

Are We Building on the Rock? On the Importance of Data Preprocessing for Code Summarization

Lin Shi, Fangwen Mu, Xiao Chen, Song Wang,
Junjie Wang, Ye Yang, Ge Li, Xin Xia, Qing Wang

Proceedings of the 30th ACM Joint European Software Engineering Conference and
Symposium on the Foundations of Software Engineering

発表者：松尾 和輝

背景

- コード要約 (Code Summarization)
 - ソースコードを自然言語に翻訳する手法
 - 開発者がコードを理解し効果的に文書化できるようにする
 - ソフトウェアの開発と保守の促進につながる
 - コードスニペットをコメントに変換
 - 現在に至るまで様々な手法が提案されてきた
 - IRベースの手法
 - NMTベースの手法

問題点

- 利用するデータセットの品質が考慮されていない
 - コード要約手法はデータセットに依存
 - ベンチマークデータセットを使用
 - Funcom [41], TLC [33], CSN [35], PCSD [67]
 - 自然言語の違い, ToDoコメント, 不完全なコメントなどのノイズが存在
 - 精度低下の原因となる

本研究の目的 ベンチマークデータセットの品質調査, および改善方法の提案

[41] Alexander LeClair, Siyuan Jiang, and Collin McMillan. [n. d.]. A neural model for generating natural language summaries of program subroutines. In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). 795-806.

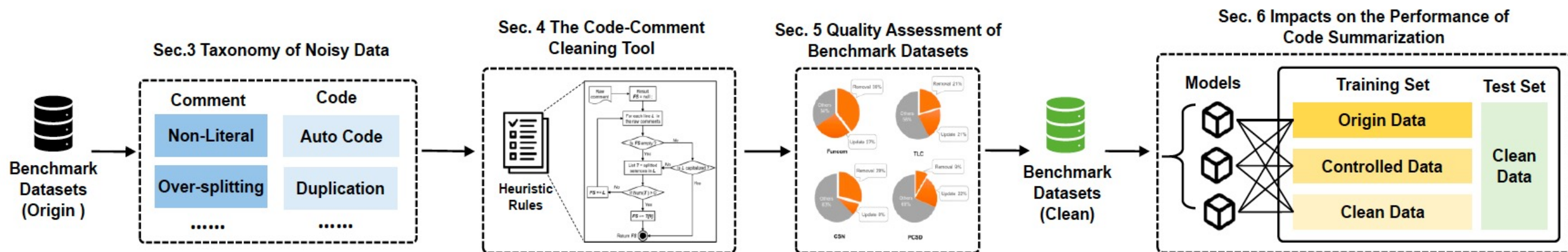
[33] Xing Hu, Ge Li, Xin Xia, David Lo, Shuai Lu, and Zhi Jin. 2018. Summarizing source code with transferred api knowledge. (2018).

[35] Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Codesearchnet challenge: Evaluating the state of semantic code search. arXiv preprint arXiv:1909.09436 (2019).

[67] Yao Wan, Zhou Zhao, Min Yang, Guandong Xu, Haochao Ying, Jian Wu, and Philip S Yu. 2018. Improving automatic source code summarization via deep reinforcement learning. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. 397-407.

研究手法

- Step
 - ①ノイズデータの分類
 - ②ノイズデータをクリーニングするツールの提案
 - ③ベンチマークデータセットの品質評価
 - ④既存のコード要約手法に対するノイズの影響調査



対象データ

- コード要約研究でよく利用される4種類のベンチマークデータセットを使用
 - Funcom [41], TLC [33], CSN [35], PCSD [67]
 - コードとコメントのペアが含まれる
 - メソッドレベルのコード
 - コメントは, メソッドに関するJavadoc (Java) , Docstrings (Python) を対象

Name	Funcom	TLC	CSN	PCSD
Year	2019	2018	2019	2017
Source	Sourcerer	Github	Github	Github
Download	[4]	[2]	[3]	[1]
Language	Java	Java	Java	Python
#Pairs	2,149,121	87,136	496,688	105,540
Train/Val/Test	9/0.5/0.5 by project	8/1/1 by function	8/1/1 by project	6/2/2 by function
Trained-on Models	[23, 41, 42]	[33, 72, 81]	[17, 35, 52]	[22, 67, 75]
	[29, 40, 73]	[8, 61, 80]	[59, 60, 69]	[8, 23, 72]
	[12, 28, 61]	[16, 22, 60]	[21, 46, 70]	[22, 75, 80]
	[26, 39, 47]	[78, 79]	[25, 43, 48]	[16, 68, 78]

