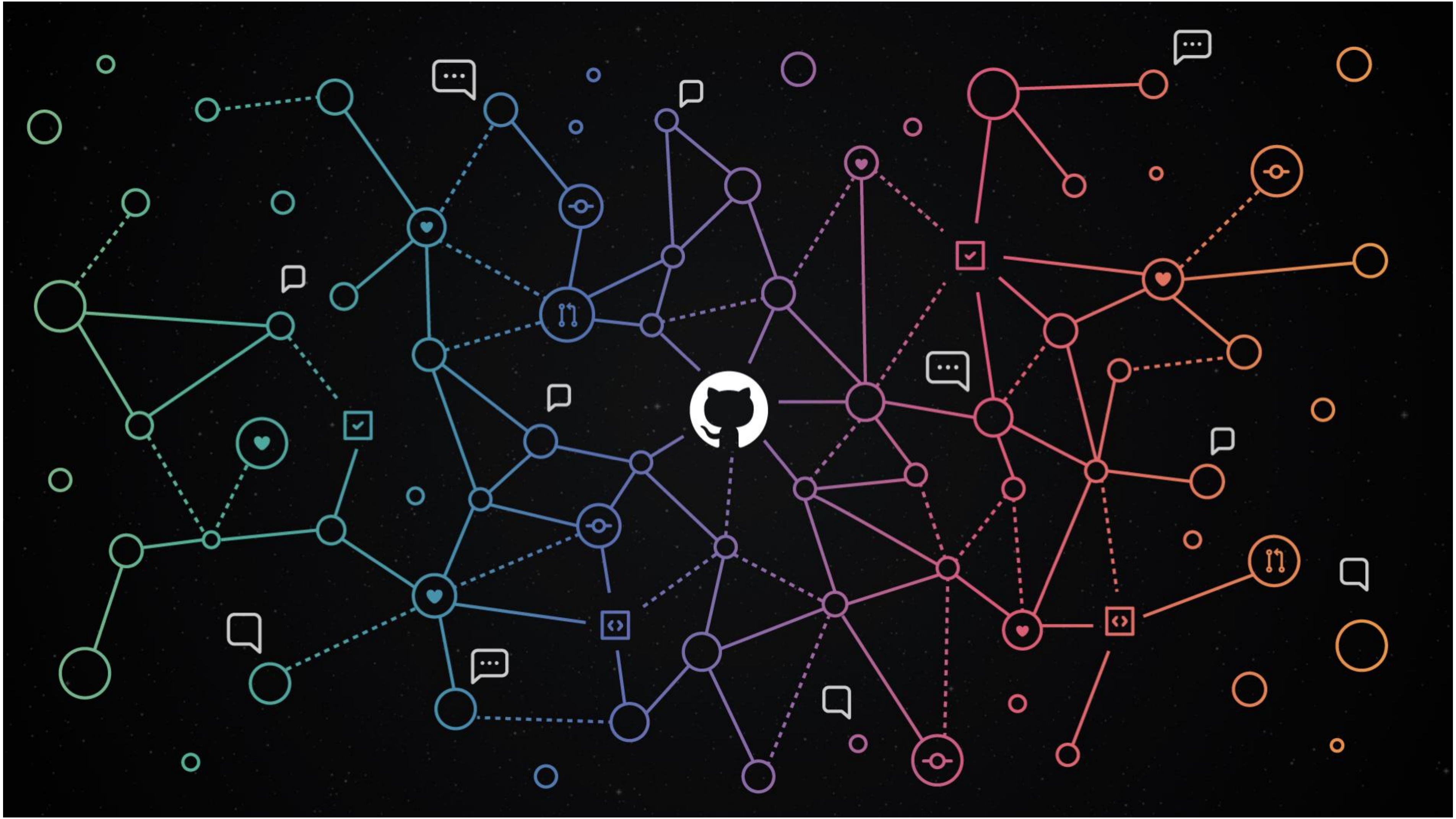
DATA INNOVATION SUMMIT 2019

GitHub and Deep Learning on Graphs of Code

Clair J. Sullivan, PhD
Machine Learning Engineer, GitHub



Outline

- Introduction to problem: detection of duplicate code
- Primer on types of duplicate code
- Our approach: machine learning on Abstract Syntax Trees (ASTs)
- Results
- Future work

