Prompt Engineering for the PyCRAM Experiments

J.-P. Töberg, P. Frese & P. Cimiano

April 8, 2024

Abstract

In this experiment, we establish a prompt for generating PyCRAM action designators using the same one-shot prompting method we introduced in [1]. First, we create a basic prompt structure based on manual testing. Our main objective of this experiment is to determine the appropriate level of detail for the action descriptions in the prompt. We also describe how we adapt the engineered prompt for an additional experiment, in which we convert CRAM designators into PyCRAM designators.

1 Establishing a Prompt for Generating PyCRAM Designators

Prior to performing the experiment, we establish a prompt for generating a target PyCRAM designator based on a another reference PyCRAM designator. We design a template, which we refer to as the *prompt structure*, containing a number of input slots [2]. The prompt structure is displayed in Figure 1. We use a prompting function to fill the input slots based on the combination of reference and target actions and the specific action descriptions for the generation task. The resulting prompt is sent to the LLM, which generates an answer, from which we extract the generated designator.

Considering the technical details of the employed LLMs, our prompt structure consists of two parts, the system message and the user message. In the *system message*, we provide the name and the action description for the reference action, as well as the reference PyCRAM designator. Based on the recommendation by Singh et al. [3], we inform the LLM of the available libraries and the predefined designators in the PyCRAM framework. For this purpose, we include import statements in the reference designator and we insert the constructors of the predefined designators as comments beneath the respective import statements. Additionally, we include natural language comments in the reference to give the LLM a better understanding of the action's sub-tasks [3].

The *user message* contains the request for generating a new PyCRAM designator. For this purpose, we specify the name and the action description for the

System message: The following Python source code describes a PyCRAM designator for the action of "*[reference action]*", where the robot would *[reference action description]*
```python
[reference designator]
```

User message: Take the example and create a new designator for the action "*[target action]*", where the robot should *[target action description]*. Output only the designator with no additional text. Do not include comments in the code. Use only the imported libraries and designators. Use the following constructor: *[target action constructor]*

Figure 1: Prompt structure for generating a target PyCRAM designator based on a reference PyCRAM designator.

target action. We also define specific restrictions on the format of the generated designator to facilitate its evaluation. Finally, we provide the constructor for the target designator to ensure the generated designator is compatible with the existing PyCRAM framework.

1.1 Experiment Design

After establishing a prompt, our next objective is to determine the optimal level of detail for both the reference and target action descriptions. We examine three action descriptions with varying levels of detail for each of the six actions, with 1 referring to the lowest level of detail and 3 referring to the highest level of detail. The used descriptions for each level are listed in Table 1.

Our hypothesis for this experiment is that providing more detail in both the reference and target action descriptions positively influences the quality of the generated designators. We also expect the reference action descriptions to have a smaller impact on the generation performance than the target action descriptions.

In order to examine these hypotheses, we vary the levels of detail for the reference and target action descriptions independently of each other. For each combination of the levels of detail ($3 \times 3 = 9$ combinations), we use each of the six actions once as the reference and once as the target. This results in a total of 54 possible combinations, which we generate using the `gpt-4-0613` model. We analyse the generated designators using the code generation metrics CodeBERTScore (CBS) [4], CodeBLEU (CoB) [5], and CrystalBLEU (CrB) [6], the machine translation metrics chrF [7] and ROUGE-L (R-L) [8], as well as the string comparison metric Edit Distance (ED).

Table 1: Action descriptions for the PyCRAM designators for each level of detail.

Action	1	2	3
transport	transport an object to a position	get an object and transport it to a position using a specified arm	find an object, pick it up, transport it to a target location, and place it down using a specified arm
grasp	grasp an object	move into a a pre-grasp position and grasp an object using a specified arm	move the gripper into a pre-grasp position 10 cm before the object, open the gripper, move to the object, and close the gripper using a specified arm
cut	cut an object	cut an object in half (halving) or cut an object in slices (slicing) with a specified slice thickness using a tool in a specified arm using a specified grasp	cut an object in half (halving) by performing a vertical slicing motion along the middle of a specified object or cut an object in slices (slicing) with a specified slice thickness by performing vertical slicing motions along the width of a specified object using a tool in a specified arm using a specified grasp
mix	perform a mixing motion	perform a mixing motion using a tool held in a specified arm using a specified grasp	perform a counterclockwise, outward spiraling mixing motion in a bowl using a tool held in a specified arm using a specified grasp
pour	pour the contents of one object into another container	get a source object, transport it to the target container's location, and pour its contents into the target container for a specified duration	get a source object, transport it to the target container's location, and pour its contents into the target container for a specified duration by tilting the source object at a 90° angle, its top being located 20 cm above the target container's center
wipe	wipe a rectangular area of a surface using a cloth object	get a cloth object and wipe a rectangular area of a surface defined by the center, length, and width	get a cloth object and wipe a rectangular part of a surface defined by the center, length, and width in a zigzag pattern with a 10 cm gap between the strips

