CS846
Machine Learning for Software Engineering

Pengyu Nie

Introduction to Class

How this course works

Round-table introduction

Overview of ML4SE research

Communication Tools

- Webpage: https://pengyunie.github.io/cs846mlse-1249/
  - Course information, schedule, reading list
- Teams chat
  - Mostly for paper discussion
- Learn: https://learn.uwaterloo.ca/d2l/home/1046525
  - (rare) Announcements
- Email pynie@uwaterloo.ca
  - Project report submission, questions/concerns
  - Prefix your email with [CS846]

Agenda

Date	Topic	Reading
Sep 09	Introduction to class; overview of ML4SE research	None
Sep 16	Language modelling and n-gram models	
Sep 23	Sequence-to-sequence models and transformers	
Sep 30	Large language models for code	
Oct 07	Software datasets (GitHub, StackOverflow, etc.)	
Oct 14	Reading week - no class	None
Oct 21	Build systems essentials and parsing	
Oct 28	Static analysis	
Nov 04	Dynamic analysis	
Nov 11	Task: program comprehension	
Nov 18	Task: code completion and generation	
Nov 25	Task: code translation	
Dec 02	Task: bug detection and fixing	

today

Schedule 1

Lecture and demo (40min)
Break (10min)
Paper discussion (40min)
Flexible time

Schedule 2

Paper discussion 1 (40min)
Break (10min)
Paper discussion 2 (40min)
Break (10min)
Project presentations (<=60min)

Paper Discussion

- Discussion leads (x2 per paper)
  - Choose a paper from the reading list of the day
  - Announce (via Teams chat) your choice by Wednesday of the prev week
  - Familiarize yourselves with the paper
    - If needed, ask the instructor / your friends / ChatGPT before it is too late
  - Lead the discussion in class
    - Summarize the paper (5-10min); no need to make slides
    - Positive leader: find reasons to accept the paper
    - Negative leader: find reasons to reject the paper
- Audience
  - Read the paper before class
  - Participate in the discussion, ask questions, answer questions, etc.

Paper Discussion: Assessment

Task	Due Date	Weight
Attendance	-	20%
Paper discussion lead	-	20%

- Missing a few classes due to deadlines/conferences/etc is fine
- You will be deducted points if and only if, by the end of the term, I determine, at my own discretion:
  - You failed to show up in most of the classes for no good reason
  - You showed up, but seldomly participate in the paper discussions

- Be the discussion leader at least once :)
- Judged by your performance as a discussion leader (but normally, I don't deduct points unless it was really bad)

Paper Discussion: Reading List and Signup Sheet

- Reading list on course webpage
  - Obtain my approval if you want to choose a paper not on the list
- Signup sheet
  - https://tinyurl.com/2yxshejy
  - First come first serve
  - Only swap slots with others if you reach an agreement
- Benefits of signing up for earlier slots:
  - The papers tend to be easier to read (in fact, I haven't decided about the later papers)
  - Be done with the paper discussion lead duty and focus on other things

Reading List

Sep 16: language modeling and n-gram models

- Big code != big vocabulary: open-vocabulary models for source code
- Capturing Structural Locality in Non-parametric Language Models
- On the Localness of Software
- Mining Source Code Repositories at Massive Scale Using Language Modeling

Sep 23: sequence-to-sequence models and transformers

- Empirical study of transformers for source code
- Retrieval Augmented Code Generation and Summarization
- Long-Range Modeling of Source Code Files with eWASH: Extended Window Access by Syntax Hierarchy
- Synchromesh: Reliable code generation from pre-trained language models

Sep 30: large language models for code

- Show Your Work: Scratchpads for Intermediate Computation with Language Models
- A Static Evaluation of Code Completion by Large Language Models
- Do Large Language Models Pay Similar Attention Like Human Programmers When Generating Code?
- Traces of Memorisation in Large Language Models for Code

Project

- Timeline →
- Team size: 2-4

Project: team formation	Sep 25 (Wed)	-
Project: proposal	Oct 11 (Fri)	10%
Project: progress report	Nov 01 (Fri)	20%
Project: final report	Dec 05 (Thu)	30%

- Expectation: a short-paper-level (or above) project
  - Tool paper: research prototype -> implementation & integration
  - Dataset paper: task -> dataset/benchmark
  - Mining challenge: dataset -> statistics & observations
  - Replication study: prior work -> new dataset/model
- Find your teammates by Sep 25 (after Add/Drop ddl)
  - Changes to the list of team members not allowed after this date
  - Start discussing what you want to do for the project