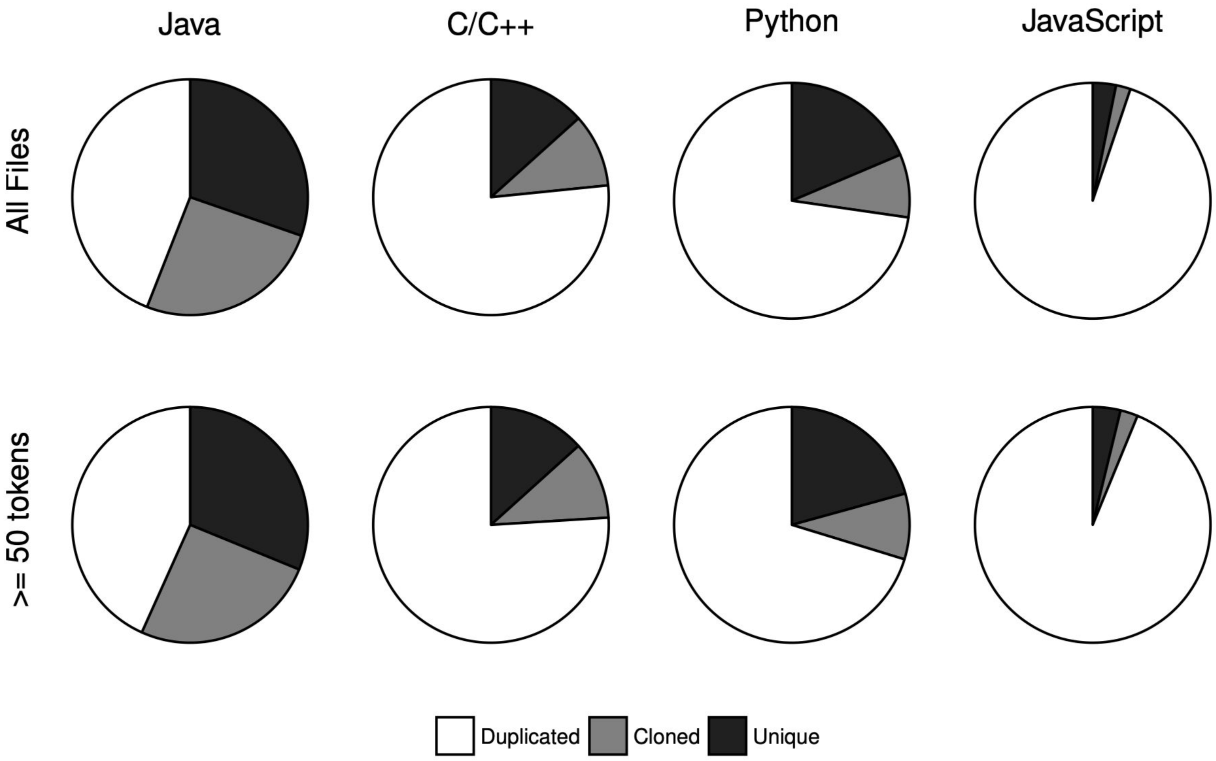


Fig. 5. File-level duplication for entire dataset and excluding small files.

DejuVu: A Map of Code Duplication on GitHub (Lopes et al., *Proc. ACM Program. Lang*. (1), 2017)

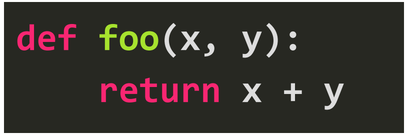
Code Duplication Primer

- **Type 0**: completely identical
- **Type 1**: only difference is in comments and whitespace
- **Type 2**: can also include variations in identifier names and literal values
- **Type 3**: syntactically similar with differences as the statement level (can be added, removed, or modified)
- **Type 4**: syntactically different but semantically similar

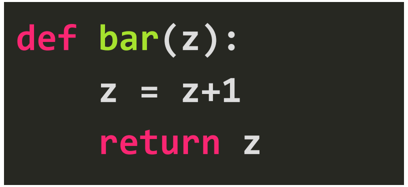
Code Duplication Primer

- **Type 0**: completely identical
- **Type 1**: only difference is in comments and whitespace
- **Type 2**: can also include variations in identifier names and literal values
- **Type 3**: syntactically similar with differences as the statement level (can be added, removed, or modified)
- **Type 4**: syntactically different but semantically similar

How similar are these functions?

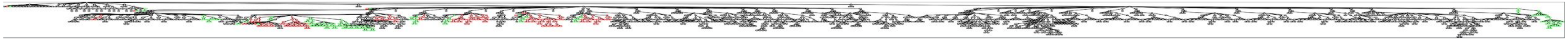


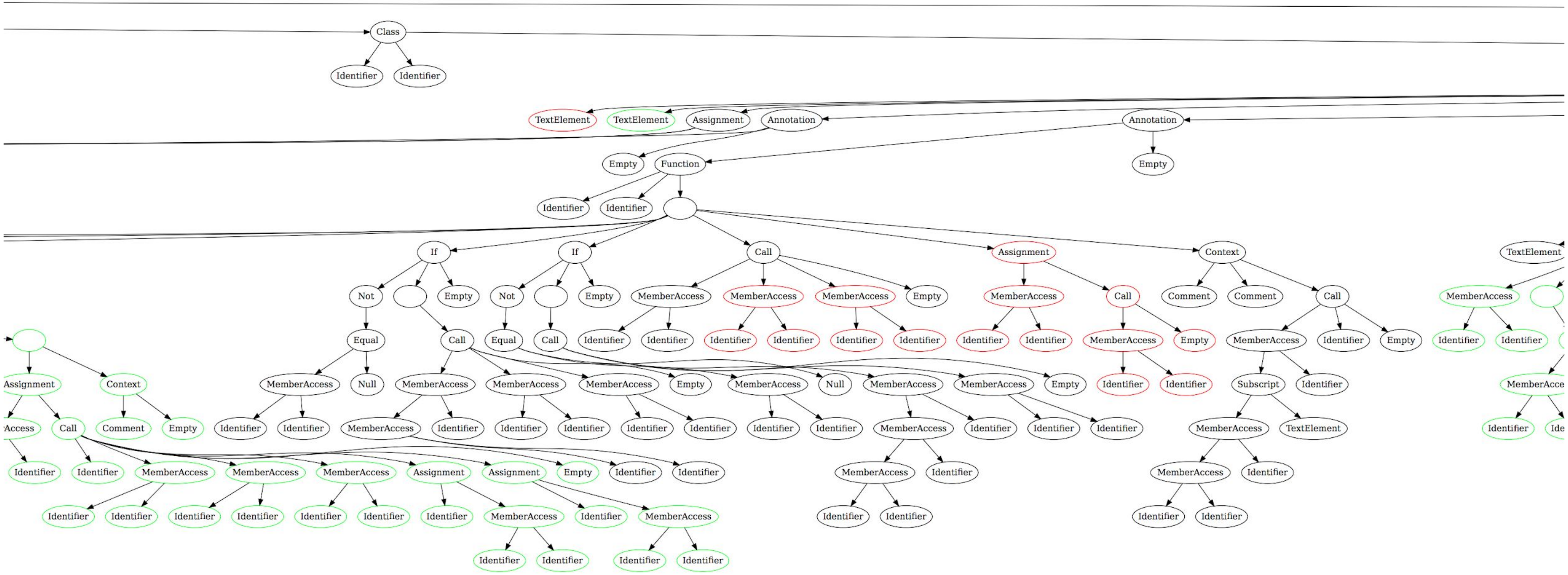
```python
def foo(x, y):
    return x + y
```



