MCoNaLa: A Benchmark for Code Generation from Multiple Natural Languages

Zhiruo Wang[*] **Grace Cuenca**[*] **Shuyan Zhou** **Frank F. Xu** **Graham Neubig**
Carnegie Mellon University Princeton University Inspired Cognition
{zhiruow,shuyanzh,fangzhex,gneubig}@cs.cmu.edu, gcuenca@princeton.edu

Abstract

While there has been a recent burgeoning of applications at the intersection of natural and programming languages, such as code generation and code summarization, these applications are usually English-centric. This creates a barrier for program developers who are not proficient in English. To mitigate this gap in technology development across languages, we propose a multilingual dataset, MCoNaLa, to benchmark code generation from natural language commands extending beyond English. Modeled off of the methodology from the English Code/Natural Language Challenge (CoNaLa) dataset, we annotated a total of 896 NL-Code pairs in three languages: Spanish, Japanese, and Russian. We present a systematic evaluation on MCoNaLa by testing state-of-the-art code generation systems. Although the difficulties vary across three languages, all systems lag significantly behind their English counterparts, revealing the challenges in adapting code generation to new languages.[1]

1 Introduction

There are an increasing number of applications related to "code intelligence", such as code summarization (Allamanis et al., 2016; Hu et al., 2018; Ahmad et al., 2020) and natural language (NL) to code generation (Ling et al., 2016; Rabinovich et al., 2017; Yin et al., 2018a; Xu et al., 2020; Norouzi et al., 2021; Wang et al., 2021), accompanied by code-specific tasks and benchmarks (Oda et al., 2015; Zhong et al., 2017; Yin et al., 2018b; Lu et al., 2021). However, in the cases where these benchmarks include natural language, that language is almost invariably English.

There are a few exceptions, but most of them either focus on languages of specific domains (Sherborne and Lapata, 2021; Sherborne et al., 2020; Moradshahi et al., 2020) or types of code (Oda et al., 2015; Liang et al., 2021), or contain NL intents collected via automatic translation (Li et al., 2021) (Appendix A). However, similarly to how Kwiatkowski et al. (2019) argue that "natural questions" are necessary to appropriately benchmark QA systems, we argue that ensuring the naturalness and coverage of questions is essential for benchmarking code generation systems as well.

Spanish	¿Cómo sumar el campo `*precio*` de todos los elementos del modelo `*Precompra*` en Django? (How to sum the `*precio*` field of all the elements of the `*Precompra*` model in Django?)
	totaldos = Precompra.objects.aggregate(Sum(precio)).values()[0])
Japanese	2次元配列`*arr*`の要素となっている1次元配列から先頭の値のみを抜き出す (Extract only the first value from the 1D array that is the element of the 2D array `*arr*`)
	arr[:, 0]
Russian	Установить кодировку `*my_encode*` для переменных окружения пользователя '*username*' (Set `*my_encode*` encoding for '*username*' environment variables)
	os.environ('username').decode(my_encode)

Figure 1: Examples in the MCoNaLa dataset, that aim to generate general-purpose Python code snippets from source intent of multiple natural languages.

A dataset for English code generation based on natural programming questions is the CoNaLa dataset (Yin et al., 2018a). It is based on natural developer questions harvested from the Stack Overflow (SO) question answering forum. In fact, in addition to English, SO also supports four other languages (Spanish, Portuguese, Japanese, and Russian) that have strong developer communities and engage in non-English programming environments. In this work, we utilize this resource to construct the MCoNaLa dataset, consisting of 341, 210, and 345 manually curated parallel samples with natural intents in Spanish, Japanese, and Russian, along with corresponding Python code snippets. Like CoNaLa, these snippets are collected from language-specific SO sites and annotated by na-

[*] Equal contribution.
[1] Code and data are available at https://github.com/zorazrw/multilingual-conala

tive speakers who are also proficient in the Python programming language.