Project: Deliverables

Project: team formation	Sep 25 (Wed)	-
Project: proposal	Oct 11 (Fri)	10%
Project: progress report	Nov 01 (Fri)	20%
Project: final report	Dec 05 (Thu)	30%

- Compulsory reports
  - Proposal report: 1 page
  - Progress report: 2-4 pages
  - Final report: 4-10 pages
  - Use ACM format by default
    - Feel free to use a different template if you're targeting a specific conference
- Optional presentations
  - Present your proposal/progress in class (Schedule 1)
    - ~60min flexible time ≈ 4 teams x 15min
    - If you're interested, let the instructor know before class; first come first serve
  - Final presentations (Schedule 2)
    - Invited by the instructor for teams making good progress

Round-Table Introduction

- Name
- Masters/PhD
- Research interest
- Expectations from this course
- Fun fact

Overview of ML4SE Research

- Context of this course: a brief history from 2012 to 2021
- Topics covered in this course

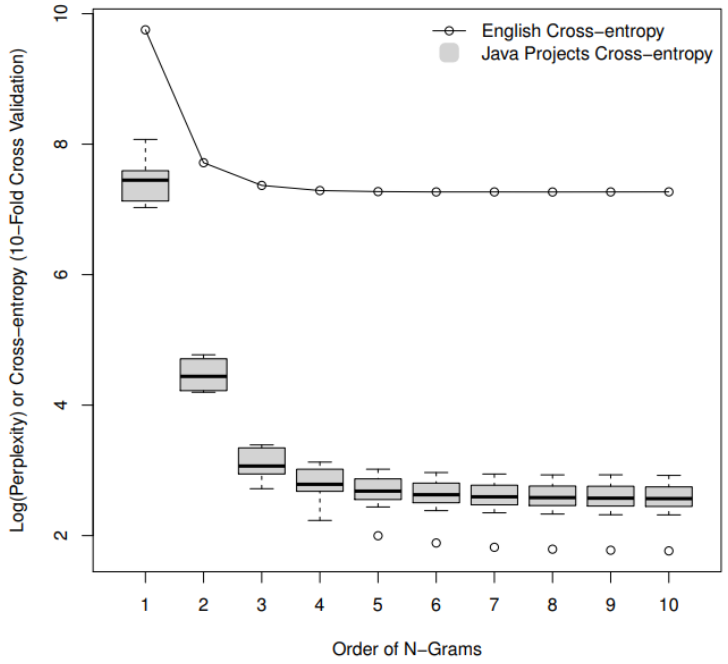
2012: Naturalness of Software

- Software code written in PLs are natural
  - Repetitive, predictable
- Experiments: more "natural" than English

> *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.*

n-gram language models

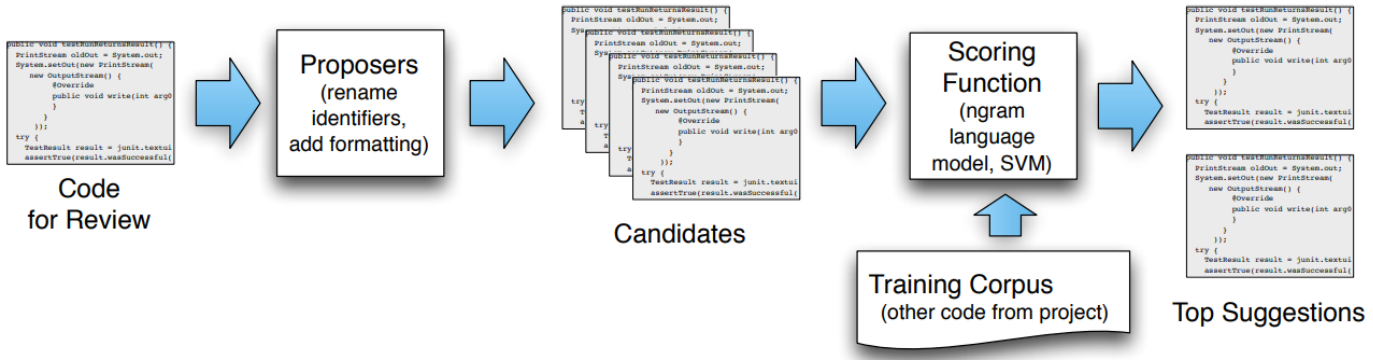
Java Project	Version	Lines	Tokens	
			Total	Unique
Ant	20110123	254457	919148	27008
Batik	20110118	367293	1384554	30298
Cassandra	20110122	135992	697498	13002
Eclipse-E4	20110426	1543206	6807301	98652
Log4J	20101119	68528	247001	8056
Lucene	20100319	429957	2130349	32676
Maven2	20101118	61622	263831	7637
Maven3	20110122	114527	462397	10839
Xalan-J	20091212	349837	1085022	39383
Xerces	20110111	257572	992623	19542