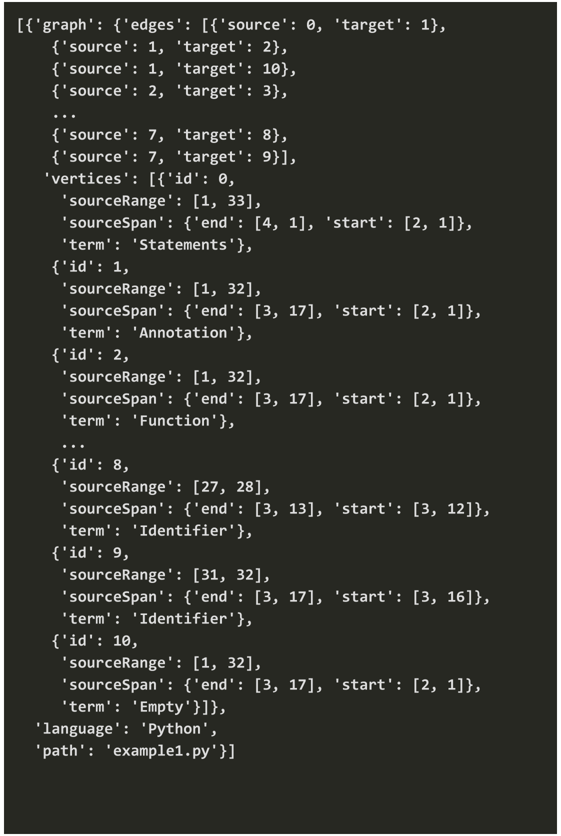
```python
def bar(z):
    z = z+1
    return z
```

ASTs are Directed, Acyclic Graphs




Class
Identifier Identifier
TextElement TextElement Assignment Annotation Annotation
Empty Function Empty
Identifier Identifier
If If Call Assignment Context TextElement
Not Empty Not Empty MemberAccess MemberAccess MemberAccess Empty MemberAccess Call Comment Comment Call MemberAccess
Equal Call Equal Call Identifier Identifier Identifier Identifier Identifier Identifier Identifier Identifier MemberAccess Empty MemberAccess Identifier Empty Identifier Identifier
Assignment Context MemberAccess Null MemberAccess MemberAccess MemberAccess Empty MemberAccess Null MemberAccess MemberAccess Empty Identifier Identifier Subscript Identifier MemberAcce
Access Call Comment Empty Identifier Identifier MemberAccess Identifier Identifier Identifier Identifier Identifier Identifier Identifier MemberAccess Identifier Identifier Identifier MemberAccess TextElement Identifier Ide
Identifier Identifier MemberAccess MemberAccess MemberAccess Assignment Assignment Empty Identifier Identifier MemberAccess Identifier MemberAccess Identifier
Identifier Identifier Identifier Identifier Identifier Identifier Identifier MemberAccess Identifier MemberAccess Identifier Identifier Identifier Identifier
Identifier Identifier Identifier Identifier

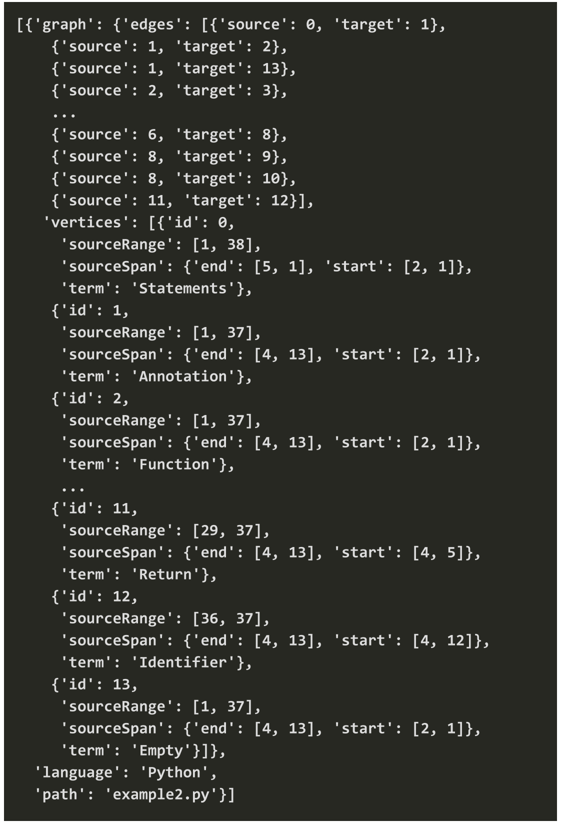
ASTs to Graphs



```
[{'graph': {'edges': [{'source': 0, 'target': 1},
    {'source': 1, 'target': 2},
    {'source': 1, 'target': 10},
    {'source': 2, 'target': 3},
    ...
    {'source': 7, 'target': 8},
    {'source': 7, 'target': 9}],
   'vertices': [{'id': 0,
     'sourceRange': [1, 33],
     'sourceSpan': {'end': [4, 1], 'start': [2, 1]},
     'term': 'Statements'},
    {'id': 1,
     'sourceRange': [1, 32],
     'sourceSpan': {'end': [3, 17], 'start': [2, 1]},
     'term': 'Annotation'},
    {'id': 2,
     'sourceRange': [1, 32],
     'sourceSpan': {'end': [3, 17], 'start': [2, 1]},
     'term': 'Function'},
     ...
    {'id': 8,
     'sourceRange': [27, 28],
     'sourceSpan': {'end': [3, 13], 'start': [3, 12]},
     'term': 'Identifier'},
    {'id': 9,
     'sourceRange': [31, 32],
     'sourceSpan': {'end': [3, 17], 'start': [3, 16]},
     'term': 'Identifier'},
    {'id': 10,
     'sourceRange': [1, 32],
     'sourceSpan': {'end': [3, 17], 'start': [2, 1]},
     'term': 'Empty'}]},
  'language': 'Python',
  'path': 'example1.py'}]
```