To provide insights in the state of code generation on this new resource, we conduct comprehensive experiments with three state-of-the-art text generation models in the context of cross-lingual transfer, by unifying training and testing NL via translation (Ruder and Sil, 2021; Shi et al., 2021; Shima and Mitamura, 2010; Hartrumpf et al., 2008), or utilizing a multilingual NL encoder such as MBART (Liu et al., 2020). Our results suggest that cross-lingual NL-to-Code generation is challenging. Among all languages and experiment settings, the highest average BLEU score is 7.28, far behind that of English, which achieves 33.41, presumably because English resembles Python more than other NLs. In addition, we find models with task-specific modules and training outperform generic seq2seq models, yet the discrepancy between languages are consistent across all baseline models. In all, our corpus and experiments demonstrate the varied difficulty of the NL-to-Code generation task under different languages, emphasizing the need to develop a language-comprehensive approach to code intelligence.

2 The MCoNaLa Dataset

2.1 Task Definition

Concerning the task of answering natural language questions with machine-executable programs, our focus is to build a benchmark dataset to evaluate models for their ability to encode NL *intents* in multiple languages and generate code *snippets*. For each example in Figure 1, the *intent* above asks how to achieve a particular goal, and the *snippet* below responds with a piece of Python code.

2.2 Annotation Workflow

Our goal is to collect *intent-snippet* parallel data in multiple natural languages. In this section, we outline the main workflow for data annotation: (1) language source selection, (2) valid SO post identification, and (3) parallel sample annotation.

Language source and selection Besides the English version, Stack Overflow also has forums available in four other languages: Spanish, Portuguese, Japanese, and Russian. Data annotation in each language requires a native speaker of that language, who should also be proficient in both English and Python. Due to the high cost and difficulty of hiring

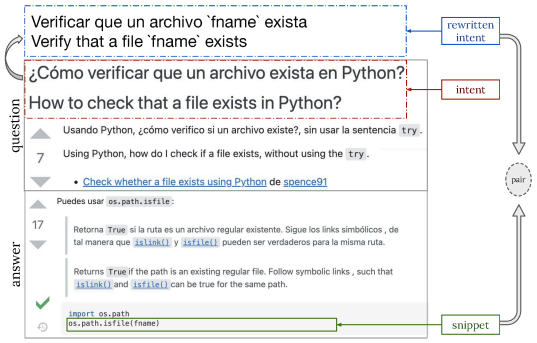

Figure 2: Illustration of the annotation process.

reliable annotators with such a specialized skill set, we only employ one Upwork annotator for each of Spanish, Japanese, and Russian. From the official SO data dump[2] dated March 2021, we obtained all posts in these languages. However, we were unsuccessful in finding a Portuguese-speaking annotator at the time of corpus collection.

Identifying how-to questions Following Yin et al. (2018a), annotators are first asked to identify valid posts that contain how-to type questions, which are imperative utterances seeking particular goals achievable by code. They are often in the post title or description, such as the example in Figure 2.

Posts are sent in 100-sample batches, and then categorized by annotators. To improve annotation efficiency, we bootstrapped a MBART how-to question classifier, with English examples, then iteratively multilingual samples. It achieves an accuracy of 72.50%. We then automatically filter the probable invalid posts using this classifier and designate the rest for manual annotation. We collect all valid posts and extract questions as raw intents, for subsequent parallel data annotation.

Collecting intent-snippet pairs For each post, we ask the annotators to find at most three snippets of Python code that correctly answer the extracted question. However, questions from post title or description are often ambiguous, especially in respective context of answer snippet, such as the example in Figure 2, that the question does not specify the names of "data" and "list" variables to allow precise code implementation. To disambiguate the intent and align it with a snippet, we ask annotators to rewrite the intent by: (1) specifying variable names appearing in the answer snippet, and (2) clarifying commands with reference question descriptions. Concretely, variable names and data types in the rewritten intent