①ノイズデータの分類 | 手順

- 9人の参加者によるオープンカードソート^[56]を実施
 - ラウンドごとに4つのベンチマークデータセットから160組（各40組）のコード・コメントのペアをランダムに抽出
 - 最初のラウンドでは，各参加者がそれぞれラベル付け
 - 分類を一致させるためのディスカッションを実施
 - 分類カテゴリを作成
 - 次のラウンドでは，カテゴリを元にラベル付け
 - 新たなカテゴリがあれば追加
 - 計10ラウンド実施
 - 1,600組のコードと対応するコメントにラベル付け（各400組）

①ノイズデータの分類 | 結果

- 12種類に分類
 - コメントに関するノイズ
 - 7種類
 - コードに関するノイズ
 - 5種類

例)Partial Sentence ※他の例はP17-P21に

下図のように、コメントが2行あるが、
Funcomでは最初の1行のみを収集している

```
/* Returns the high-value  
 * for an item within a series. */  
Comment (Funcom): returns the high value
```

Category	
Comment	Partial Sentence
	Verbose Sentence
	Content Tampering
	Over-Splitting
	Non-Literal
	Interrogation
	Under-Development
Code	Empty Function
	Commented-Out Method
	Block-Comment Code
	Auto code
	Duplicated Code

②クリーニングツールの提案 | 手順

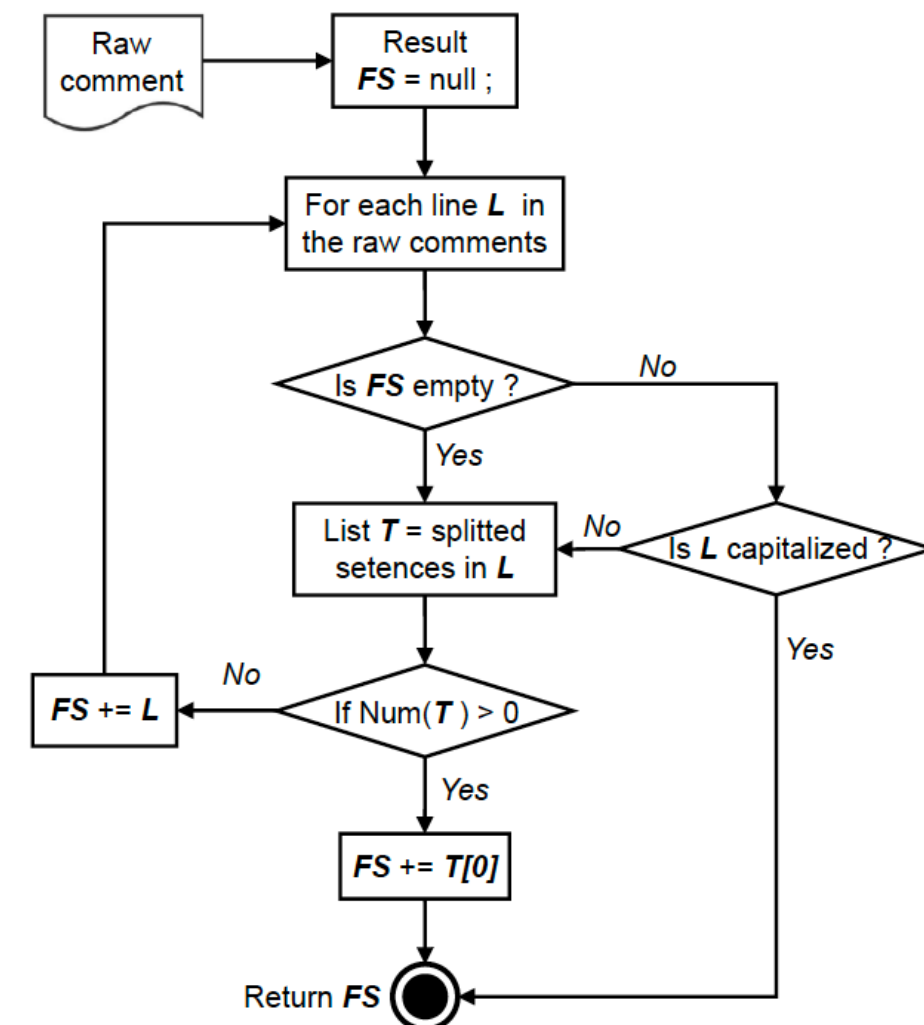
- ①で作成したラベル付きデータの80%から、ノイズを検出するif-elseルールを設計

