Automated Python Translation

Anonymous ARR Submission

Abstract

Python is one of the most commonly used programming languages in industry and education. Its English keywords and built-in functions/modules allow it to come close to pseudocode in terms of its readability and ease of writing. However, those who do not speak English may not experience these advantages. In fact, they may even be hindered in their ability to understand Python code, as the English nature of its terms creates an additional layer of overhead. To that end, we introduce the task of automatically translating Python's natural modality (keywords, error types, identifiers, etc.) into other human languages. This presents a unique challenge, considering the abbreviated nature of these forms, as well as potential untranslatability of advanced mathematical/programming concepts across languages. We therefore create an automated pipeline to translate Python into other human languages, comparing strategies using machine translation and large language models. We then use this pipeline to acquire translations from five common Python libraries (`pytorch`, `pandas`, `tensorflow`, `numpy`, and `random`) in seven languages, and do a quality test on a subset of these terms in French, Greek, and Bengali. We hope this will provide a clearer path forward towards creating a universal Python, accessible to anyone regardless of nationality or language background.[1]

1 Introduction

Python is not only growing to be one of the most well-known programming languages by emerging developers today, but perhaps becoming one of the most popular as well (Johnson, 2023). It is used extensively in industry, and especially in education, where teachers can leverage the English nature of its keywords and built-in functions and modules to allow students to understand code on a more simplified level. In this way Python mimics programmable pseudocode, allowing programmers to specify in near-language terms what the code is meant to do instead of memorizing superfluous lists of terminology and acronyms, worrying themselves with low-level details.

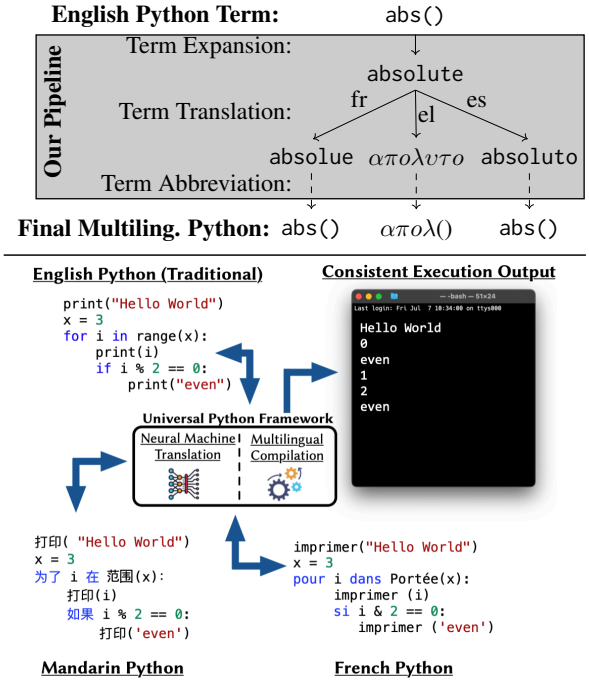

Figure 1: Illustration of our pipeline (top) with an example of the function `abs()` (absolute), which first expands English terms and then translates them into various languages, optionally abbreviating them. These translations could then be fed into the Universal Python Framework (bottom) from Otten et al. (2023).

However, while English-speakers enjoy these advantages, non-English-speakers may struggle to learn or memorize these seemingly-strange terms. Several studies show that the speed with which a student learns to program correlates with their understanding of the human language it utilizes (Hill, 2017). Additionally, Piech and Abu-El-Haija (2020) found that many users write comments/-commit messages in GitHub in their native lan-

[1]Link to Github repository, containing code and results: https://anonymous.4open.science/r/AutomatedPythonTranslation-7545/

guages, suggesting that people desire to code in non-English languages. *If we want computer science and technology to be accessible to all people of all cultures, then why is our code English-oriented?* The NLP community has seen a growing shift in attempting to broaden perspectives to other cultures. But when all of the tools are built with English programming languages, what does that say about the underlying system? A first step to rectifying this is making Python multilingual.

We are aware of two recent approaches that attempt to tackle this problem: `CodeInternational` (Piech and Abu-El-Haija, 2020), which automatically translates comments, identifiers, and (optionally) string literals in Python/Java code, and UNIPY (Otten et al., 2023), which deterministically translates the natural modality of the code itself, leaving comments/identifiers untouched for consistency. This translation is reversible, and allows the execution of non-English code. Figure 1 outlines the UNIPY framework, and Table 1 details similarities/differences between the two approaches. While `CodeInternational` translates programmer-specified aspects of the code, it does nothing to translate the natural modality of the code itself. It may be useful for tracing and collaboration between people of different languages, but unlike UNIPY, it does not allow someone to write and execute code from scratch that is meaningful to them in their own language. Therefore, we feel that UNIPY presents a superior approach for our endeavor of making Python universal.

However, UNIPY's translations were, in large part, constructed by hand using native-speaking annotators. While this process worked well for a small prototype, Python has an ever-growing set of libraries, with the most recent tally at 137,000 according to Coding Ninjas (kumar, 2023). When this number of packages is taken with the multitude of human languages for which UNIPY could be expanded, hand-annotating Python's terms is simply not scalable. If a truly universal Python is to be established, it would be very helpful to find a way of automating the translation process.

Automating this task is more difficult than it may seem at first glance due to two major factors (i) *conventions for writing identifiers*, and (ii) *ambiguity in mapping technical terms to human languages*. Python's terms largely consist of concatenated abbreviations (i.e., `snake_case` identifiers with abbreviations such as `nan_to_num()`), which can confuse automated translators. Additionally, mathematical or context-specific terms may not map to a single word in another language, complicating the translation process.

Feature/translates	`CodeInter`	UNIPY
Code component		
comments/identifiers	✓	✗
natural modality	✗	✓
Capabilities		
code execution	✓	✓
deterministic	✗	✓
Right-To-Left code	✗	✓

Table 1: Compares/contrasts `CodeInternational` with UNIPY. The functionality of these two projects is fundamentally different; `CodeInternational` translates/transliterates comments/identifiers/strings, while UNIPY translates the natural modality of the code.

Therefore, our work provides three contributions toward reaching the ideal implementation of a "universal Python:"

- First, we introduce the task of *automatically* translating Python's terms into other human languages.
- Then, we create a pipeline for this process after comparing state-of-the-art models/methods.
- Finally, we use this pipeline to expand the current base of Python translations from UNIPY, extending them to five additional common libraries (`pytorch`, `tensorflow`, `pandas`, `numpy`, and `random`) in seven human languages (Spanish, French, Greek, Hindi, Bengali, Mandarin, Arabic), and evaluate its effectiveness on a subset of these terms.

We also consider the effectiveness of fine-tuning an LLM to translate Python code, and provide analysis and discussion of this and our pipeline results.

2 Automated Python Translation

Unlike most translation pipelines, converting Python's keywords and built-in functions/modules to a second language presents a unique challenge. While nearly all of the terms represent a word/phrase in English, most are abbreviated and/or concatenated with other words or abbreviations. Automatic translators can take these forms literally; for instance, Google Translate interprets the `abs` from Python's "absolute value" function as short for "abdominal muscles."