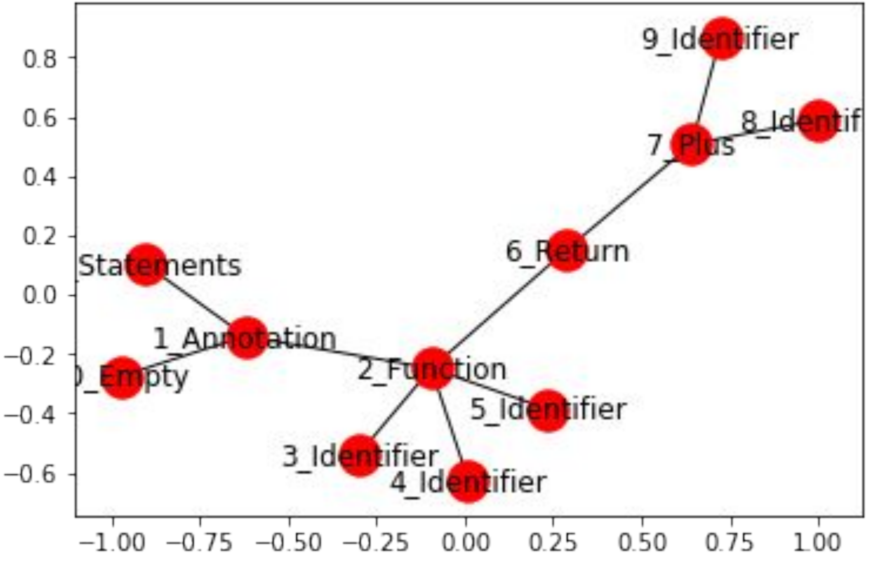
```
def foo(x, y):
    return x + y
```

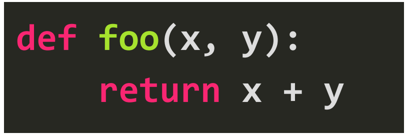
```
def bar(z):
    z = z+1
    return z
```



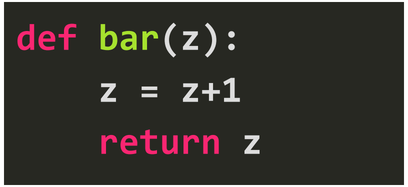
```
[{'graph': {'edges': [{'source': 0, 'target': 1},
    {'source': 1, 'target': 2},
    {'source': 1, 'target': 13},
    {'source': 2, 'target': 3},
    ...
    {'source': 6, 'target': 8},
    {'source': 8, 'target': 9},
    {'source': 8, 'target': 10},
    {'source': 11, 'target': 12}],
   'vertices': [{'id': 0,
     'sourceRange': [1, 38],
     'sourceSpan': {'end': [5, 1], 'start': [2, 1]},
     'term': 'Statements'},
    {'id': 1,
     'sourceRange': [1, 37],
     'sourceSpan': {'end': [4, 13], 'start': [2, 1]},
     'term': 'Annotation'},
    {'id': 2,
     'sourceRange': [1, 37],
     'sourceSpan': {'end': [4, 13], 'start': [2, 1]},
     'term': 'Function'},
     ...
    {'id': 11,
     'sourceRange': [29, 37],
     'sourceSpan': {'end': [4, 13], 'start': [4, 5]},
     'term': 'Return'},
    {'id': 12,
     'sourceRange': [36, 37],
     'sourceSpan': {'end': [4, 13], 'start': [4, 12]},
     'term': 'Identifier'},
    {'id': 13,
     'sourceRange': [1, 37],
     'sourceSpan': {'end': [4, 13], 'start': [2, 1]},
     'term': 'Empty'}]},
  'language': 'Python',
  'path': 'example2.py'}]
```

ASTs to Graphs (cont.)