Table 2: Results of the code generation metrics for the target levels of detail, averaged over the reference levels of detail. For each action, the highest results and the selected target level of detail are marked in **bold**.

Target		Code Metrics					
Action	Detail	chrF	CBS	CoB	CrB	ED	R-L
cut	1	0.506	0.838	0.370	0.297	0.489	0.480
	2	0.535	**0.848**	0.386	**0.325**	0.492	0.509
	3	**0.536**	0.848	**0.392**	0.321	**0.494**	**0.512**
grasp	1	0.662	0.881	0.481	0.384	0.603	0.653
	2	0.740	0.898	0.553	0.440	0.660	0.661
	3	**0.792**	**0.913**	**0.596**	**0.478**	**0.662**	**0.671**
mix	1	0.473	0.827	0.320	0.169	0.406	0.434
	2	**0.503**	0.831	0.350	0.203	0.423	0.463
	3	0.500	**0.847**	**0.394**	**0.235**	**0.439**	**0.476**
pour	1	**0.655**	**0.877**	0.452	0.324	0.586	**0.589**
	2	0.620	0.875	0.438	0.289	0.566	0.589
	3	0.649	0.873	**0.468**	**0.338**	**0.594**	0.576
transport	1	0.817	0.939	0.774	0.679	0.776	0.820
	2	0.813	0.942	0.774	**0.696**	0.772	**0.822**
	3	**0.882**	**0.945**	**0.817**	0.686	**0.825**	0.811
wipe	1	0.619	0.915	0.573	0.404	0.541	**0.739**
	2	0.651	**0.916**	0.596	0.416	0.555	0.709
	3	**0.677**	0.909	**0.633**	**0.428**	**0.592**	0.697

1.2 Results

For each target action and each target level of detail, we average the results of the metrics over all the reference levels of detail. The results are displayed in Table 2. We select the target level of detail producing the highest code metric results[1] as the optimal level of detail for the action description when using this action as the target. We apply the same procedure for the reference levels of detail, averaging the results over all target levels of detail (refer to Table 3) and selecting the levels producing the highest results. The optimal target and reference levels of detail for each action are listed in Table 4.

Our results show that using the highest level of detail for the target action descriptions is optimal, which corresponds with our expectation. The results are, however, less distinct when looking at the reference levels of detail. In this case, no single level of detail is optimal for all reference actions. Instead, we

[1]If two levels of detail produce the highest results for an equal number of metrics, we assign a higher weight to the code generation metrics (i.e. CodeBERTScore, CodeBLEU & CrystalBLEU)

Table 3: Results of the code generation metrics for the reference levels of detail, averaged over the target levels of detail. For each action, the highest results and the selected reference level of detail are marked in **bold**.

Reference		Code Metrics					
Action	Detail	chrF	CBS	CoB	CrB	ED	R-L
cut	**1**	**0.750**	**0.905**	**0.559**	**0.479**	**0.666**	**0.681**
	2	0.713	0.891	0.534	0.404	0.625	0.647
	3	0.731	0.896	0.536	0.419	0.635	0.658
grasp	1	0.487	0.834	0.349	0.196	0.416	0.455
	2	0.489	0.832	0.357	0.203	0.424	0.454
	3	**0.500**	**0.839**	**0.358**	**0.208**	**0.428**	**0.463**
mix	1	0.642	**0.876**	0.446	0.316	0.582	0.586
	2	0.627	0.876	0.448	**0.326**	**0.583**	**0.603**
	3	**0.654**	0.873	**0.464**	0.310	0.581	0.565
pour	1	0.831	0.941	0.783	0.672	0.784	0.813
	2	**0.842**	**0.944**	0.790	0.692	**0.795**	**0.820**
	3	0.840	0.942	**0.791**	**0.697**	0.794	0.819
transport	1	0.646	**0.914**	0.589	0.415	0.554	0.715
	2	**0.654**	0.913	**0.610**	0.410	**0.571**	0.704
	3	0.647	0.914	0.603	**0.423**	0.564	**0.726**
wipe	1	**0.534**	**0.847**	0.380	0.312	0.496	**0.507**
	2	0.510	0.843	0.382	0.296	0.483	0.488
	3	0.533	0.843	**0.386**	**0.334**	**0.497**	0.506

Table 4: Optimal levels of detail for the action descriptions when using an action as the target or as the reference.

