每周汇报

蔡碧瑜

2022/04/14

论文列表

No.	title	Publication Source	Year
1	Boosting Coverage-Based Fault Localization via Graph-Based Representation Learning	FSE/ESEC	2021
2	A First Look at Developers’ Live Chat on Gitter	FSE/ESEC	2021
3	Empirical Study of Transformers for Source Code	FSE/ESEC	2021

/01

Boosting Coverage-Based Fault Localization via Graph-Based Representation Learning (2021.FSE/ESEC)

Boosting Coverage-Based Fault Localization via Graph-Based Representation Learning (2021.FSE/ESEC)

SBFL (Spectrum Fault Localization)

Program entities covered by more failed tests and less passed tests are more likely to be faulty.

ID	Method signature	Coverage								SBFL	Rank
		ft_1	ft_2	pt_1	pt_2	pt_3	pt_4	pt_5	pt_6		
m_1	`public String getNullText()`	✓	✓	✓	✓	✓				0.63	1
m_2	`public StrBuilder appendFixedWidthPadLeft(Object, int, char)`	✓		✓			✓			0.41	2
m_3	`public StrBuilder appendFixedWidthPadRight(Object, int, char)`		✓		✓			✓	✓	0.35	3
m_4	`public StrBuilder()`	✓	✓	✓	✓	✓	✓	✓		0.12	5
m_5	`public StrBuilder(int)`	✓	✓	✓	✓	✓	✓	✓	✓	0.12	5
m_6	`public StrBuilder ensureCapacity(int)`	✓	✓	✓	✓		✓	✓	✓	0.10	6

```
public String getNullText(){
  return nullText;}
```

```
public StrBuilder appendFixedWidthPadLeft(Object, int, char) {
  if (width > 0) {
     ensureCapacity(size + width);
     String str = (obj == null ? getNullText() : obj.toString());
     int strLen = str.length(); // bug
     ...
```

Figure 1: Code snippets of m_1 and m_2

Boosting Coverage-Based Fault Localization via Graph-Based Representation Learning (2021.FSE/ESEC)

Nodes:

- Program entities
- Tests

Edges:

- Coverage relationships
- Code structures

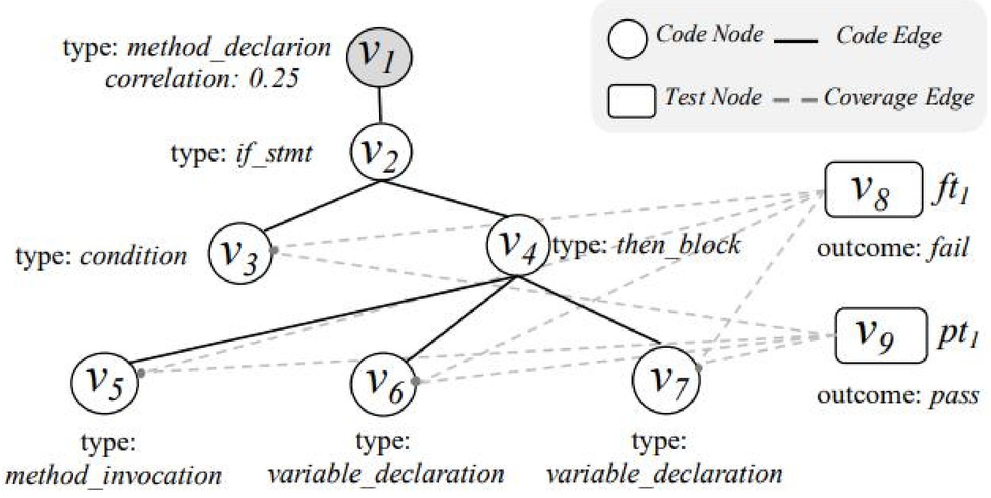


Figure 2: Representations for the method m_2

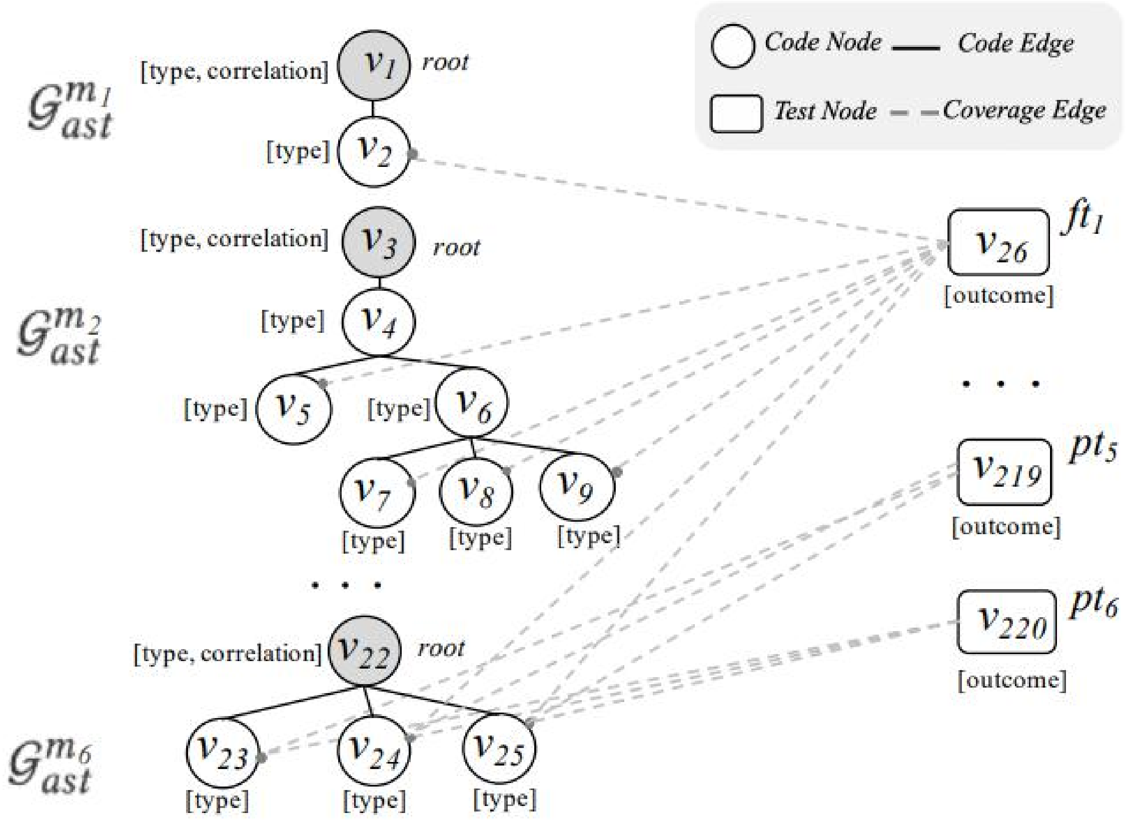


Figure 3: Unified coverage graph for Lang-47

$$\mathrm{cor}(w_m, w_t) = \mathrm{len}(w_m \cap w_t) / \mathrm{len}(w_t)$$

Boosting Coverage-Based Fault Localization via Graph-Based Representation Learning (2021.FSE/ESEC)

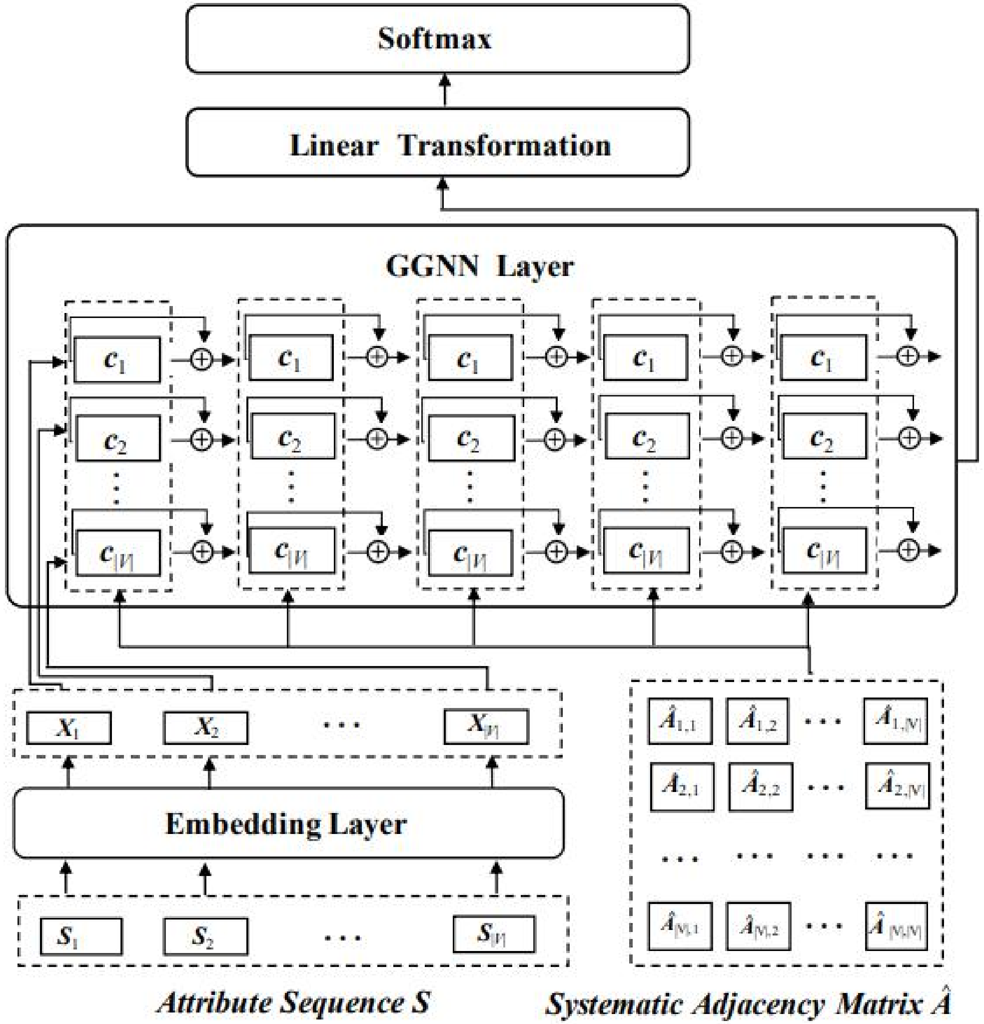


Figure 4: Architecture of GRACE

/02

A First Look at Developers' Live Chat on Gitter (2021. FSE/ESEC)

A First Look at Developers' Live Chat on Gitter (2021. FSE/ESEC)

RQ1(Communication Profile)

- Do Gitter communities demonstrate consistent community communication profiles?

RQ2(Community Structure)

- What are the structural characteristics of social networks built from developer live chat data?

RQ3(Dialog Topic)

- What are the primary topic types frequently discussed by developers in live chat?

RQ4(Interaction Pattern)

- How do developers typically interact with each other in live chat?