Furthermore, certain questions arise regarding

grammatical function when translating to a language altogether different in structure. For instance, the term "print" in the English language is inherently ambiguous–it could be a verb as in "to print" something, or a noun meaning "the print." If it is a verb, there is ambiguity regarding its nature. Is it a command, an indicative action, or a suggestion? Who is the one printing? Since the English language is not morphologically rich, such information is not marked and the interpretation is left to the reader. However, when translating into a language such as Spanish, where verbal grammar is built in to the word itself, one must understand the nature of what is being translated, knowing the part of speech, tense, mood, person, etc. Human translators of Python might disagree on these implicit grammatical assumptions, which complicates the overall process and could lead to inconsistency in term translation across libraries, etc.

Our intuition is that a pipeline for this task (outlined in Figure 1) should solve several problems:

1. **Python Term Expansion**, that is, unabbreviate and un-concatenate if they are not proper English words, allowing translators to better understand the meaning behind what is being translated.

2. **Python Term Translation** of expansions to a target language in a consistent manner.

3. **Python Term Abbreviation**, where, possibly, one may abbreviate and/or concatenate translations. We implement an (optional) rudimentory abbreviation scheme for our pipeline. For details, see Appendix B.

Each of these tasks should be as automated and human-free as possible, to better allow a large number of Python libraries to be translated into just as many languages. Below, we first explore methods for term expansion and then term translation, before putting together a pipeline comprised of our best approaches to evaluate on new Python libraries for three languages.

3 Python Term Expansion

Many Python terms are abbreviations and/or concatenations of existing English words. Therefore, in order to process them so they can be more accurately translated, we attempt to "expand" these into unabreviated, standard English phrases. As in Figure 1, we provide original Python terms, and the model should output the proper English words or phrases that they represent. For instance, `abs` would be expanded to "absolute" (value), and `delattr` would become "delete attribute."

3.1 Experimental Settings

Data We evaluate expansion of the 222 unique terms from the Python standard library, testing model outputs against the hand-expanded forms from Otten et al. (2023).

Models We use zero, one, and few-shot prompting with GPT-4 Turbo[2] (OpenAI, 2024) to expand Python standard library terms. We also evaluate with a "naive" baseline that does not modify any terms; this simulates what accuracies would result if no expansion had been done in the first place.

Prompts We try four different prompting strategies–zero-shot, zero-shot with motivation, one-shot, and 5-shot–in order to determine which prompting strategy will provide the most reliable output. Following are examples of our prompts.

- `0-shot`: We use the following instruction: "Please expand (i.e. split and unabbreviate) these Python terms into the word or phrase that they are intended to represent. If no abbreviation or splitting into separate words is necessary, then the expanded form will be the same as the original term. Do not provide any other response; simply list each term (each on a separate line) followed by => and its corresponding expansion (as in '[term] => [expansion]'). Here are the terms, separated by commas: "
- `0+Motive`: We augment the previous prompt to provide a motivation for the task. "I am trying to translate Python's key terms into other languages, so that people can code in their native language. However, I first need to know the expanded form of the abbreviations. Please ... [same as above]"
- `1-shot`: Same as above, but now we add one example in the prompt: "[same as above] ... For example: `abs => absolute value`. Please expand these terms: "
- `5-shot`: As above, but instead we provide five examples, as below: "[as above]... For example: `abs => absolute value`, `memoryview => memory view`, `pow => power`, `print => print`, `SyntaxError => Syntax Error`. Please expand these terms: "

[2] gpt-4-0125-preview

We compare this with a "naive" baseline that leaves the input as is (performing no expansion).

3.2 Metrics

We evaluate the above strategies using exact match accuracy and chrF scores (using `sacrebleu` package). Exact match provides a solid base with which to roughly judge how often a model was able to translate/expand the term *exactly* how the humans did. However, we also need to account for slight differences in spelling and grammar, or if a translation was partially correct. Therefore, we also use chrF (Popović, 2015), since that judges character-level similarity for n-grams rather than word similarity, which we need given the short nature of Python terms.

3.3 Results

Our results for expansion are outlined in Table 2. All approaches outperform the naive baseline. We obtain our best score for exact match accuracy (93.2%) in the `5-shot` setting. ChrF scores tend to be quite close to each other, and we feel that the differences here are negligible. Therefore, it seems reasonable to conclude that `5-shot` is currently the best setting for ChatGPT.

The fact that few-shot seems better than a 1-shot prompt suggests something significant – this expansion task is novel. The idea that LLMs need little instruction-tuning to perform a task they are pre-trained for is supported by findings from Min et al. (2022), Xie et al. (2022), and Ratner et al. (2023). Therefore, these results suggest that Python term expansion may be a completely new task, and one that could be improved by showing examples of this during pre-training.

Regardless, accuracies this high are very promising that GPT expansion will be helpful in our translation pipeline. Since the baseline scores are also relatively high (84.9% for chrF), these results do not necessarily speak to GPT-4's overall ability; they do however demonstrate that for our task of Python term expansion, using an instruction-tuned LLM such as ChatGPT may be a reasonable approach.

4 Python Term Translation

In order to determine the best translation strategy for the second stage of our pipeline, we test and evaluate three primary models with different strategies: Google Translate, ChatGPT-4 Turbo, and Llama2. We evaluate using the same metrics as in the expansion experiments (§3.2): exact match accuracy and chrF score.

Prompt	Accuracy	chrF
naive baseline	46.9	84.9
`0-shot`	89.6	**96.1**
`0+Motive`	92.3	95.4
`1-shot`	91.0	95.4
`5-shot`	**93.2**	**95.7**

Table 2: Expansion accuracy of Python's standard library using ChatGPT-4 Turbo on four prompts, showing both raw and chrF scores. Base represents the baseline of original (unmodified) Python terms. In this case, `5-shot` (5-shot) clearly performs with the highest accuracy, suggesting that more context may be beneficial.

4.1 Experimental Settings

Data For each system/method, we evaluate translations of the expanded form of the 222 unique Python standard library keywords and built-in functions/modules, acquired from Otten et al. (2023). Our references include eight languages in all: Spanish, French, Greek, Mandarin, Hindi, Begnali, Sorani Kurdish, and Arabic.[3]

Models For our models, we opted to use Google Translate (as a baseline machine translation system), GPT-4 Turbo, and Llama2 with 70 billion parameters. We also provide translation scores from ChatGPT-3 Turbo and Davinci models OpenAI (2023) in Appendix A, showing the extent to which version might affect in ChatGPT's abilities.

Prompts We craft prompts for the two LLMs using similar strategies to §3.1. As before, we have `0-shot`, `1-shot`, and `5-shot` prompts. We also consider that it might help a model to see preexisting translations in another language (to better understand the task), so we include an additional prompt (referred to here as `all-other`) with an entire set of the 222 terms. For consistency, we chose Spanish as this reference for all languages except itself, in which case we use French examples. All complete prompts are listed in Appendix E.