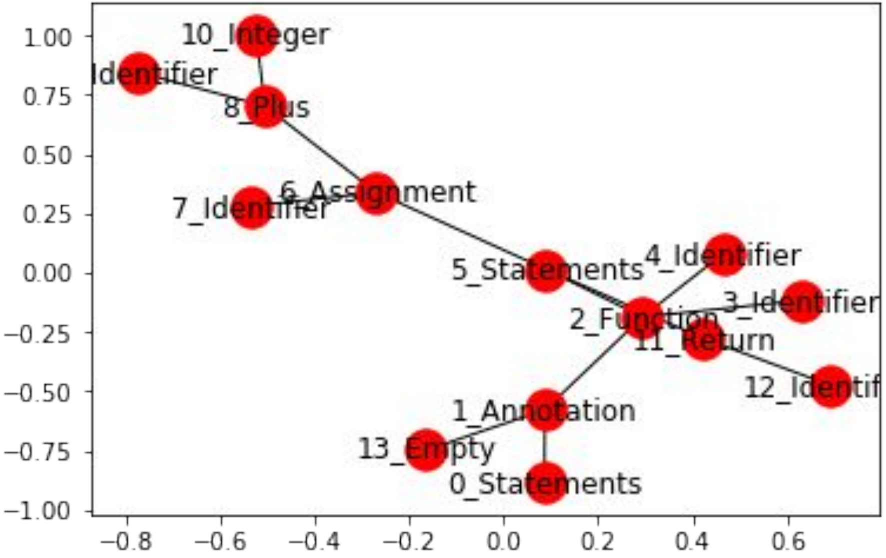




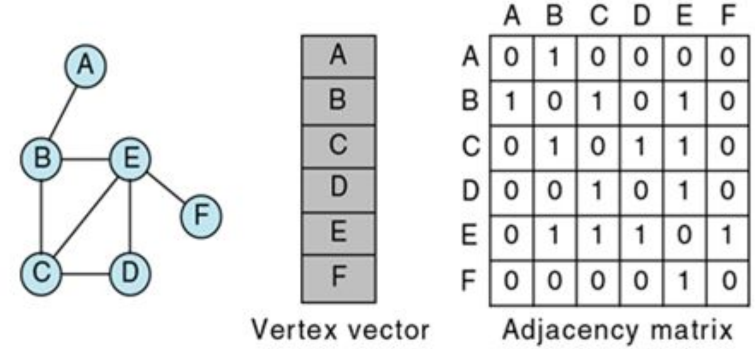
```
def foo(x, y):
    return x + y
```



```
def bar(z):
    z = z+1
    return z
```

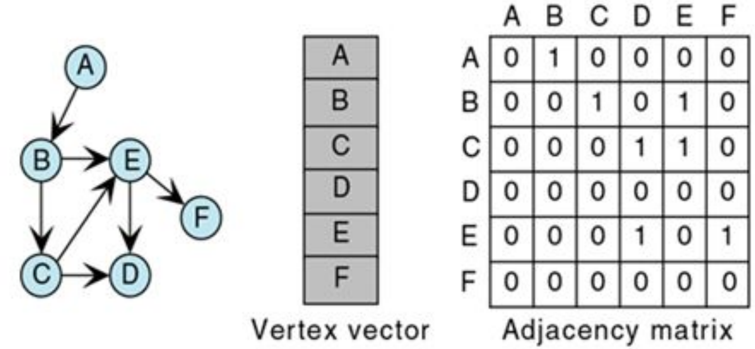


Graph calculations are typically done through the adjacency matrix...



	A	B	C	D	E	F
A	0	1	0	0	0	0
B	1	0	1	0	1	0
C	0	1	0	1	1	0
D	0	0	1	0	1	0
E	0	1	1	1	0	1
F	0	0	0	0	1	0

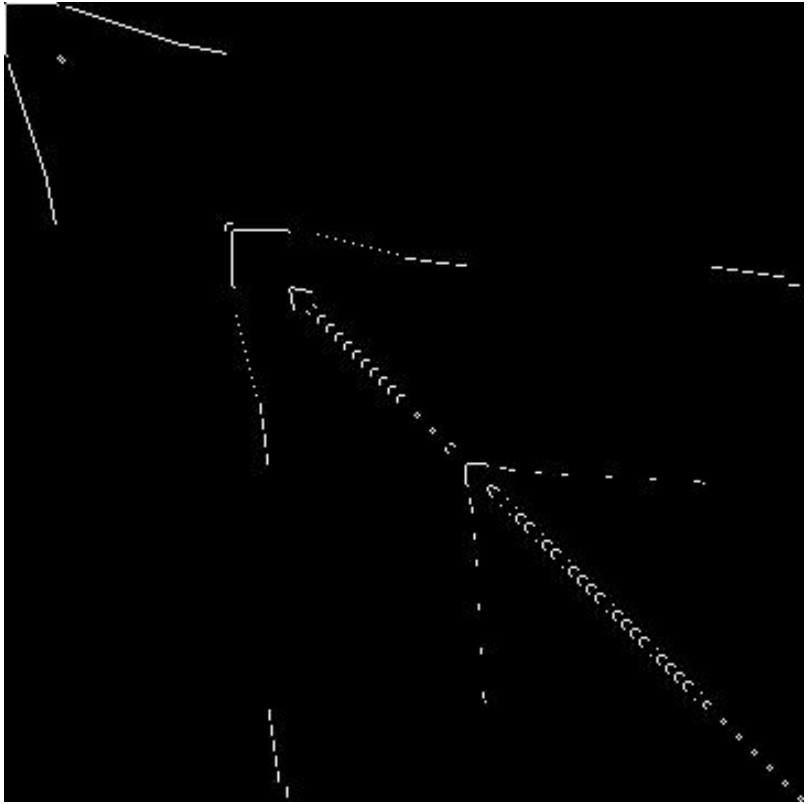
(a) Adjacency matrix for non-directed graph