A First Look at Developers' Live Chat on Gitter (2021. FSE/ESEC)

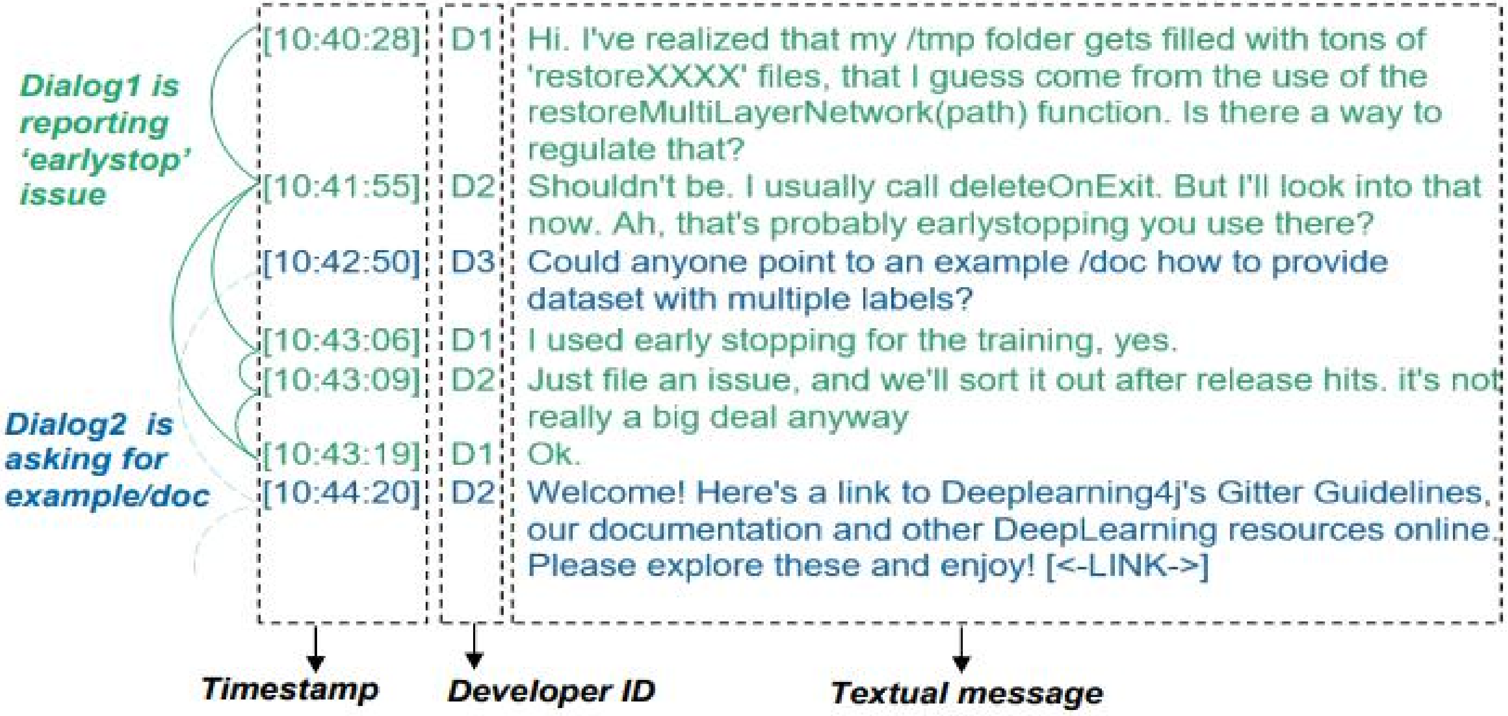


Figure 1: A slice of live chat log from the DL4J community.

A First Look at Developers' Live Chat on Gitter (2021. FSE/ESEC)

1. BiLSTM model
2. BERT model
3. E2E model
4. **FF model**

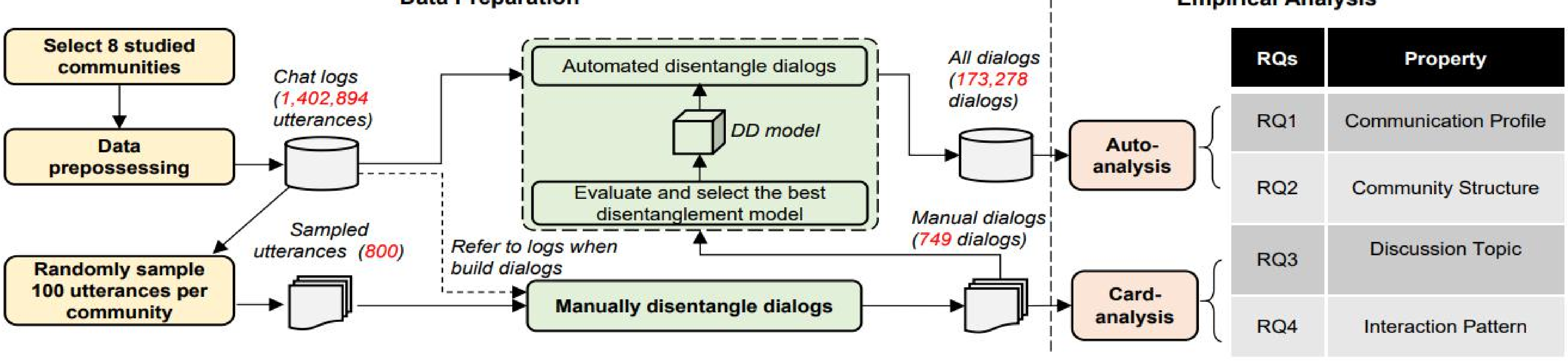
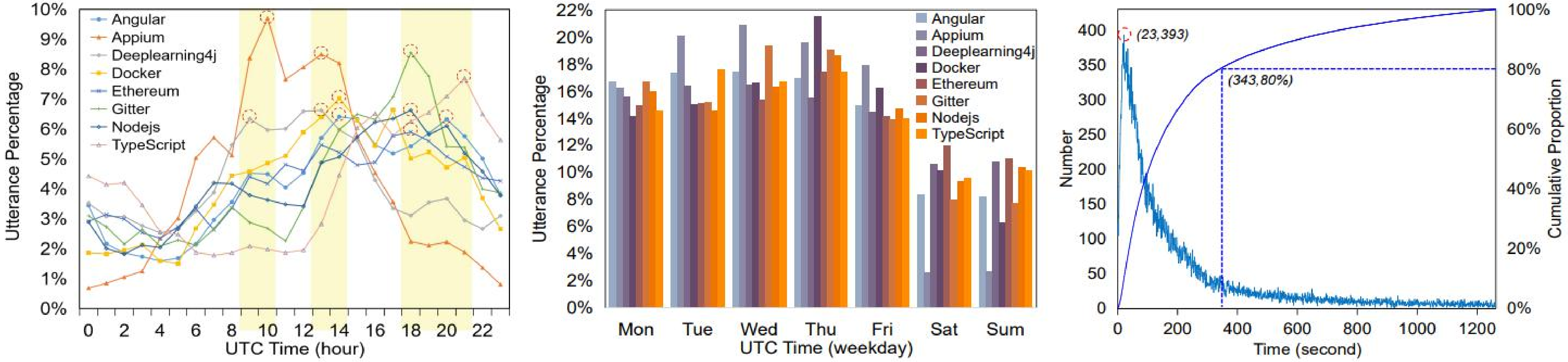