As a translation model and not a LLM, these types of prompting strategies are not applicable to Google Translate. However, we can emulate

[3]While the Arabic translations were vetted to be used in prototypes, several translations (around 17) were marked as not confident before we conducted these experiments; we mitigate this by leaving out these terms from our prompts.

this methodology by providing three levels of context in the source sentence (the input to be translated), which we refer to as `no-cntxt`, `def`, and `expl`. Our first level of context (`no-cntxt`) is similar to a zero-shot prompt–we simply translate the expanded forms of the terms. The second level (`def`) provides additional context by providing the term and a Python definition for it, and the final contextual level (`expl`) provides a sentence explanation of the term in question, followed by a colon and the term itself, as in the following examples:

- `no-cntxt`: "print"
- `def`: "print: Prints to the standard output device"
- `expl`: "In Python, to use the expression that prints to the standard output device, write: print."

4.2 Results

Our translation results are in Table 3. We evaluate using exact match accuracy (raw) and chrF scores. Note that chrF scores rarely match corresponding raw accuracies perfectly. When a chrF score is a bit lower than the raw accuracy, we can assume that non-matching translations were simply wrong. On the other hand, a higher chrF score indicates that, even when translations were wrong, some portions were indeed correct.

Additionally, the best raw score per prompt is not always the same as the highest chrF score. To determine which (if any) prompting strategy is optimal, we perform several ANOVA tests on these results (for tables, see App D). However, in all these cases, no prompting strategy can be said to be better than another at a 90% level of confidence.

ChatGPT ChatGPT-4 demonstrates noteworthy ability in term translation, especially chrF. Table 3 shows relatively good scores for Spanish and French, nearly in the 70s range, and hovering around 80 for chrF. With Greek and Mandarin the scores begin to drop, and then they continue down to as low as 29% raw accuracy. We found these differences in language scores to be statistically significant for both Raw and chrF accuracy at 99% confidence, using single Factor ANOVA tests according to scores averaged across languages.[4] This analysis suggests that ChatGPT can be fairly trusted for high resource languages, which resonates with other multilingual LLM findings Hendy et al. (2023). However, as the level of resources for a language drops, so does its accuracy. As far as which of the five prompting strategies gives us the best performance, we used ANOVA testing for the five prompting "alternatives" (see Appendix Table 10) and found that there are no statistically significant differences among them; our prompting strategies did not have enough of an effect on ChatGPT's performance to suggest that one is better than another.

Llama2 Overall, Llama2 appears to perform even worse than ChatGPT; Table 3 shows that it is outperformed for almost every data point. Though we again see statistical significance with respect to scores across languages (see App. Table 15), our ANOVA tests revealed no significant differences based on prompting strategy.

Google Translate Google Translate without additional context (`no-cntxt`) yields the best results, outperforming ChatGPT in most of the cases. It also generally appears to have a more even distribution with regard to scores across languages, although ANOVA tests reveal statistical significance in the scores across languages at 90% confidence.

Regarding the effect of context in translation, Table 3 shows a clear drop-off from `no-cntxt` to `def`, and then `def` to `expl`, where `expl` (containing the most context) is absolutely terrible. Using ANOVA and the Method of Contrasts, we found the differences between the versions to be statistically significant with 99% confidence, both for raw and chrF scores (see App. Table 16).

Upon analysis of the translations, we find that in the majority of cases, `expl` did not even attempt to translate the Python term, instead translating everything but the terms in question. This indicates that Google Translate was perhaps given too much context–the longer sentences reveal that the terms are indeed Pythonic and therefore were not translated, which is reasonable given that these are indeed typically not translated across languages. In other words, since Python's key terms are historically only English, the translation model reasonably opted to keep them untranslated so that the initial context could be maintained.

Difficulties with Parsing and Formatting Responses from ChatGPT and Llama2 sometimes proved difficult to parse and/or format. For instance, they often provided terms in mixed scripts, or simply restated the Python terms to translate. Even after receiving formatting specifica-

[4] These results can be found in Appendix Table 11.

Model	Prompt	Spanish		French		Greek		Mandarin		Hindi		Bengali		Arabic		Kurdish	
		Raw	chrF	Raw	chrF	Raw	chrF	Raw	chrF	Raw	chrF	Raw	chrF	Raw	chrF	Raw	chrF
GPT-4	`0-shot`	**71.6**	83.4	**62.6**	78.0	43.2	64.3	54.0	69.9	13.5	23.8	34.2	56.6	26.6	52.1	31.1	50.6
	`0+Motive`	68.9	*85.2*	60.8	77.5	41.0	63.9	**56.3**	*71.0*	28.8	49.8	31.1	55.1	27.5	*54.2*	**35.6**	*54.4*
	`1-shot`	71.2	83.8	61.7	78.3	**45.1**	*69.1*	50.5	67.5	29.7	51.0	32.0	55.0	**29.7**	54.1	30.6	52.0
	`5-shot`	67.6	81.7	61.3	*78.7*	44.1	66.6	27.5	37.9	**35.6**	*55.1*	**36.9**	*57.6*	13.5	27.9	34.2	54.3
	`all-other`	70.3	83.6	61.2	78.3	44.1	66.3	51.8	69.2	29.3	50.6	28.4	56.2	25.7	51.6	28.4	50.7
Llama2	`0-shot`	**66.2**	*81.5*	49.6	68.9	12.6	34.8	27.0	42.2	5.9	19.1	**5.0**	*17.3*	12.6	30.8	1.8	9.2
	`0+Motive`	59.0	76.5	49.6	69.7	13.5	*35.8*	25.2	41.1	6.3	20.9	4.5	16.3	9.5	28.1	1.8	9.8
	`1-shot`	59.0	75.2	46.4	66.9	13.1	34.2	26.6	44.4	7.2	21.4	**5.0**	15.9	**14.0**	29.9	2.3	10.3
	`5-shot`	57.2	76.7	48.2	68.4	**16.2**	35.3	**27.9**	*45.7*	**10.8**	*27.1*	2.3	13.4	12.2	*31.0*	**2.7**	*11.4*
	`all-other`	59.0	78.8	**50.0**	*71.2*	15.3	34.4	24.3	41.4	6.8	21.6	3.2	13.1	11.3	30.3	1.8	10.0
Google	`no-cntxt`	**73.4**	*82.4*	**84.2**	*88.8*	**45.5**	*64.3*	**80.6**	*86.4*	**39.2**	*56.6*	**67.6**	*80.5*	**61.3**	*73.5*	**98.2**	*98.6*
	`def`	55.9	73.1	43.2	61.1	32.9	46.1	25.2	29.3	19.4	32.1	18.5	27.5	23.0	51.4	23.9	27.4
	`expl`	0.5	66.7	16.2	39.1	27.0	37.1	1.4	3.2	30.2	45.9	32.4	47.3	23.4	47.0	23.4	32.3

Table 3: Python translation quality: exact match (raw) accuracy and chrF score, for each prompt (`0-shot`, `0+Motive`, etc.) or contextual level (`no-contxt`, `def`, `expl`), for all translation models. The best raw scores per prompt are **bolded**, while best chrF scores are *italicized*. Google Translate without context is consistently the best among models across all languages, often by large margins.

tions, they would occasionally neglect to translate or expand a particular term, and omit it in the list of outputs. In these cases we simply used "-" in place of a translation. Interestingly, certain terms were more frequently omitted than others; for example, Llama2 often neglected to translate `as`.