[2] https://archive.org/details/stackexchange

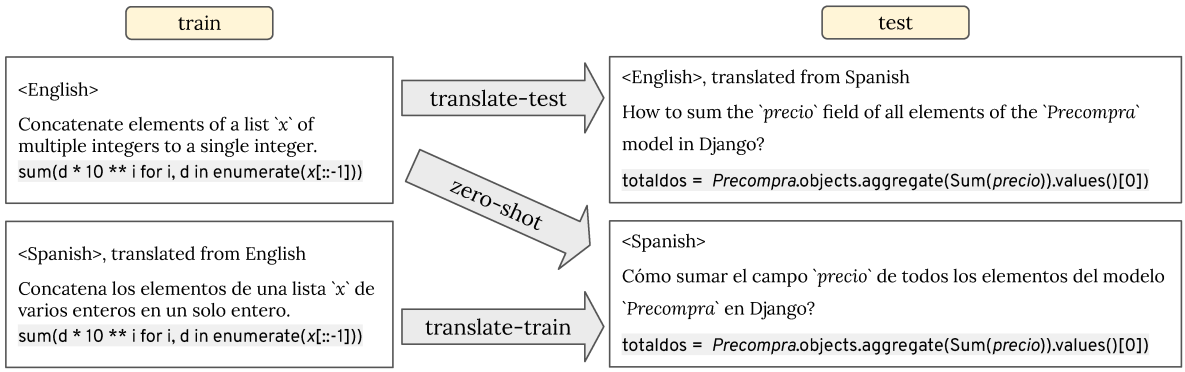

Figure 3: Example usage on the original English and Multilingual samples in three settings.

need to be surrounded by the ASCII grave accent marks (e.g., `data`), string literals or file paths should use singular typographic quotation marks (e.g., 'file1.txt', 'https://www.abc.com/').

The final MCoNaLa dataset consists of 341, 210, and 345 intent-snippet pairs in Spanish, Japanese, and Russian. It is noteworthy that the goal of MCoNaLa is to benchmark cross-lingual NL-to-Code generation task and mainly for testing purposes, instead of curating large-scale dataset for training. While its size is relatively small given the collection difficulty, we show that it can reliably inform significant method improvements (§ 3.3). We believe it is important for our dataset to be representative of the naturally occurring questions in respective language environments.

2.3 Quality Analysis

To ensure high data quality as intended, we checked 15 random samples from each language subset. Each rater score NL intents and code snippets from 1 to 5 based on their correctness and specificity.

The results demonstrate the high quality of our dataset, achieving 4.78, 4.65, 4.78 points on Spanish, Japanese, and Russian intents; and 4.84, 4.89, 4.78 points on their corresponding code snippets. Meanwhile, three raters present high agreement – the Fleiss' Kappa measure is 64.29 for NL intents and 69.49 for code snippets – both numbers indicate substantial agreement among the raters.

3 Method

To provide insights about evaluating on MCoNaLa, we demonstrate potential dataset usage in three train-test settings (§ 3.1), and propose to adapt three baseline models from either multilingual (NL) or code understanding to achieve both ends (§ 3.2).

Because the size of MCoNaLa allows only testing purposes, we resort to its larger English counterpart, CoNaLa (Yin et al., 2018a), to allow training. CoNaLa contains 2,879 manually annotated samples and 600k samples extracted from English SO forum and API documents, which can serve as a sufficient source for training. Given this usage, we denote the three test languages as *target* languages.

3.1 Train-Test Settings

We adopt three settings from two paradigms (Hu et al., 2020) as illustrated in Figure 3: (1) translating intents in train (*translate-train*) or test (*translate-test*) sets to bridge the language gap, (2) using multilingual encoder to transfer from English to target languages (*zero-shot*).

For each target language, we can align the languages of training and testing intents and use a monolingual encoder. The *translate-train* setting translates English intents in CoNaLa to each target language for training and then tests with MCoNaLa samples. *translate-test* translates MCoNaLa intents in three target languages into English. Because it is not feasible to manually translate 600K+ intents, we use existing multilingual machine translation (MMT) models to automate translation. We benchmarked several open-source options, as elaborated in § 4.2, and settled on the M2M-124 model used on the FLORES-101 dataset (Goyal et al., 2022).

Also, we can train models on English samples and directly evaluate on MCoNaLa samples in target languages *zero-shot*, requiring models to encode multiple NLs, further, transfer the code generation ability from English context to target ones.