Figure 2: Overview of research methodology

A First Look at Developers’ Live Chat on Gitter (2021. FSE/ESEC)

RQ1
- When the developers are active?
- How long the respondent replies to the dialog initiator?

$$Time\_lag = T_r - T_i$$

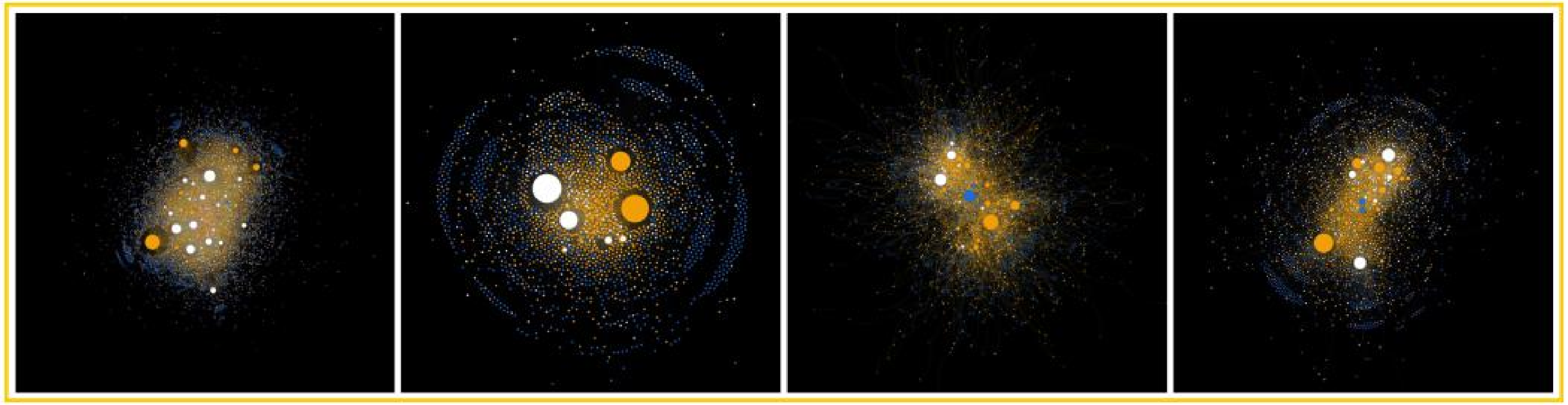


(a) Hourly distribution (b) Day of week distribution (c) Frequency distribution of response times

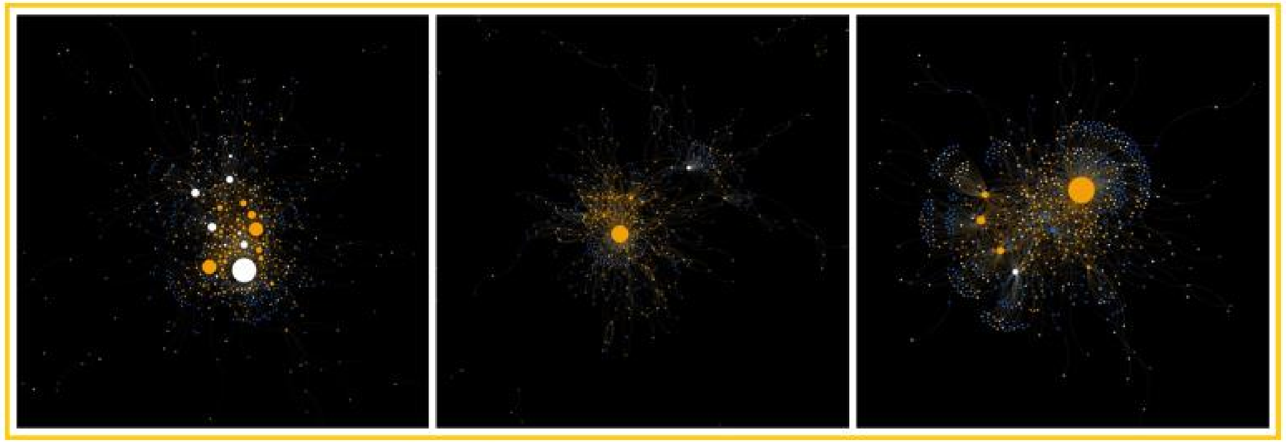
Figure 3: Statistic results about communication profiles