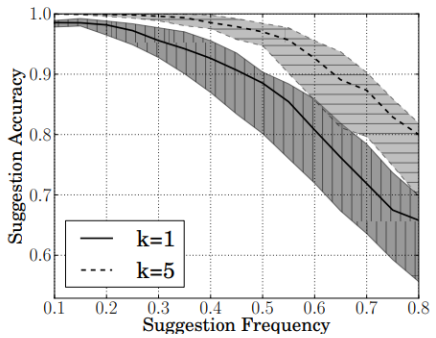


Hindle et al. On the naturalness of software. In ICSE 2012. https://doi.org/10.1109/ICSE.2012.6227135

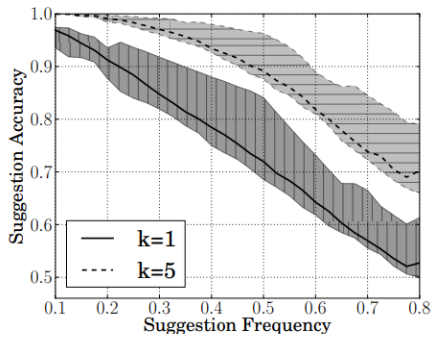
2014: Naturalize

- Predicting coding conventions
  - Identifier naming
  - Formatting (spaces, newlines)