例) Partial, Verbose Sentence

生コメントの各行が完全な文を含んでいるかどうか

Yes : 最初の完全な文を返す

No : その行の内容を保存, 次の行の判定を続行



②クリーニングツールの提案 | 手順

- 他のカテゴリの手続きについて

	Category	Syntax Feature	Action
Comment	Partial Sentence	Shorter than the corrected first sentence	UPDATE: Replace with the corrected first sentence
	Verbose Sentence	Longer than the corrected first sentence	UPDATE: Replace with the corrected first sentence
	Content Tampering	HTML tags, Doc tags, and URL format	UPDATE: Clean the tags from comment data
	Over-Splitting	Split comments on camel case and underscore	UPDATE: Replace the over-splitting variables with the original ones
	Non-Literal	non-ASCII	REMOVE
	Interrogation	“?”, “what”, “how”, <i>etc.</i>	REMOVE
	Under-Development	“todo”, “deprecate”, “copyright”, “FIXME:”, <i>etc.</i>	REMOVE
Code	Empty Function	The method body is empty	REMOVE
	Commented-Out Method	The whole method is commented out.	REMOVE
	Block-Comment Code	The method contains the block comment.	UPDATE: Clean the blocked comments from the code body
	Auto code	setter, getter, tester, <i>etc.</i>	REMOVE
	Duplicated Code	Exact Match	REMOVE

②クリーニングツールの提案 | 評価

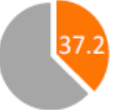
- 残りの20%のデータを用いて、ノイズを検出できるかを評価
 - 95~98%と高精度であった

Category		Dataset			Performance (%)		
		#Annotations (100%)	Rule-Build (80%)	Rule-Test (20%)	P	R	F1
Comment	Partial Sentence	176	135	41	97.5	95.1	96.3
	Verbose Sentence	129	111	18	94.7	100.0	97.3
	Content Tampering	147	120	27	92.9	96.3	94.6
	Over-Splitting	84	63	21	90.9	95.2	93.0
	Non-Literal	38	30	8	100.0	100.0	100.0
	Interrogation	16	7	9	100.0	88.9	94.1
	Under-Development	57	92	57	91.5	94.7	93.1
	Total	647	558	181	95.4	95.8	95.5
Code	Empty Function	21	14	7	100.0	100.0	100.0
	Commented-Out Method	4	2	2	100.0	100.0	100.0
	Block-Comment Code	44	31	13	100.0	92.3	96.0
	Auto Code	179	133	46	97.7	93.5	95.6
	Duplicated Code	22	16	6	100.0	100.0	100.0
	Total	270	196	74	99.6	97.2	98.3

③ベンチマークの品質

- ②で作成したツールを用いて、ベンチマークデータセット内にどれだけノイズがあるか分析

- 4つのベンチマークデータセットには、31.2%から65.8%の範囲でノイズデータが存在
- よくみられたノイズは、自動生成されたコード (Auto Code) , 無意味なテキスト (Content tampering) など

Category of Noisy Data		Funcom (%)	TLC (%)	CSN (%)	PCSD (%)
Total					
Comment	Partial Sentence	17.1	0.0	7.8	15.9
	Verbose Sentence	0.0	22.8	0.0	7.8
	Content Tampering	9.7	3.2	24.4	0.5
	Over-Splitting	24.1	0.0	0.0	0.0
	Non-Literal	0.5	0.0	7.8	0.2
	Interrogation	0.7	0.9	0.7	0.3
	Under-Development	3.7	1.2	1.2	2.3
	Total	40.9	25.4	36.1	26.5
Code	Empty Function	1.6	1.1	0.0	0.0
	Commented-Out Method	0.2	0.0	0.0	0.0
	Block-Comment Code	11.1	0.0	0.0	0.0
	Auto Code	29.8	4.6	1.6	4.3
	Duplicated Code	0.6	18.4	0.0	1.5
	Total	40.7	22.6	1.6	5.8
Removed noisy data		38.7	21.1	29.2	9.3
Updated noisy data		27.1	20.8	8.0	21.9