A First Look at Developers' Live Chat on Gitter (2021. FSE/ESEC)

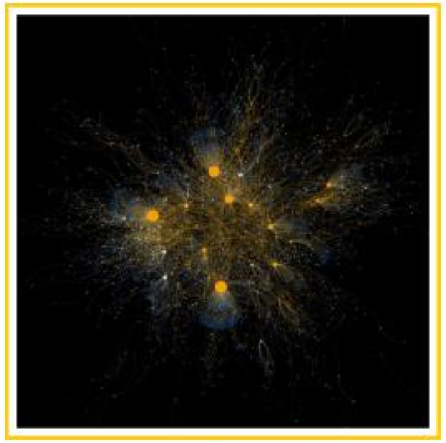
RQ2 Initiator：Blue， Respondent：White， Both：Orange



(a) Constellation {from left to right: Angular, DL4J, Nodejs, Typescript }



(b) Polaris {from left to right: Appium, Docker, Gitter }



(c) Galaxy { Ethereum }

A First Look at Developers' Live Chat on Gitter (2021. FSE/ESEC)

RQ3

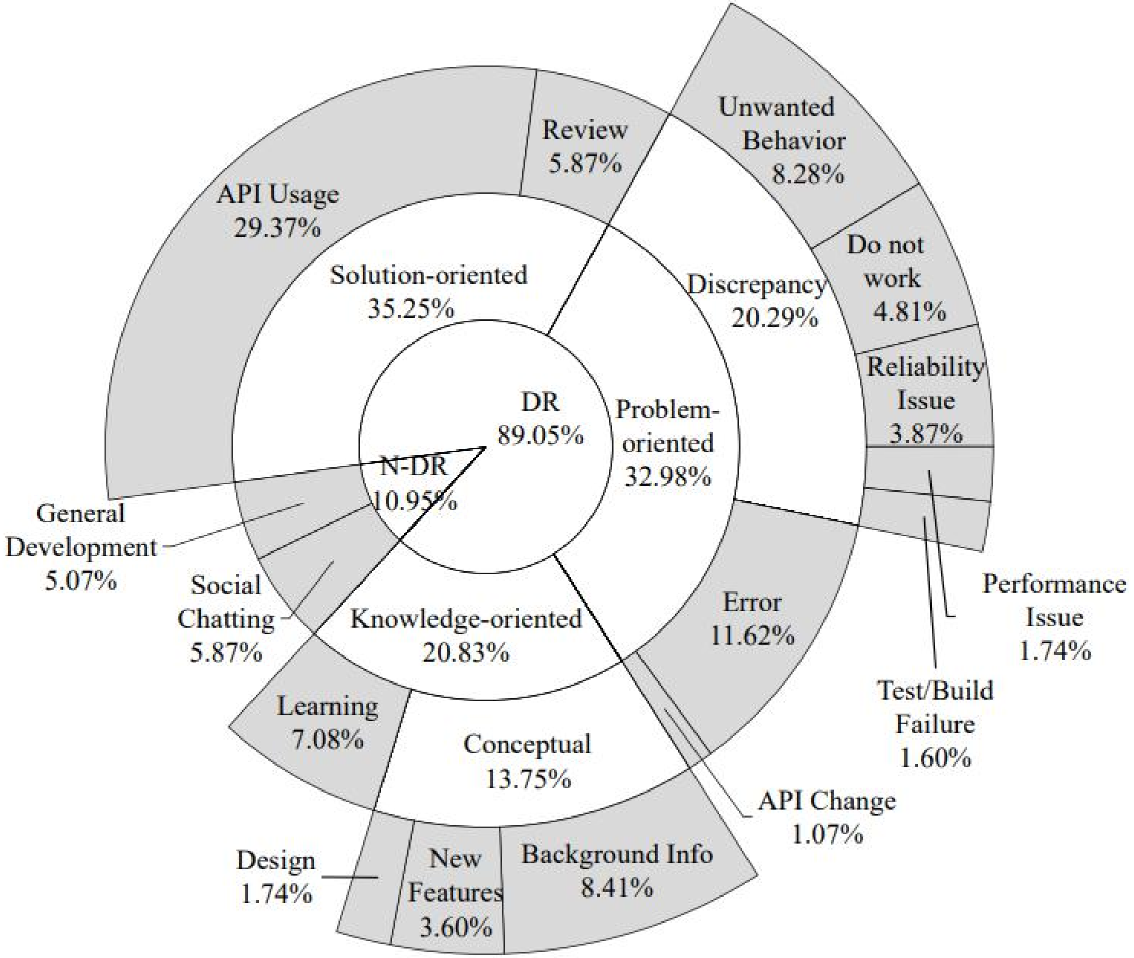


Figure 5: Distribution of discussion topics in developer live chat by reading from center to outside

A First Look at Developers' Live Chat on Gitter (2021. FSE/ESEC)

RQ4

Table 2: Developer intent category in live chat

Code	Label	Description
OQ	Original Question	The first question from the developer to initiate the dialog
CQ	Clarifying Question	Developers ask for clarifications
FD	Further Details	Developers provide more details
FQ	Follow Up Question	Developers ask for follow up questions about relevant issues
PA	Potential Answer	A potential answer or solution provided by developers
PF	Positive Feedback	Developer provides positive feedback for working solutions
NF	Negative Feedback	Developer provides negative feedback for useless solutions
GG	Greetings/Gratitude	Greetings or expressing gratitude