5 Downstream Pipeline Evaluation

After evaluating various models' ability to expand and translate the Python standard library, we now try our best pipeline in the wild, on terms from five *additional* Python libraries–`Pandas`, `Pytorch`, `TensorFlow`, `Numpy`, and `Random`–in seven languages (es, fr, el, hi, bn, ar, zh-cn). We manually extract terms from online documentation, and our resulting translation set comprises 6,119 terms. These can be made to work with UNIPY simply by augmenting its dictionary lists with these terms.

We evaluate a subset of our outputs from these libraries in Greek, Bengali, and French. We use native speakers to hand-annotate results (i.e. correct pipeline outputs). Since abbreviating terms is optional in our pipeline, we only evaluate translations for these experiments.

5.1 Setup

We combine our best techniques from the expansion and translation tasks to create an optimal pipeline. GPT-4 was able to expand Python terms with impressive accuracy, but due to cost constraints we use GPT-3.5 Turbo for this expansion[5] (OpenAI, 2023). While not as impressive as GPT-4, GPT-3 Turbo can still perform with significant accuracy in the few-shot setting (for numbers on this, see Appendix A), so we believe this will work as a first attempt. As for term translation, we observe that `no-cntxt` of Google Translate had the best performance of all models. Therefore, our pipeline has the following steps:

1. Expand the terms using ChatGPT-3 Turbo, prompting with a similar scheme as the 5-shot example in §3.1.
2. Translate the processed form of the terms using Google Translate `no-cntxt` (no additional context). We also do some minor post-processing (replacing spaces with underscores and removing determiners).

We test this for Greek, Bengali and French, on four common Python libraries, `tensorflow`, `pandas`, `pytorch`, `random`, and `numpy`,[6] translating a total of 6,119 terms. We hand-annotate (correct) a subset of the outputs (407) by asking if each given translation could be considered reasonable, and if not, correcting to something that is. Note that this process matches the envisioned scenario of language communities contributing to correct and solidify the automatically produced outputs. We evaluate with raw and chrF scores.

5.2 Results

The scores for our pipeline translations (Table 4) vary significantly, but are overall quite good considering the novelty of this task. For instance, chrF scores remain above 60% for all languages on av-

[5] gpt-3.5-turbo-1106

[6] Note that since these libraries are extensive, we only test with a subset of terms.

erage, and Bengali achieves over 90% in over half the cases. This demonstrates an important find: our pipeline can already do a fairly good job at initial translations of Python terms from scratch. Without improvements, this method should significantly cut the time needed for manual annotation. Also of note are the two differing ways Bengali-speakers express mathematical concepts: transliterated English terms, and Bengali words. While we attempted translation here for more thorough analysis, transliteration should be straightforward to implement as well. Ideally, future work would create Python versions for both so that speakers could use their preferred expressions.

Some examples where translations failed include improper/unhelpful phrasing and words used in the wrong context. For instance in French, certain phrases such as "argument partition" were translated as "partition d'argument" but corrected to "argument partition." None of the important vocabulary changed, but the phrasing was improved. For words in an improper context, "less than or equal" translated to "moins ou égal" but was corrected to "inférieur ou égal." Both "moins" and "inférieur" have similar translations, but a different context here that affects the overall meaning. We also include some examples from Hindi. While we did not have resources to do a comprehensive evaluation of this language, we analyzed some outputs and found many mistranslations. There were phrases that translated with inappropriate context from the English side; for instance "uniform" translated to the clothing rather than the distribution, "keys" translated to the tool rather than for "key/value pair," and "character" translated to the persona in a story/play, rather than an alphanumeric representation. Since the ambiguity inherently arises from the English, we suspect that other languages may have this issue as well.

Finally, occasionally ChatGPT would expand inappropriately, such as `set_tooltips` expanding to Spanish "establecer consejos." Sometimes ChatGPT's expansions can be extremely long, as in `random.rand` expanding to "random data or random values generated with uniform distribution." This, while not inaccurate, is far too long to be used as a Python term. This case would benefit from our abbreviation scheme.

6 Code-Block Translation Model

In addition to our pipeline experiments, we consider the effectiveness of finetuning an LLM to translate entire blocks of Python code. We extract code samples from `codeparrot/github-code` and translate into four languages (Spanish, French, Greek, and Hindi) using UniPy. To ensure that all appearing terms are supported by either UniPy or our own translations, we filter by import statements. We also filter prompts by a character length of 500 for efficiency. Our resulting training set comprises 32,528 examples, where translations can be in either the English $\rightarrow$ non-English or non-English $\rightarrow$ English direction. We LoRA finetune the `Llama-2-7b-chat-hf` model for 15 epochs, and test on an additional set of 13,165 code examples, evaluating with BLEU (Papineni et al., 2002) and chrF score (both with `sacrebleu`). Results are in Table 7 in Appendix C. Overall, we receive positive scores, demonstrating that an LLM can successfully translate Python code blocks. However, the model over-generated in many cases, writing additional translated code, etc. Given these clear imperfections and non-perfect scores, we cannot expect translations to be reliably executable. However, this method may be useful for code accessibility (e.g. multi-lingual documentation), or perhaps extracting translations of terms that may then be included in the UniPy tables.

7 Discussion

All in all, our initial automated translation pipeline requires successful ability to perform two tasks: expansion of the original Python terms, and translation into a target language. Fortunately, we find relatively high accuracies for both, using GPT-4's expansions under a few-shot prompt, and Google Translate `no-cntxt` for the translations.

Asking ChatGPT to expand Python terms was met with positive results. It appears that expanding Python terms may not be present in ChatGPT's training data (especially GPT-3, see Appendix A), suggesting that our task is completely new. We would likely find even better results by including more examples in the prompts, or if ChatGPT were pre-trained for expansion in the future. Google Translate (without context) achieved the highest translation accuracy and consistency across languages, demonstrating its superiority for this task. This may be partly due to issues with the pre-trained generative models failing to ad-

	PyTorch %		TensorFlow %		Pandas %		Random %		NumPy %		Total %	
# terms	80		80		80		22		145		407	
metric	raw	chrF	raw	chrF	raw	chrF	raw	chrF	raw	chrF	raw	chrF
French	50.0	75.2	41.3	71.2	61.3	75.5	31.8	65.8	52.4	71.6	**50.1**	**72.7**
Greek	32.5	63.7	38.8	64.3	33.8	59.9	27.3	51.2	46.2	66.1	**38.6**	**63.2**
Bengali	98.8	99.7	82.5	90.9	96.3	97.7	72.7	73.5	53.8	67.8	**82.1**	**90.8**

Table 4: Results from the Pipeline experiment, translating 407 terms from five Python libraries into different languages. The Total scores represent accuracies over a combined set of terms from the packages.

equately follow instructions, paired with inadequate overall performance on lower-resourced languages such as Sorani Kurdish. We test this pipeline in the wild for the `pytorch`, `pandas`, `tensorflow`, `numpy`, and `random` libraries, obtaining scores mostly over 50%, in some cases extremely high. Thus, we can expect decent initial translations from our pipeline, especially for higher-resourced languages. Overall, our pipeline provides a fast method of translating terms. We also try fine-tuning Llama2 to translate Python code blocks, and are met with positive results. However, these are not currently sufficient for our task of creating executable code.