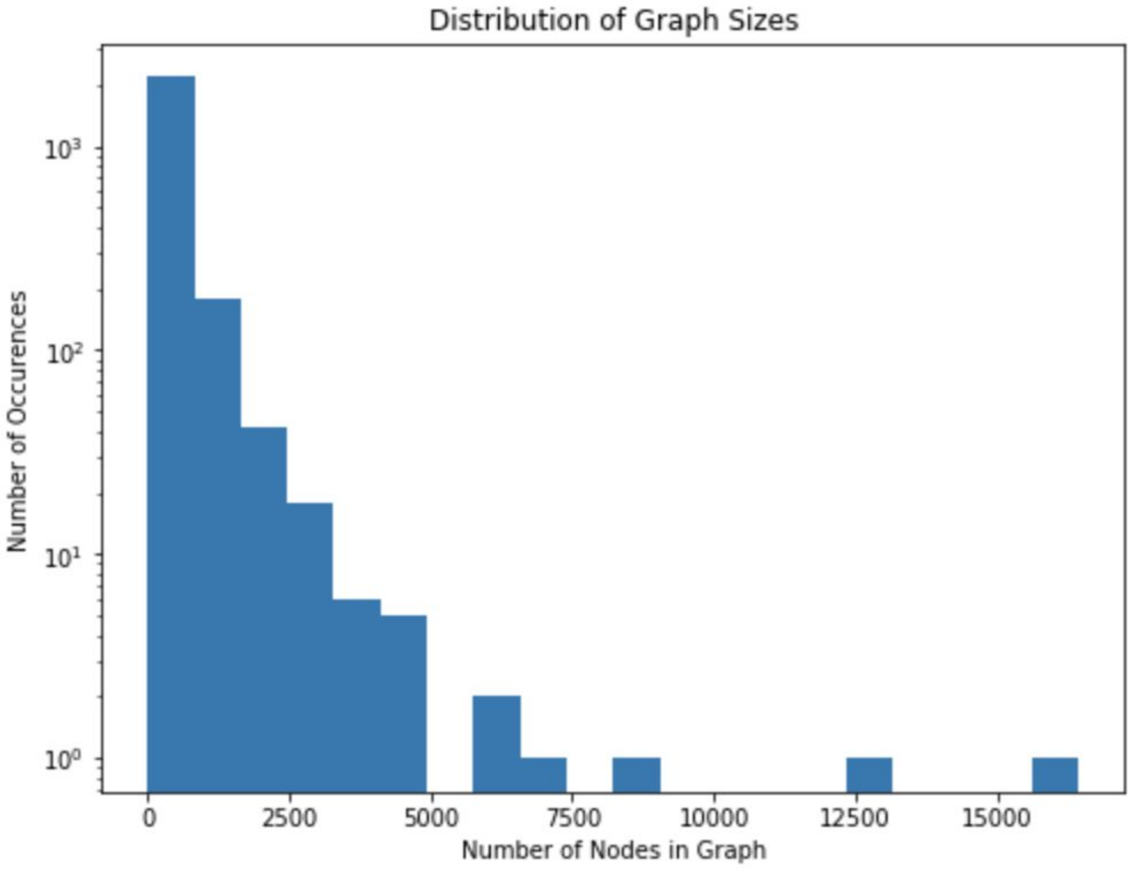
	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	1	0	1	0
C	0	0	0	1	1	0
D	0	0	0	0	0	0
E	0	0	0	1	0	1
F	0	0	0	0	0	0

(a) Adjacency matrix for directed graph

...which, really, are just (binary) images!