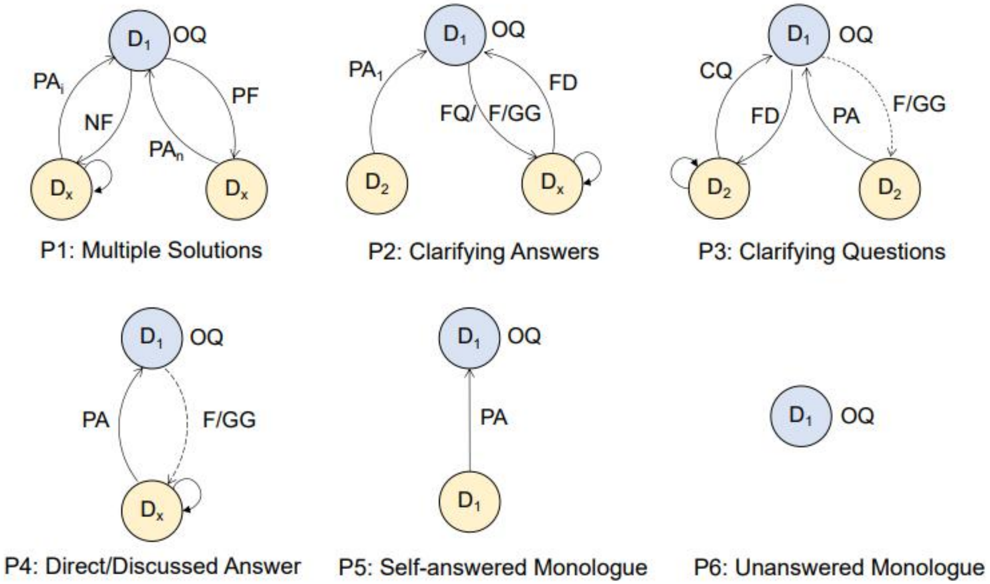


Figure 6: Interactive patterns, F denotes feedback including negative feedback and positive feedback, dashed lines denote optional interaction.

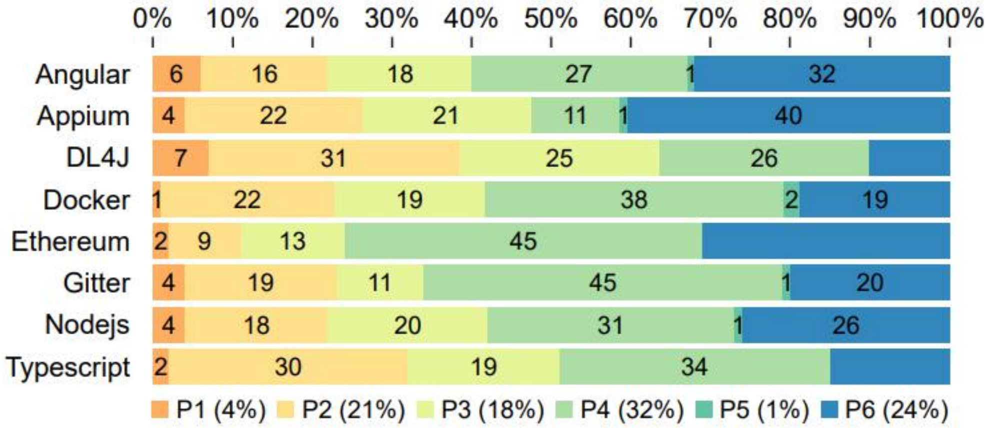


Figure 7: Distribution of interaction patterns among different communities

/03

Empirical Study of Transformers for Source Code (FSE/ESEC)

Empirical Study of Transformers for Source Code (FSE/ESEC)

Sequential positional encodings and embeddings

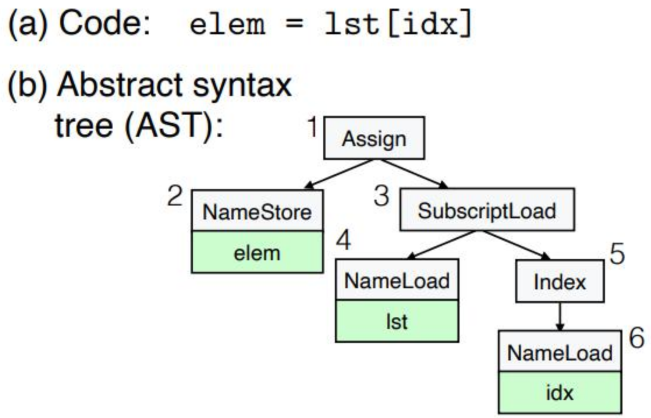
$$p_i \in R^{d_{model}} : \hat{x}_i = x_i + p_i$$

Sequential relative attention

$$z_i = \sum_j \tilde{\alpha}_{ij}(x_j^v + e_{i-j}^v), \;\; \tilde{\alpha}_{ij} = \frac{\exp(a_{ij})}{\sum_j \exp(a_{ij})}, \;\; a_{ij} = \frac{x_i^q (x_j^k + e_{i-j}^k)^T}{\sqrt{d_z}}$$

$e_{i-j}^v, e_{i-j}^k \in \mathbb{R}^{d_z}$ are learned embeddings for each relative position i-j

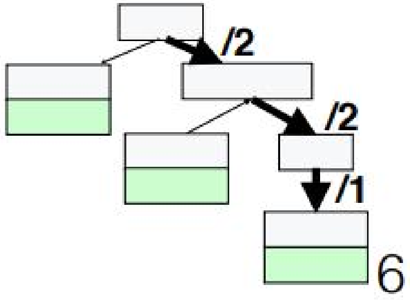
Empirical Study of Transformers for Source Code (FSE/ESEC)



Tree positional encodings

(d) Tree positional encodings:

1	2	3	4	5	6
/	/1	/2	/2/1	/2/2	/2/2/1



stack-like enc.