3.2 Baseline Models

We introduce three baseline methods targeting the above train-test settings. We encourage readers to refer to the original papers for more details.

In a monolingual context, models should function in target languages for *translate-train* and En-

glish for *translate-test*. TRANX (Yin and Neubig, 2018) is a BiLSTM-based encoder-decoder model that uses a transition-based abstract syntax parser to map NLs into formal meaning representations (MR) such as Python programs. It is agnostic to input languages and hence can be evaluated on both translated settings. TAE (Norouzi et al., 2021) is the state-of-the-art method on CoNaLa by training a generic transformer with an added target autoencoder (TAE) objective. However, it is built with (English-)BERT and is intended for English scenarios, therefore only tested on *translate-test*.

As is required by *zero-shot* evaluation, we adopt a multilingual model, MBART (Liu et al., 2020), which is a seq2seq model pre-trained on 25 natural languages including our target ones. Note that MBART can also function in monolingual contexts, for both *translate-train* and *translate-test* settings.

3.3 Experiment

We train baseline models in their available settings, then tokenize the generated and reference code snippets following Yin and Neubig (2018) to evaluate the BLEU-4 scores. We report the average scores of five rounds using different random seeds.

Model	Setting	Language				
		en	es	ja	ru	avg.
MBART	translate-test	25.20	2.38	3.07	2.04	2.50
	translate-train		**2.64**	3.45	2.65	2.91
	zero-shot		2.49	1.83	2.28	2.20
TRANX	translate-test	32.26	2.46	8.34	8.12	6.31
	translate-train		2.44	6.11	6.02	4.86
TAE	translate-test	**33.41**	2.39	**9.90**	**9.56**	**7.28**

Table 1: BLEU scores of baselines for various train-test settings in English (en) and target languages (es, ja, ru).

In Table 1, first, scores on target languages average to at most 7.28, much lower than 33.41 on English, revealing the similarity of English and Python, and the difficulty of generating code from other languages. Second, models with code-specific designs and training (TRANX and TAE) performs better in general. The lower scores of MBART potentially suggest a certain representation gap between NL and PL. Third, results on two code-specific models show consistent variations across languages: scores are lower for Spanish, but rise similarly on Japanese and Russian. As we will discuss in § 4.1, this is possibly due to the distributional gap between languages with varied complexity.

3.4 Significance Test

To verify the effectiveness of MCoNaLa, we perform significance tests (Dror et al., 2018) to show its capability of showing significant differences between systems. We conduct paired bootstrap resampling tests with each pair of models in their available settings, using a sample rate of 0.5 and a sample size of 10,000.

Setting	Language	Win Rate (%)			Tie	p-value
		MBART	TRANX	TAE		
translate-test	es	0.532	0.402	-	0.066	0.468
		0.522	-	0.396	0.102	0.478
		-	0.508	0.448	0.044	0.492
	ja	0.000	1.000	-	0.000	0.000
		0.000	-	1.000	0.000	0.000
		-	0.002	0.998	0.000	0.002
	ru	0.000	1.000	-	0.000	0.000
		0.000	-	1.000	0.000	0.000
		-	0.001	0.998	0.001	0.002
translate-train	es	0.592	0.408	-	0.000	0.408
	ja	0.000	1.000	-	0.000	0.000
	ru	0.000	1.000	-	0.000	0.000

Table 2: Significance testing results between each pair of baseline models. '-' marks the model not in the pair.

In both *translate-test* and *translate-train* settings of Table 2, code-specific systems (TRANX and TAE) significantly outperform MBART on Japanese and Russian. However, no significant differences are shown in Spanish, as expected given its relative difficulty. With significance testing, one can obtain reliable results even on a small dataset. While small sizes are not entirely desirable for informative evaluation, we view them as practical reflections of data scarcity, supporting our call for more non-English resources.

4 Analysis

4.1 Variations between Languages

We first study the differences in size and snippet length between languages subsets in MCoNaLa. As listed in Table 3, snippet lengths vary across languages, and the average snippet length in Spanish is around 2.5/1.3 times of that in Japanese/Russian. A longer snippet is presumably more complex, suggesting that snippets in Spanish samples are harder to generate, and hence models perform poorer.