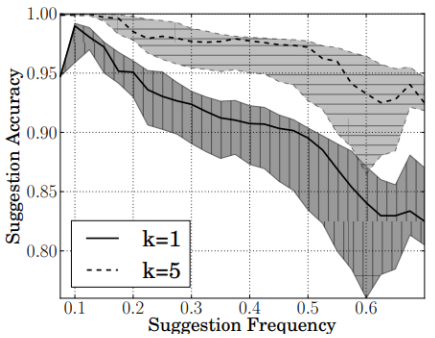




(a) Variables



(b) Method Calls



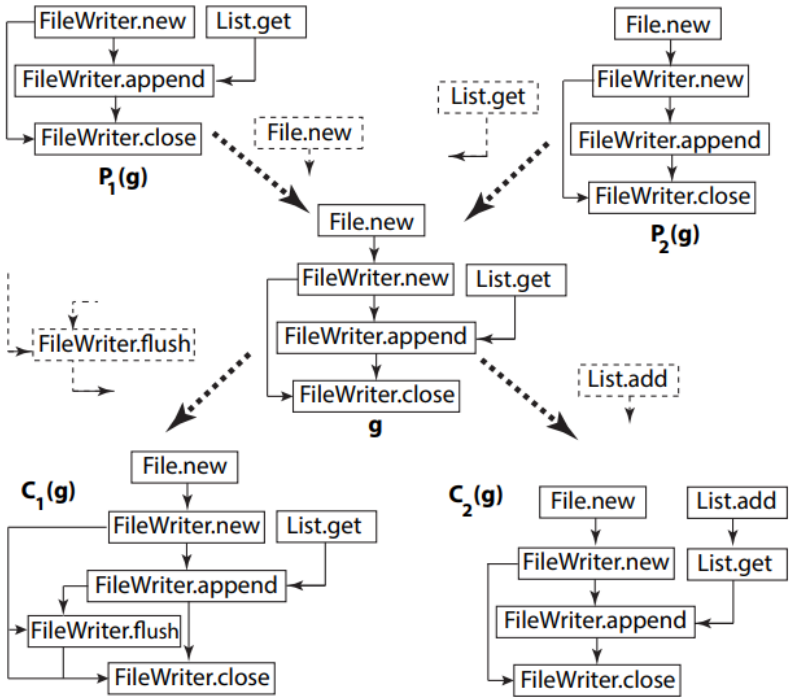
(c) Typenames

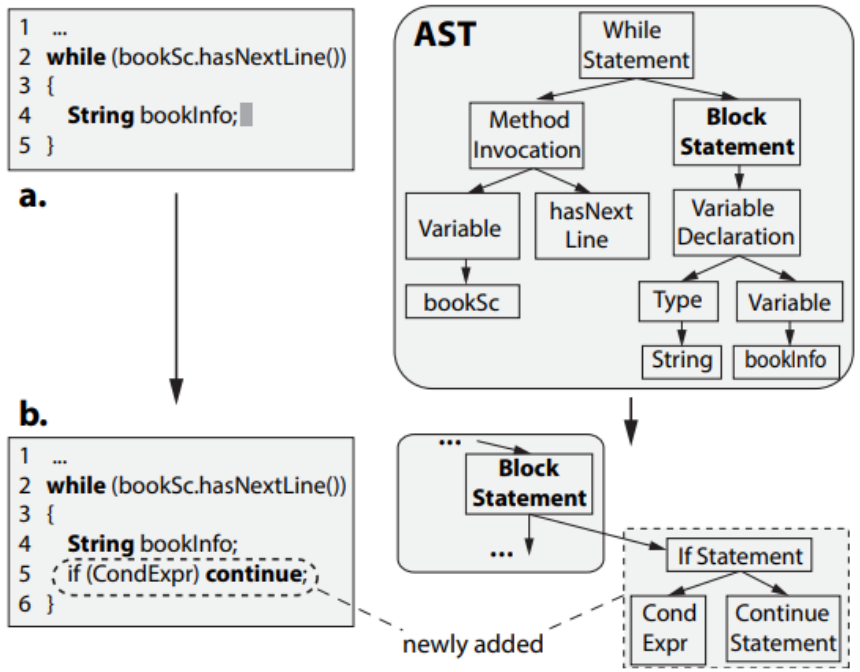
Allamanis et al. Learning natural coding conventions. In FSE 2014. https://doi.org/10.1145/2635868.2635883

2015: Graph-based Statistical Language Model

- Capture tree & graph structures in code

Still n-gram, but the graph version of it





Nguyen and Nguyen. Graph-based Statistical Language Model for Code. In ICSE 2015. https://doi.org/10.1109/ICSE.2015.336

2016: RNN, CNN

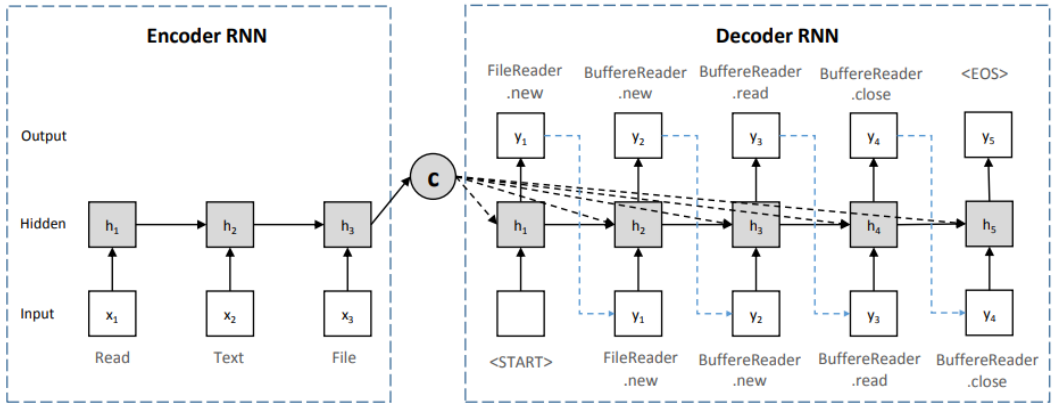


Figure 2: An Illustration of the RNN Encoder-Decoder Model for API learning

Gu et al. Deep API learning. In FSE 2016. https://doi.org/10.1145/2950290.2950334

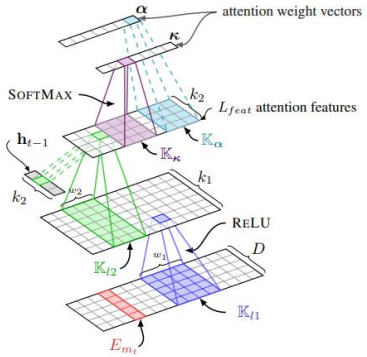
+ attention mechanism
+ copy mechanism
…

```
boolean shouldRender()

try {
  return renderRequested||isContinuous;
} finally {
  renderRequested = false;
}
```

Suggestions:

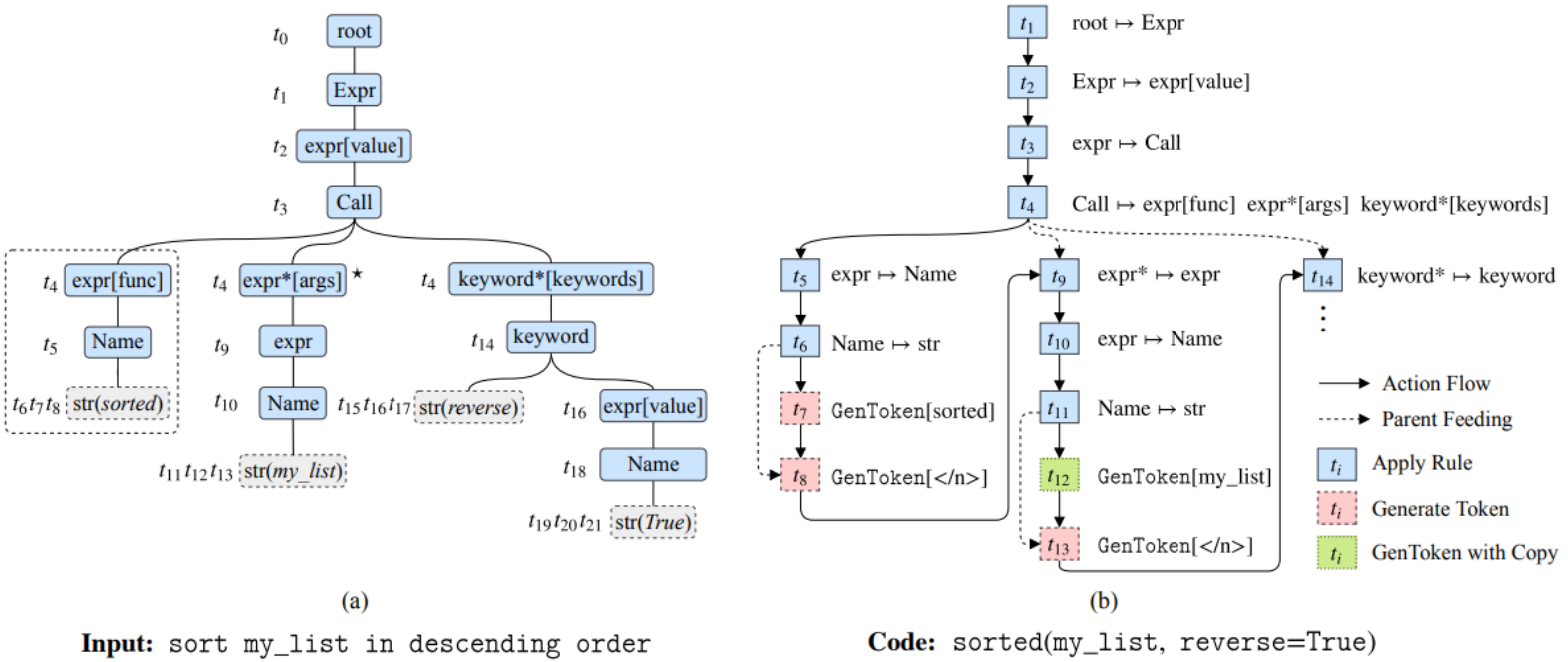
- is,render (27.3%)
- is,continuous (10.6%)
- is,requested (8.2%)
- render,continuous (6.9%)
- get,render (5.7%)



Allamanis et al. A Convolutional Attention Network for Extreme Summarization of Source Code. In ICML 2016. https://proceedings.mlr.press/v48/allamanis16.pdf

2017: Grammar-based Models

- Constrains the output to be syntactical valid programs
- Code <-> Text becomes a very popular task



