Enrique Dehaerne[1,2] Bappaditya Dey[2] Sandip Halder[2] Stefan De Gendt[1,2] Wannes Meert[1]

[1] KU Leuven – Faculty of Science; Leuven, Belgium
[2] IMEC – Advanced Patterning Group; Leuven, Belgium

A Machine Learning Approach Towards SKILL Code Autocompletion

1. Introduction

- Moore's Law increases the complexity of electronic systems → Need innovation in EDA.
- Recently, advances in ML-based code generation via transformer language models[1] (E.g., ChatGPT[2]).
- However, very little SKILL[3] code data is publicly available

Problem statement:

Can we train ML models, in a data-efficient way, to autocomplete SKILL code towards automated layout/logic synthesis?

```
/*Create a Pcell that generates rectangles along the x axis which fit within a
bounding box with width 15. This pcell accepts the following parameters :
width   X - dimension of the rectangle. ( float, default = 1. 0 )
length  Y - dimension of the rectangle. ( float, default = 5. 0 )
cd1         Space between rectangles in the X - dimension ( float, default = 1. 5 )
layer   layer type of the rectangles ( string, default = " metal1 ") */

pcDefinePCell(
   list(ddGetObj("pcells") "rectanglesRow" "layout")

   ( (width 1.0) (length 5.0)
     (cd1 1.5) (layer "metal1") )

   let((tfId bbox)
      bbox = list(0:0 15:length+1)
      masterRect = rodCreateRect(
         ?cvId   pcCellView
         ?layer  list(layer "drawing")
         ?width  width
         ?length length
         ?spaceX cd1
         ?fillBBox bbox)
   )
)
```

Input

Output

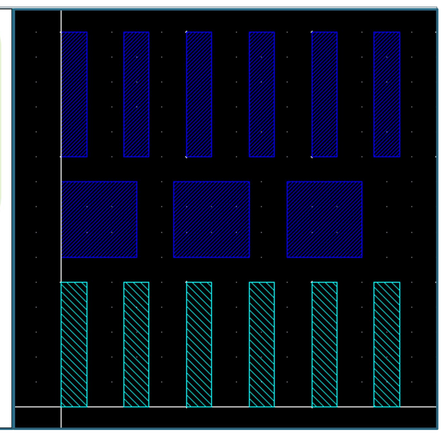

Fig 1. Left: example SKILL parameterized cell (PCell) code snippet. Right: different instantiations of the Pcell.

2. Dataset

- Total number of files/code snippet pairs: 3248/13,613
- Very small dataset!

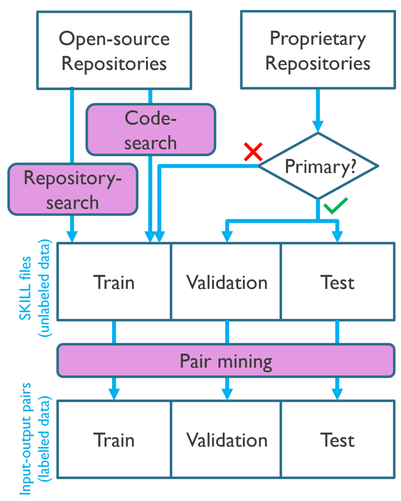

Fig 2. Dataset creation method.

3. Proposed Methodology

- Model pretrained on general programming language code (Python, Java, etc.)
  - T5[4] models: designed for transfer learning
- Finetune on full SKILL files (unsupervised) + input-output pairs (supervised) [see left Fig 1.]
- Filter files for quality and deduplicate input-output pairs

Evaluation: ablation of proposed method components (a.k.a. training strategies in Fig 3.)