4.2 Intent Auto-translation

In § 3.1 we use MMT models for intent translation. To optimize translation quality, we compare three best performing MMT models: OPUS-MT (Tiedemann and Thottingal, 2020), M2M-

original intent (English)	**Prepend** string '*hello*' to all items in list '*a*'
translated intent (Spanish)	**Preparación** (prepare) de la cadena '*hello*' a todos los elementos en la lista `a`
snippet	['*hello*{0}'.format(i) for i in *a*]
original intent (English)	add a colorbar **to plot** `*plt*` using image `*im*` on axes `*ax*`
translated intent (Japanese)	画像`*im*`を使って`*ax*`の軸にカラーバーを追加
snippet	*plt*.colorbar(*im*, ax=*ax*)
original intent (English)	extend dictionary `*a*` with key/**value** pairs of dictionary `*b*`
translated intent (Russian)	расширить словарь `*a*` с ключевыми/**значительными** (significant) парами словаря `*b*`
snippet	*a*.update(*b*)

Figure 4: Examples showing that the translation errors or omits critical words in the original intent.

Language	Size	# Snippet Tokens		
		average	max	min
English	2,879	18.2	170	2
Spanish	341	42.6	343	4
Japanese	210	17.7	94	2
Russian	345	32.0	243	3

Table 3: Data size and snippet length (in number of tokens) of MCoNaLa samples between target languages.

100 (Fan et al., 2021), and M2M-124 used in FLORES-101 (Goyal et al., 2022). Since comparing in *translate-train* needs intensive re-training with different model translations, we ablate in the *translate-test* setting, using each model to translate test intents and evaluate NL-to-Code respectively.

Baseline	MMT	Language		
		Spanish	Japanese	Russian
mBART	M2M-124	2.38	3.08	2.04
	OPUS-MT	2.28	3.21	2.46
	M2M-100	1.83	2.79	2.00
TranX	M2M-124	2.46	8.41	8.09
	OPUS-MT	2.46	5.09	5.00
	M2M-100	2.04	7.38	8.48
TAE	M2M-124	2.39	9.88	9.57
	OPUS-MT	3.15	3.89	5.30
	M2M-100	2.21	8.20	9.32

Table 4: Comparing MMT models under *translate-test*.

As in Table 4, their results are close, but M2M-124 tends to be more stable across languages and baselines. Despite its relative superiority, its translation quality may still lag behind human performance, with more examples in § 4.3.

4.3 Quality of Auto-translation

To better measure the quality of translated intents, we manually check the semantic alignment between the original and translated intents, with the assistance of the Google Translate API and dictionaries. Concretely, we take 20 English CoNaLa intents and check if their semantics preserve after being translated into three target languages (*translate-train*). We similarly examine 20 MCoNaLa intents in each target language and check their English translations (*translate-test*). We use the M2M-124 translations given its best results. As shown in Figure 4, MMT translations are still sub-optimal: often mis-translate, even omit, the key words. This is especially severe on verbs that indicate certain Python operations. Hence, the translation step may impair intent-snippet alignment, being one of the major factors to the poor results in translated settings.

5 Conclusion

In this work, we extend the task of NL-to-Code generation from English-centric to multilingual scenarios. We establish the MCoNaLa benchmark that contains NL intent and code snippet pairs available in Spanish, Japanese, and Russian. Our benchmark serves for the multilingual code generation task, requiring models of both multilingual understanding and code synthesis. We conduct systematic experiments on three baseline models and show varying difficulty across languages and settings. We hope to reveal the necessity to develop, and serve as a solid test bed for language-comprehensive approaches regarding code intelligence.

Acknowledgements

We thank all the annotators for their hard work. This work was supported by the National Science Foundation under grant number 1815287.

Limitations

Although the MCoNaLa dataset makes a first step to include more natural languages aside from English, it is currently limited to the languages supported by the StackOverflow forum, since SO provides the source data for the MCoNaLa creation. This can be mitigated by extending to more languages using programming forums in other languages that have a similar purpose to SO. Besides, MCoNaLa dataset only supports literal evaluation methods such as BLEU. Given the executable nature of Python programs, it is beneficial to support more evaluation metrics such as functional correctness, robustness, and conciseness.