In the practical setting, our pipeline could already be used to obtain preliminary translations for the rest of Python's multitude of libraries into other human languages. For languages with good performance, prototype versions could be developed and used out of the box, without necessarily requiring immediate annotation. Then, these initial translations could be made open-source and updated by native-speaking annotators all over the world for higher-quality versions. Once we have translations, all we need to do to integrate this into UniPy is update the mapping tables.

Future work should find or create even better methods for Python expansion and translation, to further alleviate the burden currently placed on annotators. One experiment might improve GPT prompting to include translated code segments, providing more context to better translate Python terms. Additionally, while our pipeline includes an initial abbreviation scheme, it would be helpful to find language-specific methods of abbreviation to make terms more meaningful to programmers.

8 Related Work

While a Universal Python is still early in development, several instances of programming languages were developed for users of particular linguistic backgrounds. For instance, Scratch and Blockly (used in the educational space) support certain non-English languages, and 'Ku-Mir'[7] and 'Glossa'[8] are Pascal-based programming languages using key terms in Russian and Greek, respectively Mcculloch (2019). Piech and Abu-El-Haija (2020) analyzed the extent of multilinguality on GitHub, and created a tool, "`CodeInternational`," to automatically translate identifiers defined in a Java or Python codebase (such as function names), comments, and optionally, string literals, to other languages using Google Translate. While this approach can certainly be helpful, it does not translate the modality of the code itself, falling short of creating a "universal" Python. Otten et al. (2023) began the process of manually translating Python's standard library into eight other human languages. We use these translations as references in our experiments.

9 Conclusion

Python translation is a necessary task, and a pipeline is essential for any large-scale translation efforts. We present the first-ever pipeline to do this and obtain reasonable results. This paper introduces the task of automatically translating Python terms, building a pipeline consisting of three main steps: expansion, translation, and abbreviation.

We use our best pipeline to translate four additional Python libraries, contributing over 6,000 new terms to the current base of Python translations in seven languages. We perform a quality test on 407 of these translations for Greek, French, and Bengali, obtaining positive initial results with room for improvement. Although automated translation of Python is nowhere near perfect, we can begin the process of translating libraries for high-resourced languages and expect positive initial results. This is an important step toward universal programming, where everyone from any culture can code in their native language.

[7] https://web.archive.org/web/20160112180533/http://lpm.org.ru/kumir2/

[8] https://web.archive.org/web/20160112180533/http://lpm.org.ru/kumir2/

Limitations

It is worth noting that human annotators may disagree as to what constitutes a reasonable translation; this may be a factor in certain model scores, especially across languages where annotators change. For our work, we were only able to have one annotator per library.

Ethics Statement

Using ChatGPT and Llama2's outputs may carry with it certain privacy concerns over where the training data for these LLMs came from, and how it was used in text generation, which is out of scope for this work.

References

Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation. *arXiv preprint arXiv:2302.09210*.

Benjamin Hill. 2017. Learning to code in one's own language.

Anna Johnson. 2023. Python popularity: The rise of a language.

Aditya kumar. 2023. Python libraries.

Gretchen Mcculloch. 2019. Coding is for everyoneas long as you speak english.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work?

OpenAI. 2023. Gpt-3.5: A large language model. *OpenAI Technical Report*.

OpenAI. 2024. Gpt-4 turbo and gpt-4.

Joshua Otten, Antonios Anastasopoulos, and Kevin Moran. 2023. Towards a universal python: Translating the natural modality of python into other human languages.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Chris Piech and Sami Abu-El-Haija. 2020. Human languages in source code: Auto-translation for localized instruction. In *Proceedings of the Seventh ACM Conference on Learning@ Scale*, pages 167–174.

Maja Popović. 2015. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the tenth workshop on statistical machine translation*, pages 392–395.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. Parallel context windows for large language models.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2022. An explanation of in-context learning as implicit bayesian inference.

A Comparisons between LLM performance

Table 5 provides translation accuracies for GPT-3 Turbo and GPT-3 Davinci, and Figures 2 and 3 show comparisons between some of the LLM performance for each prompt, averaged across all languages. It is worth noting that ChatGPT and Llama2's results appear somewhat worse in the graphs than they are in reality; this is due to the lower-resource languages deflating the overall averages (even though some languages, such as Spanish and French, achieved relatively high accuracy).

For expansion accuracies, Figure 4 provides a comparison of the results for each prompt. Note that terms were only expanded in English.

We additionally include expansion accuracies of GPT-3 Turbo in Table 6, since we use it in our pipeline.

B Abbreviation Scheme

In an attempt to follow general Python conventions, we abbreviate according to syllable structure. For a given word, we first separate into syllables, where each consists of either a [vowel] or [set of consonants]+[vowel]. Then, we abbreviate by keeping the first two syllables plus one additional consonant, discarding the rest of the word. If there is a collision, we iteratively add back letters until it is a unique term again.

If the term is multi-word (e.g. separated by underscores), we first eliminate unnecessary articles and coordinating conjunction words, and then abbreviate each individual word according to the above process.

We believe this works as an initial attempt to ensure that newly translated terms are short enough to be reasonably used in Python code for many languages. However, our current approach would

Model	Prompt	Spanish		French		Greek		Mandarin		Hindi		Bengali		Arabic		Kurdish	
		Raw	chrF	Raw	chrF	Raw	chrF	Raw	chrF	Raw	chrF	Raw	chrF	Raw	chrF	Raw	chrF
TURBO	0-shot	70.3	83.9	**64.0**	78.3	41.4	64.1	52.7	68.1	27.5	46.7	27.5	47.7	25.7	51.8	10.4	29.1
	0+Motive	71.2	*85.2*	63.5	78.1	41.0	64.7	50.5	67.5	27.0	46.6	26.6	45.1	25.7	51.3	11.7	27.1
	1-shot	**73.0**	84.7	57.7	76.5	**44.1**	66.3	**54.5**	*69.1*	25.2	48.3	**32.0**	*50.4*	**27.9**	*52.7*	**14.4**	*32.1*
	5-shot	72.5	82.0	64.4	*79.6*	43.7	66.3	50.5	67.8	**28.8**	*50.5*	31.5	50.3	26.6	52.4	13.5	31.6
	all-other	55.4	80.7	60.4	77.4	44.1	*67.4*	47.3	68.8	26.6	47.9	25.7	48.9	23.4	49.0	8.1	21.8
DAVI	0-shot	66.2	81.5	53.6	75.3	36.0	57.2	38.7	53.3	26.1	48.5	7.7	18.1	22.5	49.9	3.6	17.8
	0+Motive	64.9	*82.2*	55.4	75.1	36.5	59.3	46.0	61.1	22.1	43.0	**28.4**	*46.3*	22.1	*50.1*	8.1	20.4
	1-shot	64.0	79.0	58.6	76.6	**39.6**	*62.5*	**53.6**	66.3	**27.5**	*49.6*	23.0	41.8	**25.2**	48.8	9.9	26.2
	5-shot	**68.0**	82.0	56.3	74.9	35.4	60.0	47.3	61.8	**27.5**	48.4	25.7	46.2	24.3	50.0	**14.4**	*29.5*
	all-other	65.8	81.4	**62.2**	*78.8*	35.1	58.4	50.9	*66.0*	27.0	48.5	23.9	43.1	22.1	48.5	5.9	17.5

Table 5: Translation results of the 222 standard library terms with GPT-3 Turbo and GPT-3 Davinci. GPT-3 Turbo outperforms the Davinci model, but in general is not as good as GPT-4 Turbo.