	Levels of Detail			Levels of Detail	
Action Name	Target	Reference	Action Name	Target	Reference
cut	3	1	grasp	3	3
mix	3	2	pour	3	2
transport	3	2	wipe	3	3

find that the level of detail should be varied depending on the specific reference action used.

We note that the averaged results of the code metrics for both the target and reference levels of detail in Tables 2 and 3 often show only minor differences between the levels of detail. Generally, the variations for the target levels of detail are greater than the variations for the reference levels of detail. This supports our assumption that the level of detail in the reference action descriptions has a smaller impact on the quality of the generated designators than the level of detail in the target action descriptions.

2 Adapting the Prompt for CRAM Conversion

Before conducting the experiment, we adapt the prompt structure shown in Figure 1. In the system message, we include a CRAM designator and the basic structure of a PyCRAM designator. We do this to ensure the generated designators are compatible with the PyCRAM framework. Within the basic structure, we also add an example of how to call one of the predefined PyCRAM designators and the previously established import statements. We modify the user message to request the conversion of the given CRAM designator into a PyCRAM designator for the same action. Additionally, we adapt the requests on the output format.

The resulting prompt structure is displayed in Figure 2. For the CRAM reference designators, we use the action descriptions from the CRAM generation experiment [1]. The target action descriptions are identical to those in the preceding PyCRAM generation experiment and we select the levels of detail according to the results in Tables 2 and 3.

References

[1] J.-P. Töberg and P. Cimiano, Generation of Robot Manipulation Plans Using Generative Large Language Models, in *2023 Seventh IEEE International Conference on Robotic Computing (IRC)* (IEEE, Laguna Hills, CA, USA, December 2023), pp. 190–197.

[2] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi and G. Neubig, Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing, *ACM Comput. Surv.* **55** 1–35 (September 2023).

[3] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason and A. Garg, ProgPrompt: Program generation for situated robot task planning using large language models, *Auton Robot* **47** 999–1012 (December 2023).

[4] S. Zhou, U. Alon, S. Agarwal and G. Neubig, CodeBERTScore: Evaluating Code Generation with Pretrained Models of Code, in *Deep Learning for*

System message: The following Lisp source code describes a CRAM designator for the action of "*[action name]*", where the robot would *[CRAM action description]*

```lisp
[CRAM designator]
```

PyCRAM designators are written in Python and have the following basic structure:

```python
[PyCRAM basic structure]
```

User message: Take the CRAM designator and convert it into a PyCRAM designator (include an implementation of the perform method) for the same action of "*[action name]*", where the robot should *[PyCRAM action description]*. Output only the designator with no additional text. Do not include comments in the code. Follow the provided basic structure and use only the imported libraries and designators. Use the following constructor: *[target action constructor]*

Figure 2: Prompt structure for generating a target PyCRAM designator based on a reference CRAM designator.

Code (DL4C) Workshop at the 11th International Conference on Learning Representations (ICLR) (Kigali, Rwanda, 2023).

[5] S. Ren, D. Guo, S. Lu, L. Zhou, S. Liu, D. Tang, N. Sundaresan, M. Zhou, A. Blanco and S. Ma, CodeBLEU: A Method for Automatic Evaluation of Code Synthesis (2020).

[6] A. Eghbali and M. Pradel, CrystalBLEU: Precisely and Efficiently Measuring the Similarity of Code, in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (ACM, Rochester MI USA, October 2022), pp. 1–12.

[7] M. Popović, chrF: Character n-gram F-score for automatic MT evaluation, in *Proceedings of the 10th Workshop on Statistical Machine Translation* 2015, pp. 392–395.

[8] C.-Y. Lin, ROUGE: A Package for Automatic Evaluation of Summaries, in *Text Summarization Branches Out* (Association for Computational Linguistics, Barcelona, Spain, 2004), pp. 74–81.