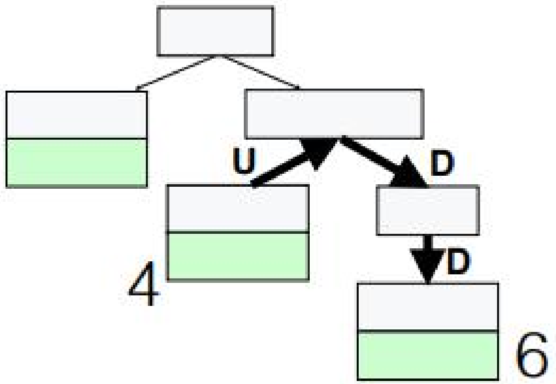
1: 000 000 000
2: 100 000 000
3: 010 000 000
4: 100 010 000
5: 010 010 000
6: 100 010 010

n_w——the maximum number of children node
n_d——the maximum depth of the tree

Empirical Study of Transformers for Source Code (FSE/ESEC)

Tree relative attention

(e) Tree relative attention:



	1	2	3	4	5	6
1	I	D	D	DD	DD	DDD
2	U	I	UD	UDD	UDD	UDDD
3	U	UD	I	D	D	DD
4	UU	UUD	U	I	UD	UDD
5	UU	UUD	U	UD	I	D
6	UUU	UUUD	UU	UUD	U	I

$$\tilde{\alpha}_{ij} = \frac{\exp(a_{ij} \cdot r_{ij})}{\sum_j \exp(a_{ij} \cdot r_{ij})}$$

Empirical Study of Transformers for Source Code (FSE/ESEC)

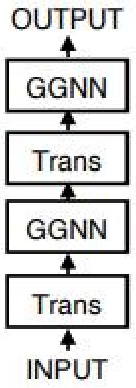
GGNN Sandwich

(f) GGNN Sandwich:

Types of edges:

- Parent (P)
- Left (L)
- Child (C)
- Right (R)
- Self (S)

	1	2	3	4	5	6
1	S	P, L	P			
2	C, R	S	L			
3	C	R	S	P, L	P	
4			C, R	S	L	
5			C	R	S	P, L
6					C, R	S



Empirical Study of Transformers for Source Code (FSE/ESEC)

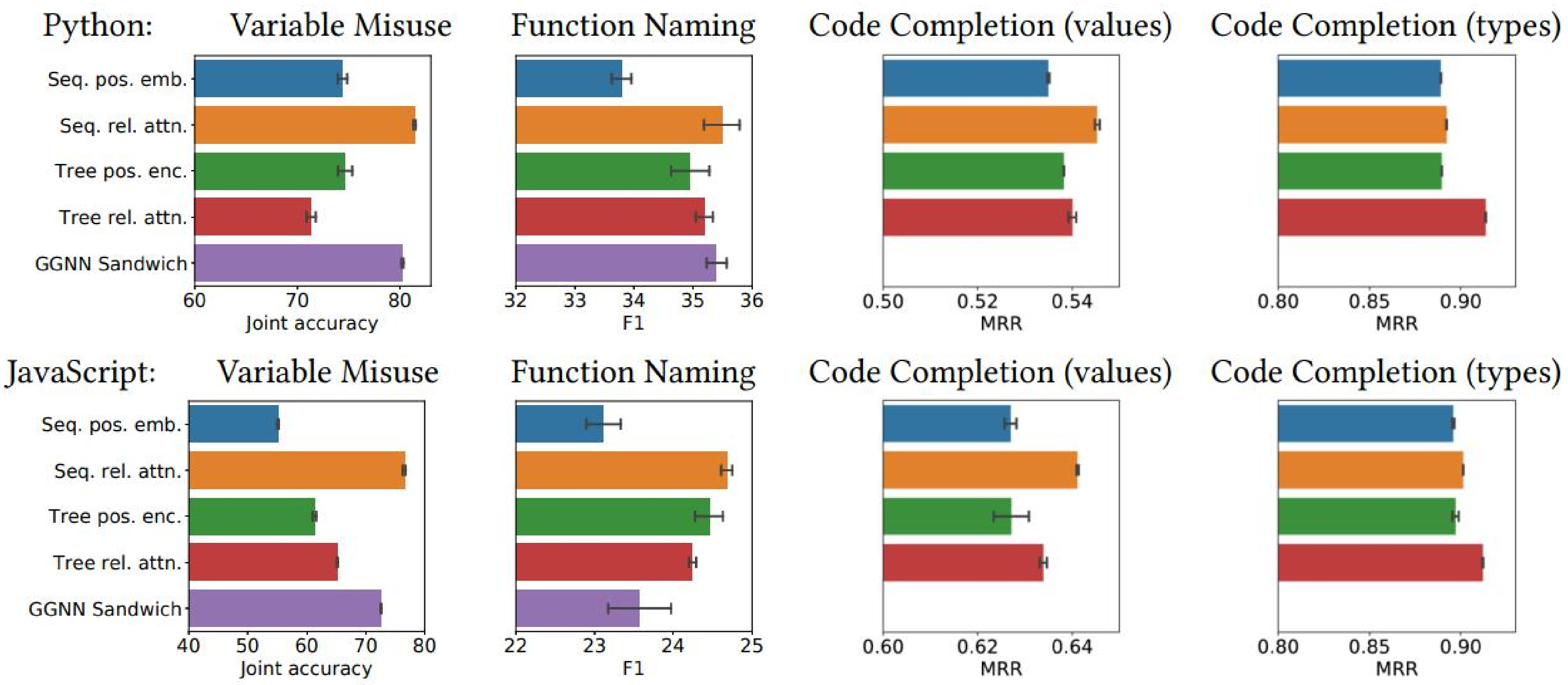


Figure 2: A comparison of different mechanisms for processing AST structure in Transformer, in the full-data setting. The numeric data for barplots is given in Supplementary materials.