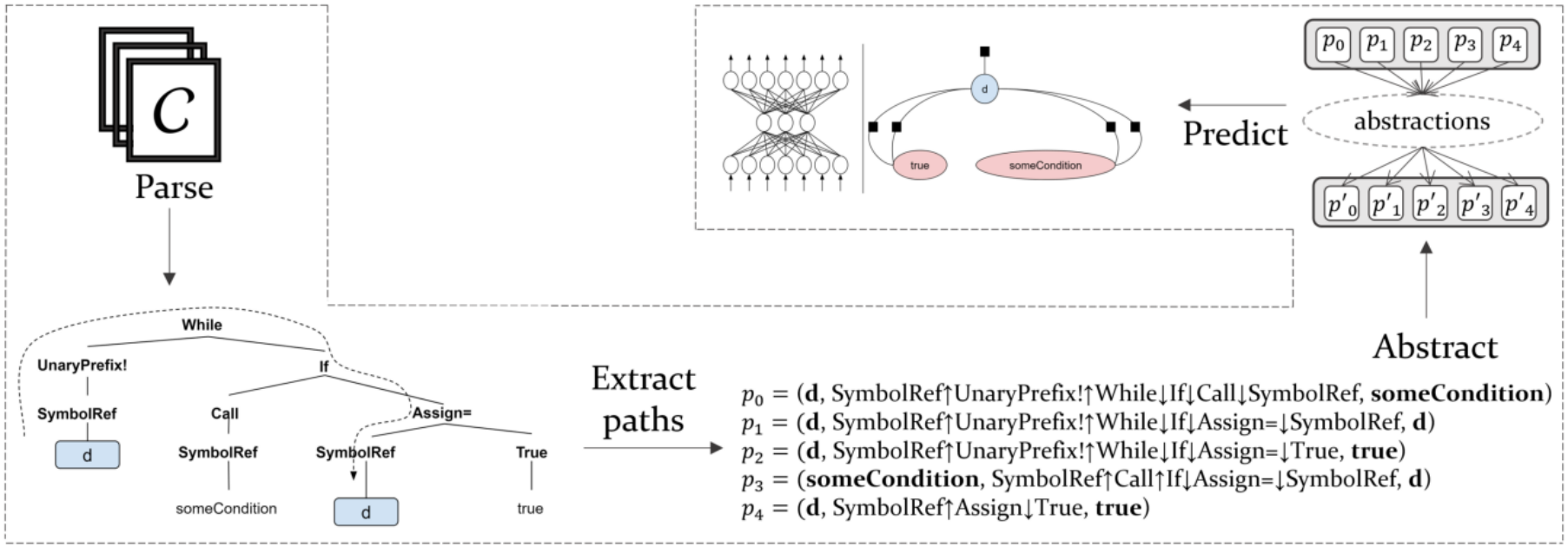
Input: `sort my_list in descending order`

Code: `sorted(my_list, reverse=True)`

Yin and Neubig. A Syntactic Neural Model for General-Purpose Code Generation. In ACL 2017. https://arxiv.org/abs/1704.01696

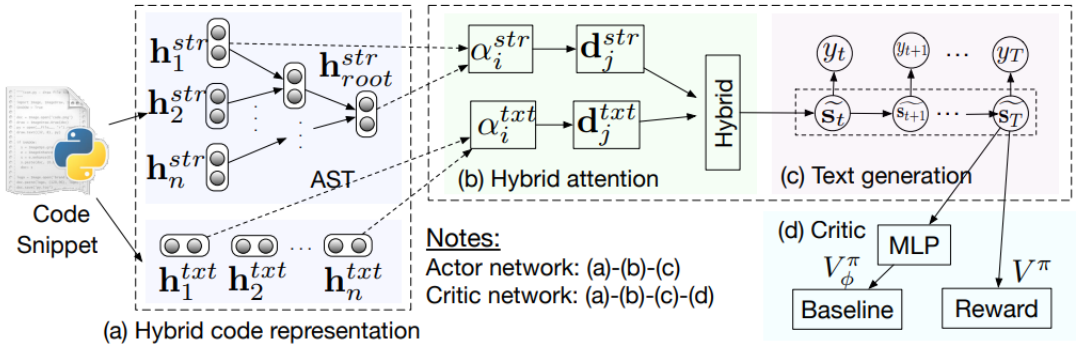
2018: Path-based Representation

- Encode program using AST paths



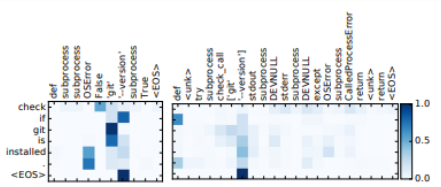
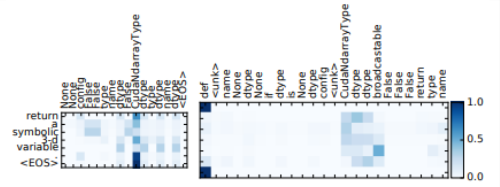
Alon et al. A general path-based representation for predicting program properties. In PLDI 2018. https://doi.org/10.1145/3192366.3192412