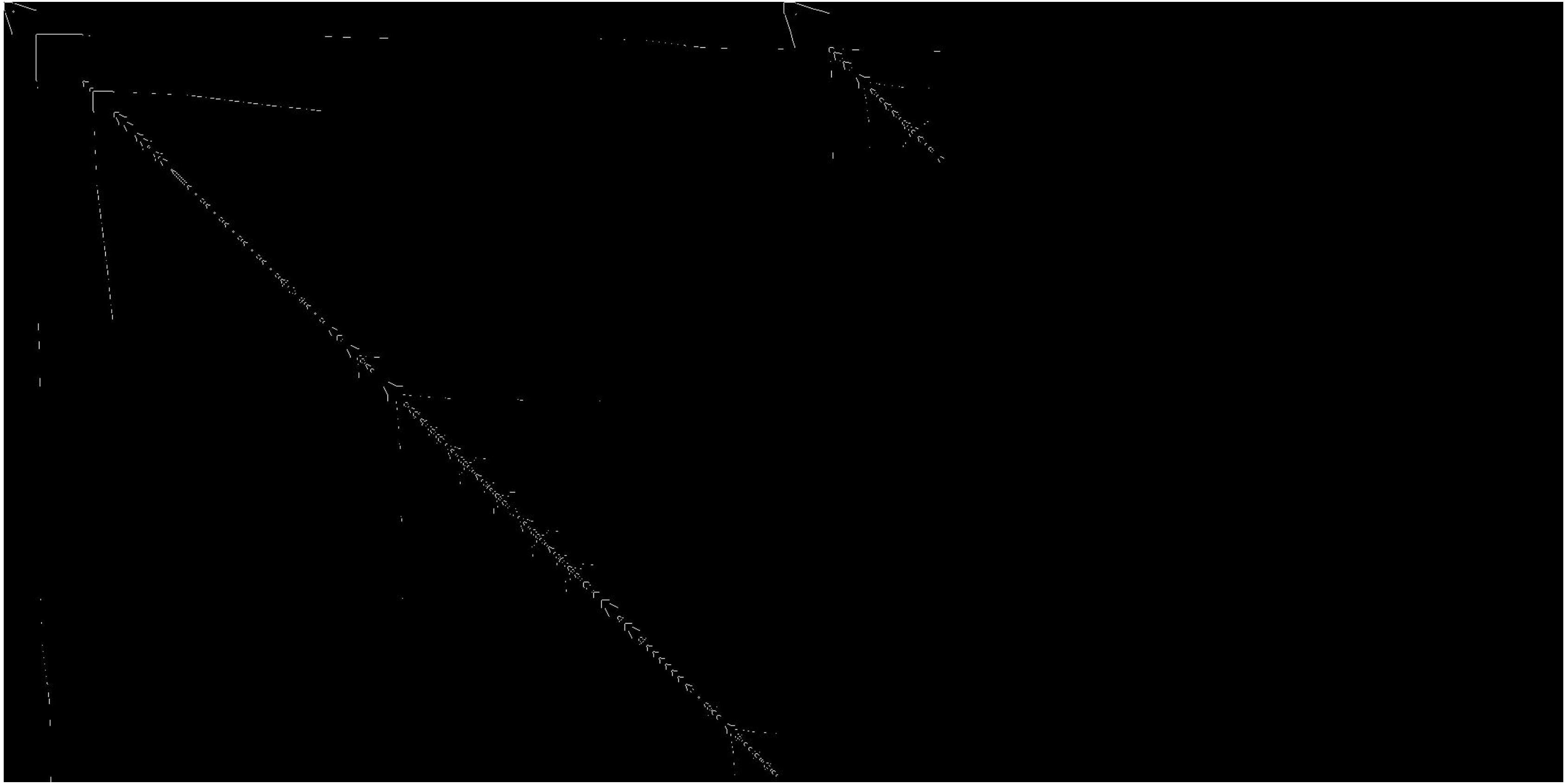
Adjacency Matrix Sizes



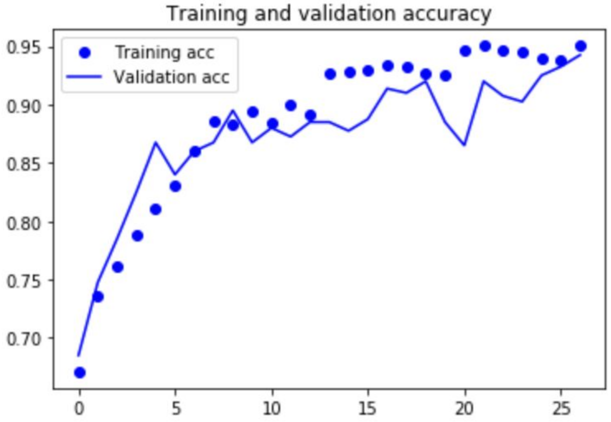
General Workflow

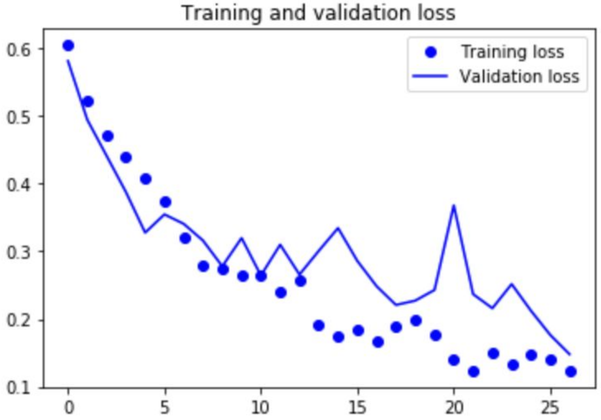
- Identify two files or functions with potential clone from metadata files
- Convert adjacency JSON to images
- Combine adjacency images two a single “clone image”
- Model!
  - file : file
  - function : function
  - Keras convolutional neural network

Sample file:file Clone Image



file:file Model Results


Training and validation accuracy


Training and validation loss

Total numbers of files:

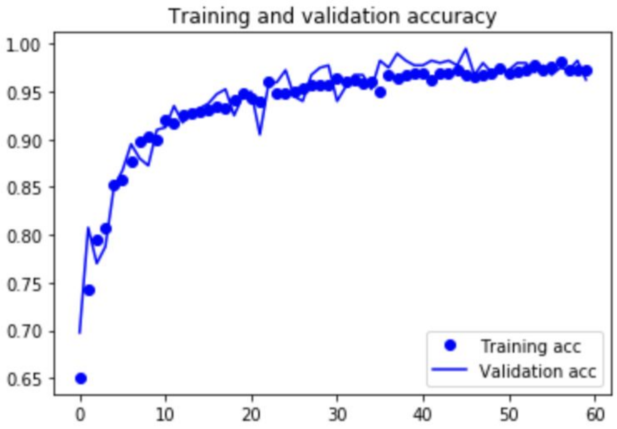
Total training neg: 5000
Total training type3: 8220
Total validation neg: 2500
Total validation type3: 4214
Total test neg: 2500
Total test type3: 4126