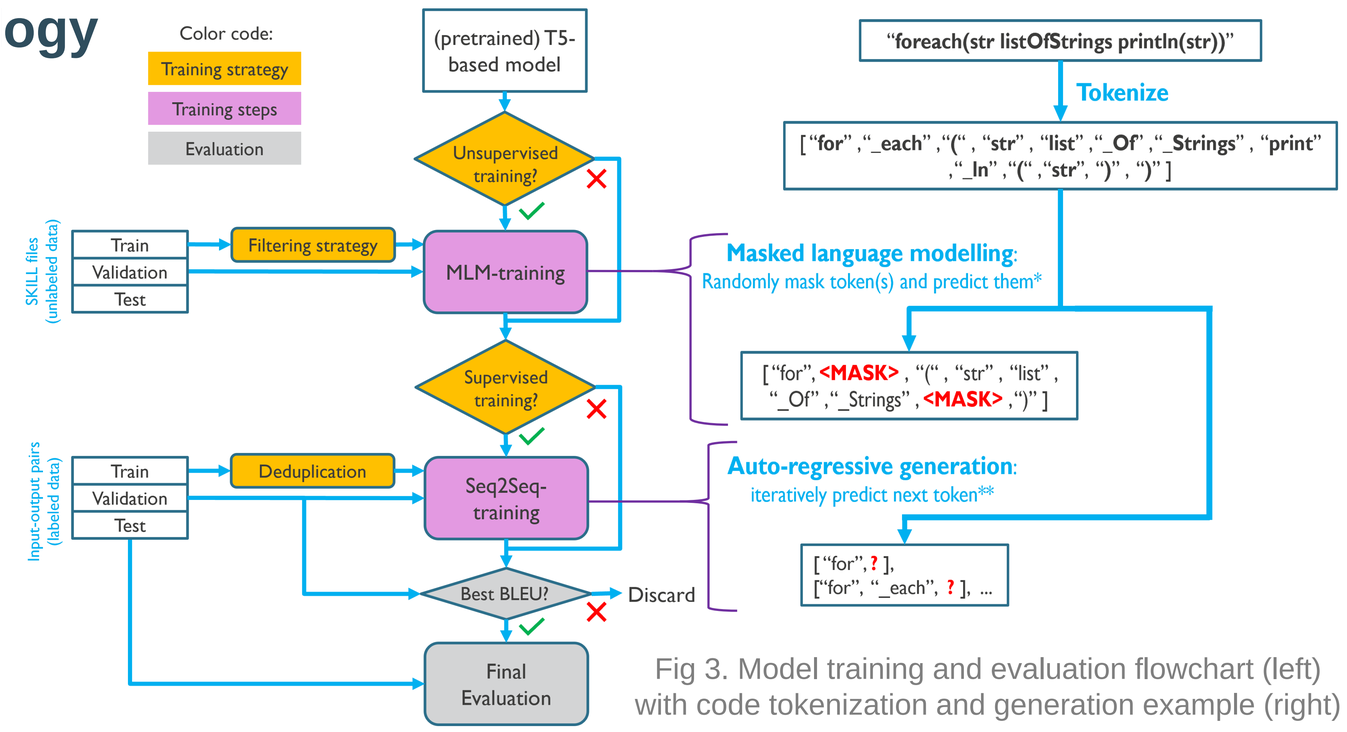

Fig 3. Model training and evaluation flowchart (left) with code tokenization and generation example (right)

4. Results

- Validation (compare models of same type [=pretraining])
  - file filtering
  - input-output pair deduplication
- Final evaluation (compare between model types):
  - Pretraining on general code
  - Unsupervised + supervised finetuning
- However, problems in generated SKILL code:
  - Repeating common tokens & copying tokens from input
  - Not compilable, in general

Curated dataset results:

- Number of filtered files: 2317-1857 filtered files (3248 total)
- Number of deduplicated code snippet pairs: 7069 (13,613 total)

Model type	Rank	Unsupervised training			Supervised training		BLEU
		applied?	file filtering?	include comments?	applied?	pair types?	
CodeTrans	1	✓	lint pass	✓	✓	deduplicated	0.0864
	3	✗	NA	NA	✓	deduplicated	0.0676
CodeT5	1	✓	lint IQ $\geq$ 10	✗	✓	deduplicated	0.0470
T5-NL	1	✓	-	✗	✗	NA	0.0623
	4	✓	$\geq$ 1 pairs mined	✓	✓	deduplicated	0.0522
T5-SKILL	1	✓	-	✗	✗	NA	0.08285
	9	✓	$\geq$ 1 pairs mined	✓	✓	deduplicated	0.0376

Tab 1. Validation results (best models for each model type).

Model	BLEU	T5-SKILL-BLEU	Mean Δ Lint IQ	Mean Human Score
CodeTrans	**0.019**	**0.032**	**-60.7**	1.93
CodeT5	0.008	0.015	**-60.3**	**2.27**
T5-NL	0.004	0.007	-70.0	1.46
T5-SKILL	0.016	0.016	-71.7	1.13
CodeTrans Sup-Only	0.014	0.024	**-59.4**	1.40
T5-NL +Sup	0.002	0.002	**-59.8**	1.67
T5-SKILL +Sup	0.002	0.002	-64.4	1.00

Tab 2. Final evaluation results of best validation models (see Tab. 1).

5. Conclusion

- Proposed data efficient method for training ML language model to autocomplete SKILL code.
- Proposed method performs better than ablation baselines using limited SKILL dataset.
- SKILL code generated not (yet) consistently compilable or executable

Limitations and future work:

- More data needed (confidentiality is a big roadblock) → Privacy-preserving ML
- Model size (bigger models usually better, require lots of compute and centralization) → Distillation
- Evaluation (how to make sure the model is consistently generating good SKILL code?) → Functional correctness

Key References

1. Dehaerne et al., "Code Generation Using Machine Learning: A Systematic Review," (2022).
2. Oulang et al., "Training language models to follow instructions with human feedback", (2022).
3. Barnes, "SKILL: a CAD system extension language", (1990).
4. Raffel et al., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer" (2019).