Prompt	Accuracy	chrF
naive baseline	46.9	84.9
0-shot	70.7	82.8
0+Motive	63.5	75.7
1-shot	53.2	73.0
5-shot	**75.7**	**86.5**

Table 6: Expansion accuracy of Python's standard library using ChatGPT-3 Turbo on four prompts, showing both raw and chrF scores. Base represents the baseline of original (unmodified) Python terms. In this case, 5-shot (5-shot) clearly performs with the highest accuracy, suggesting that more context may be beneficial.

not work with all languages (such as Mandarin) whose writing systems do not allow segmentation into syllabic structure. Additionally, languages may have differing conventions for abbreviation (or even none at all), in which case it will be necessary to develop more nuanced techniques to handle the abbreviation task.

C Finetuned Model Scores

We provide a table of our finetuning results for translation of entire code blocks (Table 7). The scores demonstrate reasonable performance and suggest that this method has potential–however, considering the precise nature of computer programming, this is unlikely to be good enough for cases requiring execution of the translated code.

D ANOVA Tests

In general for the LLMs, we find the variation of scores to be statistically insignificant across prompts, but significant across languages. This suggests that while our prompts did not have very much effect on the output, we can expect LLMs to perform much better on high resource languages than lower-resourced ones.

Google Translate At a 90% confidence interval, neither the chrF or raw scores are statistically significant (see Tables 16 and 17). It should also be noted that at 99%, language selection for chrF ceases to be relevant.

The raw accuracy for `def` and `expl` was significant at 95% confidence intervals, but we cannot make this claim at the 99% level. On the other hand, the chrF scores for these were very similar, such that the differences were not found to be significant at even a 90% confidence interval.

E Complete Prompts

We list the complete prompts here for expansion and translation.

Expansion

- **`0-shot:`** "Please expand (i.e. split and unabbreviate) these Python terms into the word or phrase that they are intended to represent. If no abbreviation or splitting into separate words is necessary, then the expanded form will be the same as the original term. Do not provide any other response; simply list each term (each on a separate line) followed by => and its corresonding expansion (as in '[term] => [expansion]'). Here are the terms, separated by commas: "
- **`0+Motive:`** "I am trying to translate Python's key terms into other languages, so that people can code in their native language. However, I first need to know the expanded form of the abbreviations. Please help me with this

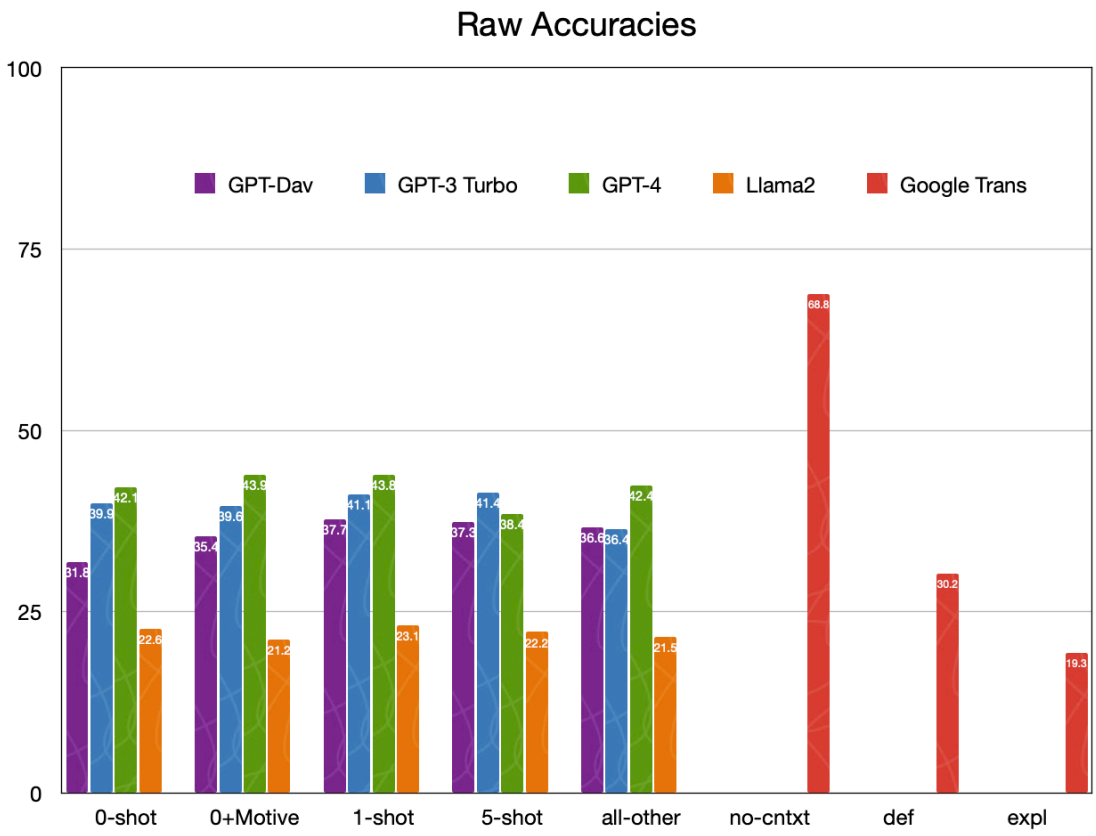

Figure 2: Plot of the raw accuracy percentage averaged over the languages, for each prompt/model.