Empirical Study of Transformers for Source Code (FSE/ESEC)

Table 2: Time- and storage-consumption of different structure-capturing mechanisms for the variable misuse task on the Python dataset.

Model	Train time (h/epoch)	Preprocess time (ms/func.)	Add. train data (GB)
Seq. pos. emb.	2.3	0	0
Seq. rel. att.	2.7	0	0
Tree pos. enc.	2.5	0.4	0.3
Tree rel. attn.	3.9	16.7	18
GGNN Sandwich	7.2	0.3	0.35

Empirical Study of Transformers for Source Code (FSE/ESEC)

Table 3: Comparison of combinations of sequential relative attention (SRA) with other structure-capturing approaches. All numbers in percent, standard deviations: VM: 0.5%, FN: 0.4%, CC: 0.1%. Bold emphasizes combinations that significantly outperform SRA. *In the VM task, SRA+GGNN Sandwich significantly outperforms SRA during the first half of epochs, but loses superiority at the last epochs, for both datasets. On the Python dataset, SRA+GGNN Sandwich outperforms SRA by one standard deviation at the last epoch.

	Model	VM	FN	CC (val.)
PY	SRA	81.42	35.73	54.53
PY	SRA + Seq. pos. emb.	80.77	33.99	54.37
	SRA + Tree pos. enc.	81.73	34.71	54.63
	SRA + Tree rel. attn.	81.58	35.41	**54.91**
	SRA + GGNN sand.	82.00*	33.39	N/A
JS	SRA	76.52	24.62	64.11
JS	SRA + Seq. pos. emb.	73.17	23.09	63.97
	SRA + Tree pos. enc.	74.73	23.70	**64.49**
	SRA + Tree rel. attn.	76.34	24.71	**64.79**
	SRA + GGNN sand.	75.33*	21.44	N/A

Empirical Study of Transformers for Source Code (FSE/ESEC)

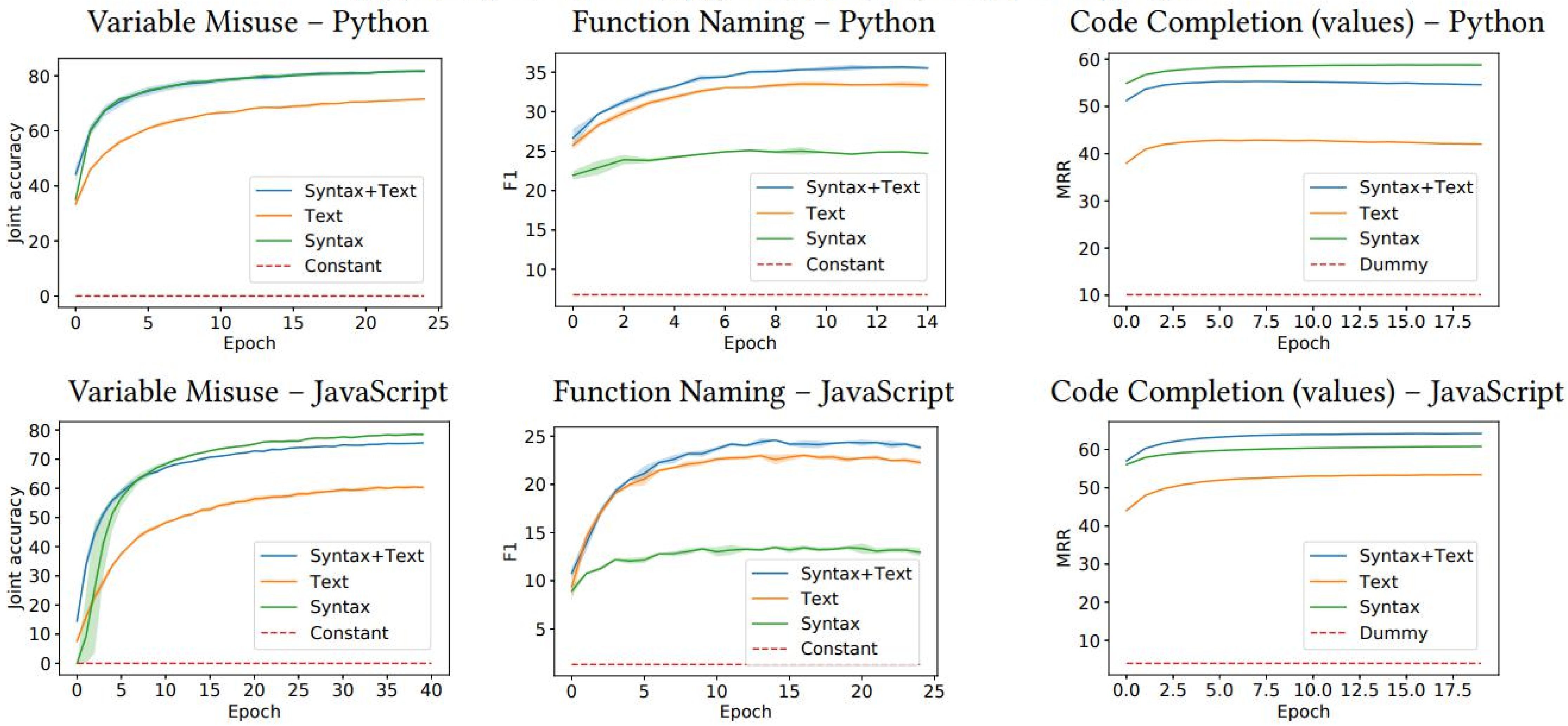


Figure 3: Comparison of syntax-based Transformer models with text-only and constant baselines.

Thanks!