④ノイズの影響調査 | 手順

- 既存のコード要約手法に対して，ベンチマークのデータとクリーニングしたデータのそれぞれを学習
 - － 精度に違いが出るか調査
 - － データセット
 - Origin：ベンチマークに含まれるデータの数
 - Filtered：クリーニング後のデータの数
 - Controlled：OriginのデータをFilteredと同じ数だけ減らしたもの

			. Funcom	TLC	CSN	PCSD
Total	Origin		2,149,121	87,136	496,688	105,540
	Filtered		1,316,532	68,743	351,394	95,793
Experimental	Train	Origin	1,937,136	69,708	454,451	63,324
		Controlled	1,184,438	53,597	323,226	57,849
		Filtered	1,184,438	53,597	323,226	57,849
	Test	Filtered	69,392	7,584	19,319	19,028

④ノイズの影響調査 | 手順

- ベースライン
 - NNGen^[45] : IRベースのモデル
 - NCS^[8] : NMTベースのモデル
 - Rencos^[78] : IRとNMTのハイブリッドモデル
- 評価指標
 - BLEU^[51], METEOR^[11], ROUGE-L^[44], CIDEr^[66]
 - BLEU : 参照文と生成文のn-gramでの一致率
 - METEOR : 1-gramでのPrecisionとRecallの調和平均を調整したもの
 - ROUGE-L : 参照文と生成文の最長共通部分列の長さ
 - CIDEr : 参照文と生成文のTF-IDFを計算, 平均コサイン類似度を算出

[45] Zhongxin Liu, Xin Xia, Ahmed E Hassan, David Lo, Zhenchang Xing, and Xinyu Wang. 2018. Neural-machine-translation-based commit message generation: how far are we?. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. 373-384.

[8] Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. [n. d.]. A Transformer-based Approach for Source Code Summarization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020. 4998-5007.

[78] Jian Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, and Xudong Liu. 2020. Retrieval-based neural source code summarization. In 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE). IEEE, 1385-1397.

[51] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. [n. d.]. Bleu: a Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA. 311-318.

[11] Satanjeev Banerjee and Alon Lavie. [n. d.]. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005. 65-72.

[44] Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In Text summarization branches out. 74-81.

[66] Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. IEEE Computer Society, 4566-4575. <https://doi.org/10.1109/CVPR.2015.7299087>

④ノイズの影響調査 | 結果

※紙面の都合上Funcomの結果のみ（他の結果はP.22に）

- 全体的に精度が向上
 - ノイズを除去するだけで，精度の向上につながる

Benchmark	Model	Train set	Training Hours	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE	METEOR	CIDEr
Funcom	NNGen	Origin	5h	23.87	14.28	11.4	10.05	26.88	12.59	1.26
		Controlled	5h	21.93	12.19	9.34	8.09	24.84	11.39	1.05
		Filtered	5h	24.58 3.0% ↑	15.26 6.9% ↑	12.49 9.6% ↑	11.2 10.3% ↑	27.08 0.7% ↑	13.24 5.2% ↑	1.38 9.5% ↑
	NCS	Origin	20h	29.95	17.79	10.2	6.42	34.84	16.14	1.38
		Controlled	20h	29.33	17.06	9.78	5.31	34.05	15.65	1.42
		Filtered	20h	30.53 1.9% ↑	18.79 5.6% ↑	11.47 12.5% ↑	7.64 16.0% ↑	35.42 1.7% ↑	16.32 1.1% ↑	1.46 5.8% ↑
	Rencos	Origin	9h	27.23	15.97	9.62	6.43	31.97	14.32	1.25
		Controlled	9h	26.90	15.71	9.48	6.42	31.79	14.16	1.23
		Filtered	9h	27.92 2.5% ↑	16.8 5.2% ↑	10.61 10.3% ↑	7.44 13.6% ↑	32.51 1.7% ↑	14.50 1.3% ↑	1.31 4.8% ↑