Metric	En→Es	Es→En	En→Fr	Fr→En	En→El	El→En	En→Hi	Hi→En	Avg
BLEU	38.7	32.6	38.8	32.6	39.8	33.0	39.7	32.8	**36.0**
chrF	66.3	59.4	66.3	59.7	62.9	59.3	64.1	59.6	**62.2**

Table 7: BLEU and chrF scores of Llama2 finetuned on translating code blocks. There appears to be little variation across languages; however interestingly we see that the English → non-English directionality performs better than the other way around.

by expanding (i.e. splitting and unabbreviating) each of the following terms into the word or phrase that they are intended to represent. If no abbreviation or splitting into separate words is necessary, then the expanded form will be the same as the original term. Do not provide any other response or translations; simply list each term (each on a separate line) followed by => and its corresonding expansion (as in '[term] => [expansion]'). Here are the terms, separated by commas: "

- **`1-shot:`** "I am trying to translate Python's key terms into other languages, so that people can code in their native language. However, I first need to know the expanded form of the abbreviations. Please help me with this by expanding (i.e. splitting and unabbreviating) each of the following terms into the word or phrase that they are intended to represent. If no abbreviation or splitting into separate words is necessary, then the expanded form will be the same as the original term. Do not provide any other response or translations; simply list each term (each on a separate line) followed by => and its corresonding expansion (as in '[term] => [expansion]'). For example: abs => absolute value. Please expand these terms: "

- **`5-shot:`** "I am trying to translate Python's key terms into other languages, so that people can code in their native language. However, I first need to know the expanded form

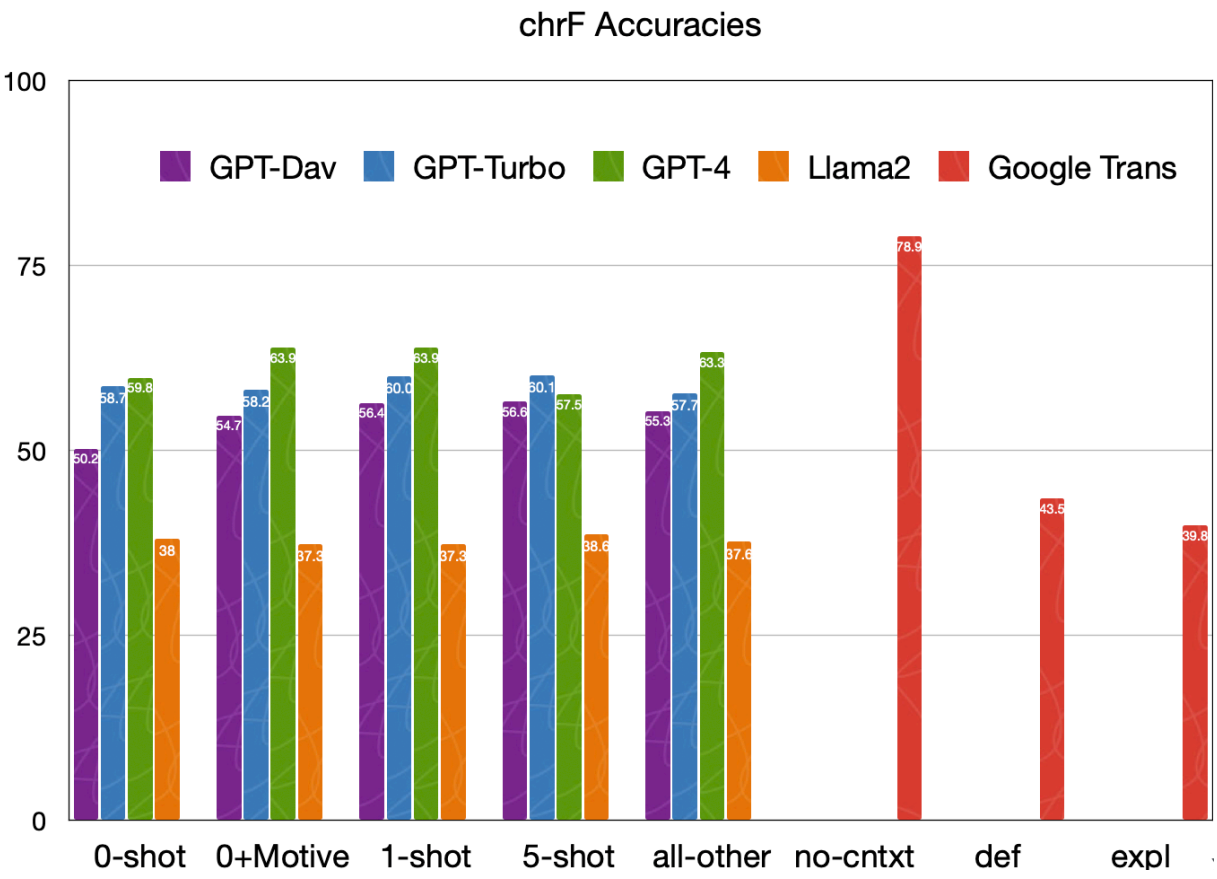

Figure 3: Plot of the chrF accuracy percentage averaged over the languages, for each prompt/model.

Variation	SS	df	MS	F	P-value	F crit
SSA	76.180	4	19.045	0.063	0.992	2.113
SSE	10521.084	35	300.602			
Total	10597.263	39				
SSA	266.589	4	66.647	0.280	0.889	2.113
SSE	8319.351	35	237.696			
Total	8585.94	39				

Table 8: Results of ANOVA test on ChatGPT-4 Turbo's five prompting strategies, at a 90% confidence interval. The first set of rows indicates analysis for raw scores, while the second is chrF. All values are rounded to 3 decimal places, for this and other ANOVA tables. Since the P-value is greater than $\alpha = 0.1$, we can conclude that variation due to the prompts is not statistically significant at this level of confidence.

Variation	SS	df	MS	F	P-value	F crit
SSA	9522.082	7	1360.297	40.486	3.91E-14	3.258
SSE	1075.181	32	33.599			
Total	10597.26	39				
SSA	6597.024	7	942.432	15.163	1.48E-08	3.258
SSE	1988.916	32	62.154			
Total	8585.94	39				

Table 9: Results of ANOVA test on ChatGPT-4 Turbo's language scores, at a 99% confidence interval. The first set of rows indicates analysis for the raw scores, while the second is for chrF.

of the abbreviations. Please help me with this by expanding (i.e. splitting and unabbreviating) each of the following terms into the word or phrase that they are intended to represent. If no abbreviation or splitting into separate words is necessary, then the expanded form will be the same as the original term. Do not provide any other response or translations; simply list each term (each on a separate line) followed by => and its corresonding expansion (as in '[term] => [expansion]'). For example: abs => absolute value
memoryview => memory view
pow => power
print => print
SyntaxError => Syntax Error. Please expand these terms: "