2018: Reinforcement Learning



	Case 1	Case 2

Code snippet	`def _has_git():`
	`  try: subprocess.check_call(`
	`  [git, -version],`
	`  stdout=subprocess.DEVNULL,`
	`  stderr=subprocess.DEVNULL)`
	`  except(OSError, subprocess`
	`  .CalledProcessError):`
	`  return False`
	`  else: return True`
		`def tensor3(name=None,dtype=None):`
		`  if (dtype is None):`
		`  dtype=config.floatX`
		`  type=CudaNdarrayType(`
		`  dtype=dtype,`
		`  broadcastable=`
		`  (False, False, False))`
		`  return type(name)`

Ground truth	check if git is installed .	return a symbolic 3-d variable .
Seq2Seq	helper function to create a new figure manager instance .	yaml
Seq2Seq+Attn	return true if the user has access to the specified resource .	a decorator that returns a new class that will return a new class name .
Tree2Seq+Attn	test that validate_folders throws a foldermissingerror .	helper function for #4957 .
Hybrid2Seq+Attn	returns the number of git modules that are not installed .	return the path to the currently running server .
Hybrid2Seq+Attn+DRL	returns true if git is installed .	return a symbolic graph .
Attention visualization		

Wan et al. Improving automatic source code summarization via deep reinforcement learning. In ASE 2018. https://doi.org/10.1145/3238147.3238206

2019: CodeSearchNet

- Race of collecting massive, high-quality dataset/benchmarks

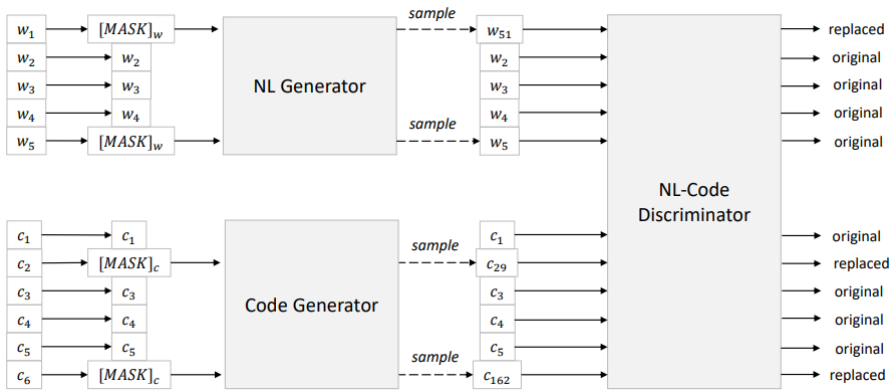
	Number of Functions	
	w/ documentation	All
Go	347 789	726 768
Java	542 991	1 569 889
JavaScript	157 988	1 857 835
PHP	717 313	977 821
Python	503 502	1 156 085
Ruby	57 393	164 048
All	2 326 976	6 452 446

Encoder		CodeSearchNet Corpus (MRR)						
Text	Code	Go	Java	JS	PHP	Python	Ruby	Avg
NBoW	NBoW	0.6409	0.5140	0.4607	0.4835	0.5809	0.4285	0.6167
1D-CNN	1D-CNN	0.6274	0.5270	0.3523	0.5294	0.5708	0.2450	0.6206
biRNN	biRNN	0.4524	0.2865	0.1530	0.2512	0.3213	0.0835	0.4262
SelfAtt	SelfAtt	**0.6809**	**0.5866**	0.4506	**0.6011**	**0.6922**	0.3651	**0.7011**
SelfAtt	NBoW	0.6631	0.5618	**0.4920**	0.5083	0.6113	**0.4574**	0.6505

Husain et al. CodeSearchNet challenge: evaluating the state of semantic code search. https://arxiv.org/abs/1909.09436

2020: CodeBERT

- Transformers & large-scale pre-training
  - w/ pre-training objectives specific to SE



	RUBY	JAVASCRIPT	GO	PYTHON	JAVA	PHP	ALL
NUMBER OF DATAPOINTS FOR PROBING							
PL (2 CHOICES)	38	272	152	1,264	482	407	2,615
NL (4 CHOICES)	20	65	159	216	323	73	856
PL PROBING							
ROBERTA	73.68	63.97	72.37	59.18	59.96	69.78	62.45
PRE-TRAIN W/ CODE ONLY	71.05	77.94	89.47	70.41	70.12	82.31	74.11
CODEBERT (MLM)	**86.84**	**86.40**	**90.79**	**82.20**	**90.46**	**88.21**	**85.66**
PL PROBING WITH PRECEDING CONTEXT ONLY							
ROBERTA	**73.68**	**53.31**	51.32	55.14	42.32	52.58	52.24
PRE-TRAIN W/ CODE ONLY	63.16	48.53	**61.84**	56.25	**58.51**	58.97	56.71
CODEBERT (MLM)	65.79	50.74	59.21	**62.03**	54.98	**59.95**	**59.12**
NL PROBING							
ROBERTA	50.00	72.31	54.72	61.57	61.61	65.75	61.21
PRE-TRAIN W/ CODE ONLY	55.00	67.69	60.38	68.06	65.02	68.49	65.19
CODEBERT (MLM)	**65.00**	**89.23**	**66.67**	**76.85**	**73.37**	**79.45**	**74.53**