まとめ

- 目的
 - ベンチマークデータセットの品質評価, および改善方法の提案
- アプローチ
 - ノイズをカテゴリに分類
 - カテゴリごとにif-elseルールを作成
- 結果
 - ベンチマークには多数のノイズが含まれていることを明らかにした
 - クリーニングしたデータを利用することで精度が向上することを示した

所感

- データをきれいにするだけで精度が向上するのになんでやらの？
 - 各々がおこなっている前処理で、キチンとデータがクリーニングできているかを検証していないのか
- ルールベースの手法で検出できるんや
 - もっとめんどくさそうなイメージ

補足 | ①ノイズデータの分類例

- Verbose Sentence
 - 文が冗長

```
"""
Generate a CSV file containing a summary of the xBlock usage
Arguments:course_data
"""
Comment (PCSD): generate a csv file containing a summary of
the xblock usage arguments course data
```

- Content Tampering
 - pなどのいないものが混ざっている

```
/* <p> Builds the JASPIC application context.</p> */
Comment (CSN): p builds the jaspic application context p
```

補足 | ①ノイズデータの分類例

- Over-Splitting
 - コメントを過剰に分割している

```
/* This method initializes jTextField. */
```

```
Comment (Funcom): this method initializes j text field
```

- Non-Literal
 - 様々な言語が混じっている

```
/* 将JSONArray转换为Bean的List, 默认为ArrayList */
```

```
Comment (CSN): jsonarray bean list arraylist
```

補足 | ①ノイズデータの分類例

- Interrogation

- 質問している

```
/* Do we need to show the upgrade wizard prompt? */  
public boolean isDue() {  
    if (isUpToDate)  
        return false; ...
```

Comment (CSN): do we need to show the upgrade wizard prompt ?

- Under-Development

- コメントがきちんと書かれていない

```
/* Description of the Method */  
protected void openFile(File f) {  
    if (f == null) { ...
```

Comment (TLC): description of the method

補足 | ①ノイズデータの分類例

- Empty Function
 - 空っぽの関数

```
/*Specifies the behaviour of the automaton in its end state*/  
protected void end(){}  
Code (Funcom): protected void end
```

- Commented-Out Method
 - コメントアウトされたメソッド

```
/* for now try mappig full type URI */  
// public String transformTypeID(Uri typeuri){  
// return typeuri.toString();}  
Code (Funcom): public string transform type id ...
```

補足 | ①ノイズデータの分類例

- Block-Comment Code
 - コードの中にコメントが含まれる

```
/* Get GPS Quality Data */  
public int getFixQuality(){  
    checkRefresh();  
    // TODO: Why is he using Math.round?  
    Return Math.round(quality);}  
Code (Funcom): public int get fix quality check refresh todo  
why is he using math round return math round quality
```

- Auto code
 - 自動生成されたコード

```
/* Test the constructor */  
public void testConstructor() {  
    System TestResult str;  
    System TestID testID1; ...  
Comment (Funcom): test the constructor  
Code (Funcom): public void test constructor ...
```