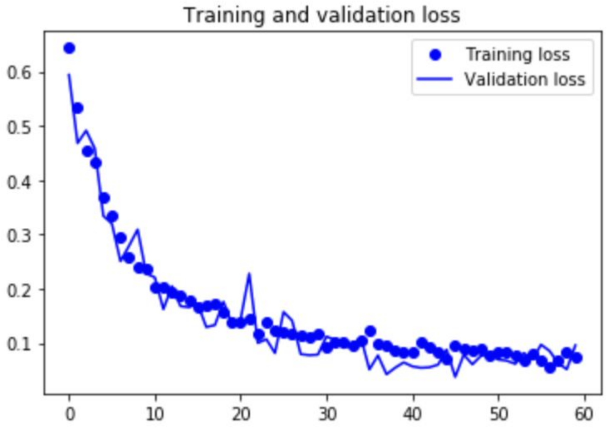
Test accuracy: 0.925
Test loss: 0.210

Sample function:function Clone Image

the best way to build and ship software

function:function Model Results



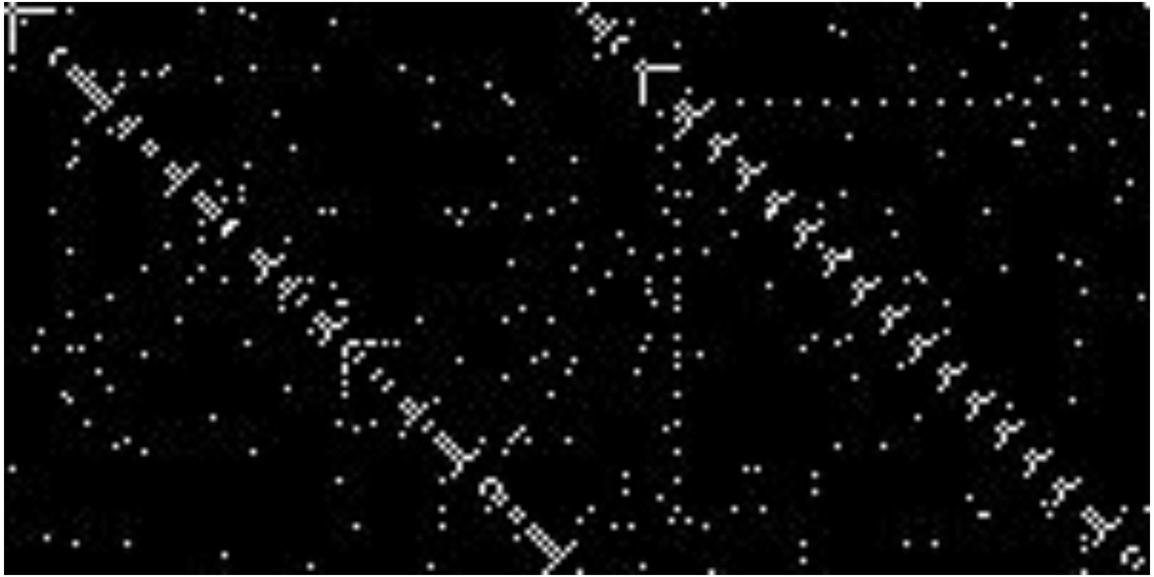


Total numbers of files:

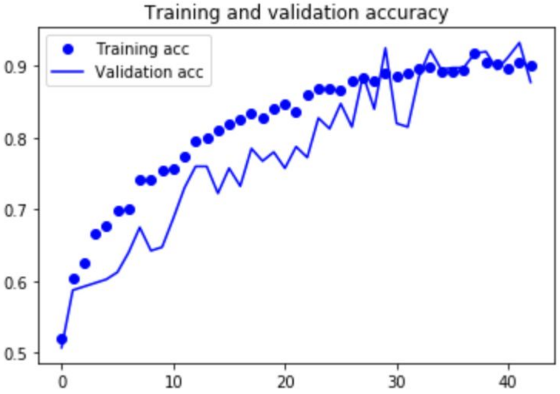
Total training neg: 27966
Total training type3: 24909
Total validation neg: 13982
Total validation type3: 12454
Total test neg: 13984
Total test type3: 12454

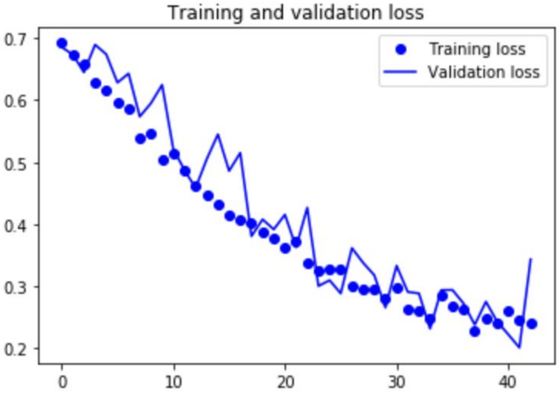
Test accuracy: 0.977
Test loss: 0.066

Adding Noise (1%)



function:function Noisy Model Results





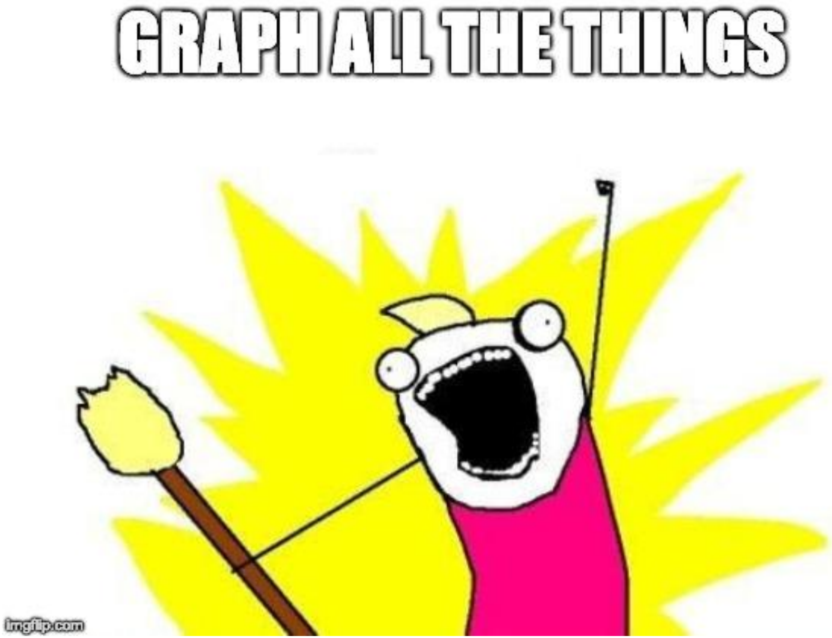
Total numbers of files:

Total training neg: 27966
Total training type3: 24909
Total validation neg: 13982
Total validation type3: 12454
Total test neg: 13984
Total test type3: 12454

Test accuracy: 0.888
Test loss: 0.349

Future Work

- Clone detection
  - Current approach is O(n^2)
    - How to get out of this?
  - Expand out to Type 4
    - With what training set?
- Graphs in general
  - All of GitHub data can be represented as a graph!



Thank You!

cj2001@github.com

cj2001@gmail.com

@cjisalock