Feng et al. CodeBERT: a pre-trained model for programming and natural languages. In Findings of EMNLP 2020. https://arxiv.org/abs/2002.08155

2021: Codex (-> GitHub Copilot)

- GPT-3 fine-tuned on coding dataset
- Introduced the "HumanEval" code generation benchmark

```python
def incr_list(l: list):
    """Return list with elements incremented by 1.
    >>> incr_list([1, 2, 3])
    [2, 3, 4]
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])
    [6, 4, 6, 3, 4, 4, 10, 1, 124]
    """
    return [i + 1 for i in l]
```

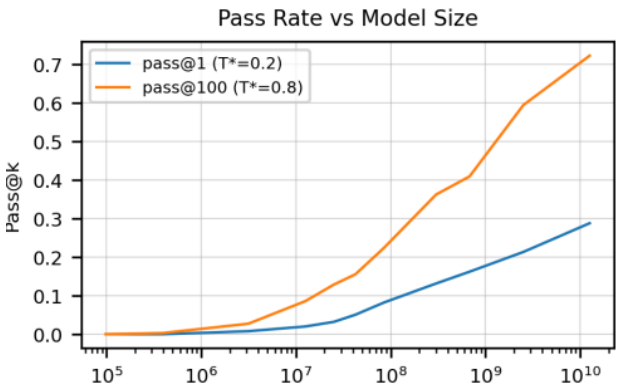
```python
def solution(lst):
    """Given a non-empty list of integers, return the sum of all of the odd elements
    that are in even positions.

    Examples
    solution([5, 8, 7, 1]) ==>12
    solution([3, 3, 3, 3, 3]) ==>9
    solution([30, 13, 24, 321]) ==>0
    """
    return sum(lst[i] for i in range(0,len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```

```python
def encode_cyclic(s: str):
    """
    returns encoded string by cycling groups of three characters.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group. Unless group has fewer elements than 3.
    groups = [(group[1:] + group[0]) if len(group) == 3 else group for group in groups]
    return "".join(groups)

def decode_cyclic(s: str):
    """
    takes as input string encoded with encode_cyclic function. Returns decoded string.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group.
    groups = [(group[-1] + group[:-1]) if len(group) == 3 else group for group in groups]
    return "".join(groups)
```

	PASS@k		
	$k = 1$	$k = 10$	$k = 100$
GPT-NEO 125M	0.75%	1.88%	2.97%
GPT-NEO 1.3B	4.79%	7.47%	16.30%
GPT-NEO 2.7B	6.41%	11.27%	21.37%
GPT-J 6B	11.62%	15.74%	27.74%
TABNINE	2.58%	4.35%	7.59%
CODEX-12M	2.00%	3.62%	8.58%
CODEX-25M	3.21%	7.1%	12.89%
CODEX-42M	5.06%	8.8%	15.55%
CODEX-85M	8.22%	12.81%	22.4%
CODEX-300M	13.17%	20.37%	36.27%
CODEX-679M	16.22%	25.7%	40.95%
CODEX-2.5B	21.36%	35.42%	59.5%
CODEX-12B	28.81%	46.81%	72.31%



Chen et al. Evaluating Large Language Models Trained on Code. https://arxiv.org/abs/2107.03374

Topics

Date	Topic	Reading
Sep 09	Introduction to class; overview of ML4SE research	None
Sep 16	Language modelling and n-gram models	
Sep 23	Sequence-to-sequence models and transformers	
Sep 30	Large language models for code	
Oct 07	Software datasets (GitHub, StackOverflow, etc.)	
Oct 14	Reading week - no class	None
Oct 21	Build systems essentials and parsing	
Oct 28	Static analysis	
Nov 04	Dynamic analysis	
Nov 11	Task: program comprehension	
Nov 18	Task: code completion and generation	
Nov 25	Task: code translation	
Dec 02	Task: bug detection and fixing	

Background on Machine Learning
What models to use?

Background on Software Engineering
What data to use and how to collect?

What tasks to solve? (subject to change)