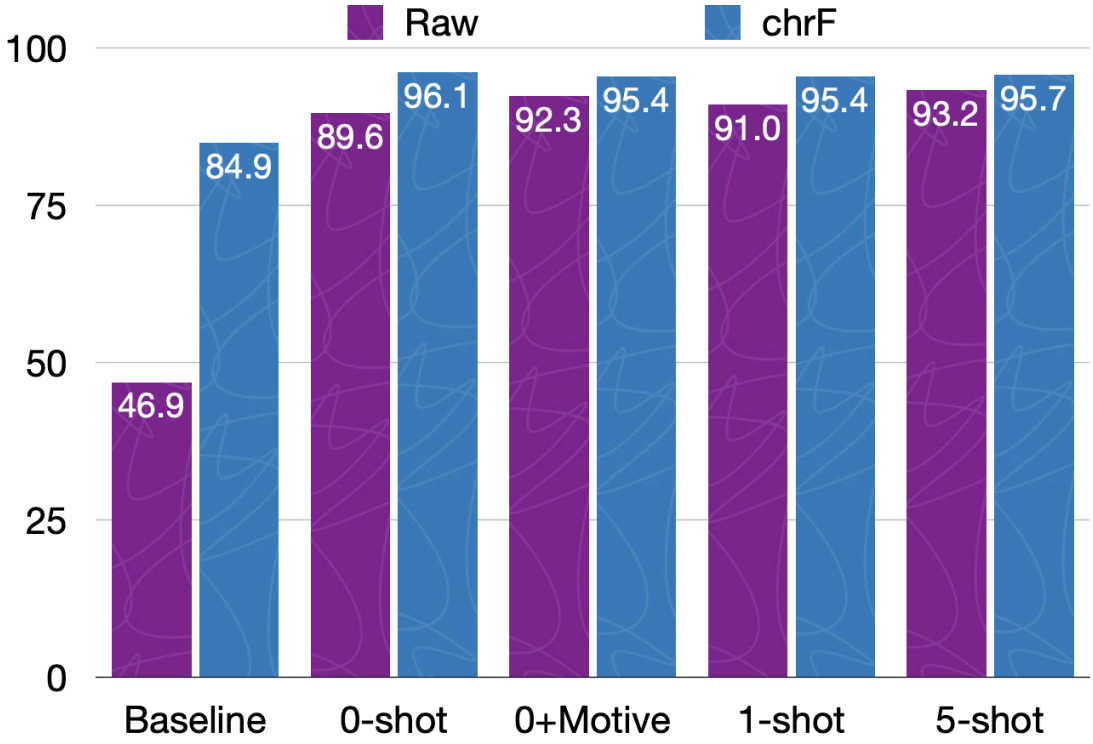

Figure 4: Plot of GPT-4's expansion accuracy percentage (Raw and chrF) for each prompt/model.

Variation	SS	df	MS	F	P-value	F crit
SSA	141.796	4	35.449	0.102	0.981	2.091
SSE	13945.827	40	348.646			
Total	14087.623	44				
SSA	10.32	4	2.58	0.004	0.999	2.113
SSE	21051.18	35	601.462			
Total	21061.5	39				

Table 10: Results of ANOVA test on ChatGPT-3 Turbo's five prompting strategies, at a 90% confidence interval. The first set of rows indicates analysis for raw scores, while the second is chrF.

Variation	SS	df	MS	F	P-value	F crit
SSA	13708.88	7	1958.412	172.650	1.44E-23	3.258
SSE	362.985 32	11.343				
Total	14071.87	39				
SSA	20951.4	7	2993.057	869.917	1.22E-34	3.258
SSE	110.1	32	3.441			
Total	21061.5	39				

Table 11: Results of ANOVA test on ChatGPT-3 Turbo's language scores, at a 99% confidence interval. The first set of rows indicates analysis for the raw scores, while the second is for chrF.

Translation

- **`0-shot:`** "Please translate the following terms into [language]. Do not provide any other response or translations; simply list each term (each on a separate line) followed by => and its corresonding translation (as in '[term] => [translation]'). Here are the terms, separated by commas: "

- **`0+Motive:`** "I am trying to translate Python's key terms into other languages, so that people can code in [language]. Please help me with this by translating each of the following terms into [language]. Do not provide any other response or translations; simply list each term (each on a separate line) followed by => and its corresonding translation (as in '[term] => [translation]'). Here are the terms, separated by commas: "

- **`1-shot:`** "I am trying to translate Python's key terms into [language], so that people can code in [language]. Do not provide any other response or translations; simply list each term (each on a separate line) followed by => and its corresonding translation (as in '[term] => [translation]'). For example: absolute value => [translation]. Please translate these terms into [language]: "

Variation	SS	df	MS	F	P-value	F crit
SSA	179.471	4	44.868	0.113	0.977	2.113
SSE	13857.014	35	395.915			
Total	14036.485	39				
SSA	215.066	4	53.767	0.138	0.967	2.113
SSE	13674.02	35	390.686			
Total	13889.09	39				

Table 12: Results of ANOVA test on ChatGPT Davinci's five prompting strategies, at a 90% confidence interval. The first set of rows indicates analysis for raw scores, while the second is chrF.

Variation	SS	df	MS	F	P-value	F crit
SSA	13486.926	7	1926.704	112.189	1.107E-20	3.258
SSE	549.558	32	17.174			
Total	14036.485	39				
SSA	13035.34	7	1862.192	69.798	1.43E-17	3.258
SSE	853.748	32	26.680			
Total	13889.09	39				

Table 13: Results of ANOVA test on ChatGPT Davinci's language scores, at a 99% confidence interval. The first set of rows indicates analysis for the raw scores, while the second is for chrF.

- **`5-shot:`** "I am trying to translate Python's key terms into [language], so that people can code in [language]. Do not provide any other response or translations; simply list each term (each on a separate line) followed by => and its corresonding translation (as in '[term] => [translation]'). For example: absolute value => [translation]
  memory view => [translation]
  power => [translation]
  print => [translation]
  Syntax Error => [translation]. Please translate these terms into [language]: "

- **`all-other:`** "I am trying to translate Python's key terms into [language], so that people can code in [language]. For example, when translating Python to French, you have these translations: [set of English => French terms, separated by commas]. Please translate the following terms into [language]. Do not provide any other response or translations; simply list each term (each on a separate line) followed by => and its corresonding translation (as in '[term] => [translation]'). Here are the terms, separated by commas: "

Variation	SS	df	MS	F	P-value	F crit
SSA	10.422	4	2.605	0.006	0.999	2.113
SSE	16534.470	35	472.413			
Total	16544.892	39				
SSA	10.32	4	2.58	0.00429	0.999	2.113
SSE	21051.18	35	601.462			
Total	21061.5	39				

Table 14: Results of ANOVA test on Llama2's five prompting strategies at 90% confidence. Raw on top, chrF on bottom.

Variation	SS	df	MS	F	P-value	F crit
SSA	12613.351	7	1801.907	817.728	5.848E-27	3.496
SSE	52.885	24	2.204			
Total	12666.236	31				
SSA	20951.4	7	2993.053	869.917	1.22E-34	3.258
SSE	110.1	32	3.441			
Total	21061.5	39				

Table 15: ANOVA test on Llama2's language scores, at a 99% confidence interval. The first set of rows indicates analysis for raw scores, while the second is for chrF.

Variation	SS	df	MS	F	P-value	F crit
SSA	10799.504	2	5399.752	22.559	5.889E-06	5.780
SSE	5026.688	21	239.366			
Total	15826.192	23				
SSA	7444.426	2	3722.213	13.783	1.5E-04	5.780
SSE	5671.164	21	270.055			
Total	13115.59	23				

Table 16: Results of ANOVA test on Google Translate's three contextual version strategies, at 99% confidence. Top rows are raw, bottom are chrF.

Variation	SS	df	MS	F	P-value	F crit
SSA	922.228	7	131.747	0.141	0.993	2.128
SSE	14903.964	16	931.498			
Total	15826.192	23				
SSA	2442.63	7	348.947	0.523	0.804	2.128
SSE	10672.96	16	667.06			
Total	13115.59	23				

Table 17: ANOVA test on Google Translate's language scores, at a 90% confidence interval. The first set of rows indicates analysis for raw scores, while the second is for chrF.