- Duplicated Code
 - 重複したコード

補足 | ④ノイズの影響調査

Benchmark	Model	Train set	Training Hours	BLEU-1		BLEU-2		BLEU-3		BLEU-4		ROUGE		METEOR		CIDEr	
Funcom	NNGen	Origin	5h	23.87		14.28		11.4		10.05		26.88		12.59		1.26	
		Controlled	5h	21.93		12.19		9.34		8.09		24.84		11.39		1.05	
		Filtered	5h	24.58	3.0% ↑	15.26	6.9% ↑	12.49	9.6% ↑	11.2	10.3% ↑	27.08	0.7% ↑	13.24	5.2% ↑	1.38	9.5% ↑
	NCS	Origin	20h	29.95		17.79		10.2		6.42		34.84		16.14		1.38	
		Controlled	20h	29.33		17.06		9.78		5.31		34.05		15.65		1.42	
		Filtered	20h	30.53	1.9% ↑	18.79	5.6% ↑	11.47	12.5% ↑	7.64	16.0% ↑	35.42	1.7% ↑	16.32	1.1% ↑	1.46	5.8% ↑
	Rencos	Origin	9h	27.23		15.97		9.62		6.43		31.97		14.32		1.25	
		Controlled	9h	26.90		15.71		9.48		6.42		31.79		14.16		1.23	
		Filtered	9h	27.92	2.5% ↑	16.8	5.2% ↑	10.61	10.3% ↑	7.44	13.6% ↑	32.51	1.7% ↑	14.50	1.3% ↑	1.31	4.8% ↑
TLC	NNGen	Origin	<1h	32.58		24.16		21.92		20.74		36.07		18.14		2.01	
		Controlled	<1h	39.84		32.01		29.24		27.51		43.57		23.22		2.64	
		Filtered	<1h	46.88	43.9% ↑	39.27	62.5% ↑	36.81	67.9% ↑	35.19	41.1% ↑	49.08	36.1% ↑	25.53	40.7% ↑	3.62	80.5% ↑
	NCS	Origin	6h	42.09		32.95		29.09		27.09		46.30		24.18		2.65	
		Controlled	6h	39.28		29.61		25.83		23.89		43.49		22.11		2.37	
		Filtered	6h	46.52	10.5% ↑	37.19	12.9% ↑	33.41	14.9% ↑	31.38	13.7% ↑	49.40	6.7% ↑	24.67	2.0% ↑	3.30	24.5% ↑
	Rencos	Origin	6h	43.66		34.82		31.29		29.19		47.87		24.95		2.81	
		Controlled	6h	43.71		34.89		31.21		28.93		47.85		25.37		2.84	
		Filtered	6h	51.54	18.0% ↑	42.90	23.2% ↑	39.22	25.3% ↑	37.00	21.1% ↑	54.25	13.3% ↑	28.21	13.1% ↑	3.88	38.1% ↑
CSN	NNGen	Origin	<1h	14.86		6.08		4.07		3.42		18.04		8.54		0.40	
		Controlled	<1h	13.95		5.09		3.21		2.62		17.08		7.97		0.34	
		Filtered	<1h	19.89	33.8% ↑	8.28	36.2% ↑	5.72	40.5% ↑	4.96	31.0% ↑	23.17	28.4% ↑	9.67	13.2% ↑	0.65	62.5% ↑
	NCS	Origin	15h	25.47		12.34		5.81		3.02		30.47		12.48		0.80	
		Controlled	15h	25.45		12.29		5.68		2.88		31.17		12.30		0.82	
		Filtered	15h	28.68	12.6% ↑	14.01	13.5% ↑	6.96	19.8% ↑	3.87	22.0% ↑	34.29	12.5% ↑	13.84	10.9% ↑	0.95	18.8% ↑
	Rencos	Origin	11h	16.99		7.65		4.09		2.64		20.91		8.33		0.49	
		Controlled	11h	16.30		7.09		3.75		2.43		20.00		8.13		0.44	
		Filtered	11h	24.72	45.5% ↑	11.36	48.5% ↑	6.51	59.2% ↑	4.56	42.1% ↑	29.35	40.4% ↑	11.52	38.3% ↑	0.82	67.3% ↑
PCSD	NNGen	Origin	<1h	22.52		15.48		12.63		10.45		24.90		12.97		1.24	
		Controlled	<1h	21.81		14.77		11.99		9.91		24.16		12.49		1.18	
		Filtered	<1h	25.96	15.3% ↑	18.91	22.2% ↑	16.27	28.8% ↑	14.00	25.4% ↑	27.68	11.2% ↑	15.09	16.3% ↑	1.63	31.9% ↑
	NCS	Origin	6h	28.14		18.69		14.28		11.36		32.95		16.30		1.61	
		Controlled	6h	26.85		17.42		13.05		10.17		31.77		15.42		1.49	
		Filtered	6h	37.33	32.7% ↑	24.74	32.4% ↑	19.49	36.5% ↑	16.48	31.1% ↑	40.93	24.2% ↑	18.67	14.5% ↑	2.07	28.6% ↑
	Rencos	Origin	5h	30.37		21.27		16.42		12.93		33.66		17.40		1.65	
		Controlled	5h	29.73		20.55		15.71		12.37		33.05		16.96		1.59	
		Filtered	5h	33.59	10.6% ↑	24.14	13.5% ↑	19.63	19.5% ↑	16.10	19.7% ↑	36.15	7.4% ↑	19.18	10.2% ↑	2.02	22.4% ↑