Ethics Statement

The MCoNaLa dataset is built to serve as a testbed for evaluating code generation systems from natural languages extending beyond English, given that an English-centric setting can harm universal accessibility to language technologies.

We hire annotators who are proficient in target languages and assist them with clearly documented instructions, flexible annotation interfaces (e.g., Google Sheets), and automated methods (e.g., using a neural classifier to filter out possibly invalid cases) to optimize the annotation efficiency. We carefully check in line with our instructions and standards, to ensure the quality of both the question posts given and the annotation results back from our annotators. We emphasize the differences between samples in different languages, because they are natural reflections of the questions that programmers asked in each specific language, similar to many works in fields such as multilingual question answering (Clark et al., 2020) and named entity recognition (Nothman et al., 2013). We reckon that it is of paramount importance to evaluate on data that was originally produced in the target language, and results may be less reliable otherwise.

Nevertheless, with the advances in models capable of generating code from natural language inputs, we should be aware of the potentially harmful usage such as concealing malicious code (Wallace et al., 2020), or generating code with security vulnerabilities (Verdi et al., 2020; Pearce et al., 2021).

References

Wasi Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. 2020. A transformer-based approach for source code summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4998–5007, Online. Association for Computational Linguistics.

Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2091–2100. JMLR.org.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *ArXiv preprint*, abs/2108.07732.

Shaunak Chatterjee, Sudeep Juvekar, and Koushik Sen. 2009. Sniff: A search engine for java using free-form queries. In *International Conference on Fundamental Approaches to Software Engineering*, pages 385–400. Springer.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *ArXiv preprint*, abs/2107.03374.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Jacob Devlin. 2018. Multilingual bert readme. https://github.com/google-research/bert/blob/master/multilingual.md.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at

scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, et al. 2021. Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48.

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. 2022. The flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538.

Sven Hartrumpf, Ingo Glöckner, and Johannes Leveling. 2008. Efficient question answering with question decomposition and multiple answer streams. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 421–428. Springer.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. 2021. Measuring coding challenge competence with apps. *ArXiv preprint*, abs/2105.09938.

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4411–4421. PMLR.

Xing Hu, Ge Li, Xin Xia, David Lo, Shuai Lu, and Zhi Jin. 2018. Summarizing source code with transferred API knowledge. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 2269–2275. ijcai.org.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083, Berlin, Germany. Association for Computational Linguistics.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2018. Mapping language to code in programmatic context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1652, Brussels, Belgium. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2950–2962, Online. Association for Computational Linguistics.

Qingyuan Liang, Zeyu Sun, Qihao Zhu, Wenjie Zhang, Lian Yu, Yingfei Xiong, and Lu Zhang. 2021. Lyra: A benchmark for turducken-style code generation. *ArXiv preprint*, abs/2108.12144.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609, Berlin, Germany. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, et al. 2021. Codexglue: A machine learning benchmark dataset for code understanding and generation. *ArXiv preprint*, abs/2102.04664.

Mehrad Moradshahi, Giovanni Campagna, Sina Semnani, Silei Xu, and Monica Lam. 2020. Localizing open-ontology QA semantic parsers in a day using machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5970–5983, Online. Association for Computational Linguistics.

Dana Movshovitz-Attias and William W. Cohen. 2013. Natural language models for predicting programming comments. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 35–40, Sofia, Bulgaria. Association for Computational Linguistics.

Sajad Norouzi, Keyi Tang, and Yanshuai Cao. 2021. Code generation from natural language with less prior knowledge and more monolingual data. In *Proceedings of the 59th Annual Meeting of the Association for*

Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 776–785, Online. Association for Computational Linguistics.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, 194:151–175.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation. In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 574–584. IEEE.

Sebastiano Panichella, Jairo Aponte, Massimiliano Di Penta, Andrian Marcus, and Gerardo Canfora. 2012. Mining source code descriptions from developer communications. In *2012 20th IEEE International Conference on Program Comprehension (ICPC)*, pages 63–72. IEEE.

Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2021. An empirical cybersecurity evaluation of github copilot's code contributions. *arXiv preprint arXiv:2108.09293*.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 878–888, Beijing, China. Association for Computational Linguistics.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149, Vancouver, Canada. Association for Computational Linguistics.

Sebastian Ruder and Avi Sil. 2021. Multi-domain multilingual question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pages 17–21, Punta Cana, Dominican Republic & Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Tom Sherborne and Mirella Lapata. 2021. Zero-shot cross-lingual semantic parsing. *ArXiv preprint*, abs/2104.07554.

Tom Sherborne, Yumo Xu, and Mirella Lapata. 2020. Bootstrapping a crosslingual semantic parser. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 499–517, Online. Association for Computational Linguistics.

Peng Shi, Rui Zhang, He Bai, and Jimmy Lin. 2021. Cross-lingual training with dense retrieval for document retrieval. *ArXiv preprint*, abs/2109.01628.

Hideki Shima and Teruko Mitamura. 2010. Bootstrap pattern learning for open-domain CLQA. In *Proceedings of NTCIR-8 Workshop Meeting*.

Aditya Siddhant, Ankur Bapna, Yuan Cao, Orhan Firat, Mia Chen, Sneha Kudugunta, Naveen Arivazhagan, and Yonghui Wu. 2020. Leveraging monolingual data with self-supervision for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2827–2835, Online. Association for Computational Linguistics.

Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT – building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, Lisboa, Portugal. European Association for Machine Translation.

Morteza Verdi, Ashkan Sami, Jafar Akhondali, Foutse Khomh, Gias Uddin, and Alireza Karami Motlagh. 2020. An empirical study of c++ vulnerabilities in crowd-sourced code examples. *IEEE Transactions on Software Engineering*.

Eric Wallace, Tony Z Zhao, Shi Feng, and Sameer Singh. 2020. Concealed data poisoning attacks on nlp models. *arXiv preprint arXiv:2010.12563*.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Edmund Wong, Taiyue Liu, and Lin Tan. 2015. Clocom: Mining existing source code for automatic comment generation. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 380–389. IEEE.

Edmund Wong, Jinqiu Yang, and Lin Tan. 2013. Autocomment: Mining question and answer sites for automatic comment generation. In *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 562–567. IEEE.

Mengzhou Xia, Xiang Kong, Antonios Anastasopoulos, and Graham Neubig. 2019. Generalized data

augmentation for low-resource translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5786–5796, Florence, Italy. Association for Computational Linguistics.

Frank F. Xu, Zhengbao Jiang, Pengcheng Yin, Bogdan Vasilescu, and Graham Neubig. 2020. Incorporating external knowledge through pre-training for natural language to code generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6045–6052, Online. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Ziyu Yao, Daniel S. Weld, Wei-Peng Chen, and Huan Sun. 2018. Staqc: A systematically mined question-code dataset from stack overflow. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1693–1703. ACM.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018a. Learning to mine aligned code and natural language pairs from stack overflow. In *International Conference on Mining Software Repositories*, MSR, pages 476–486. ACM.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018b. Learning to mine aligned code and natural language pairs from stack overflow. In *2018 IEEE/ACM 15th international conference on mining software repositories (MSR)*, pages 476–486. IEEE.

Pengcheng Yin and Graham Neubig. 2018. TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 7–12, Brussels, Belgium. Association for Computational Linguistics.

Alexey Zagalsky, Ohad Barzilay, and Amiram Yehudai. 2012. Example overflow: Using social media for code recommendation. In *2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE)*, pages 38–42. IEEE.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *ArXiv preprint*, abs/1709.00103.

A Related Work

Natural Language to Code Generation Datasets There have been several benchmark datasets for NL-to-Code generation, such as Hearthstone (Ling et al., 2016), Django (Oda et al., 2015), CONCODE (Iyer et al., 2018), and CoNaLa (Yin et al., 2018a). Other examples include datasets for problem solving, such as HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), and APPS (Hendrycks et al., 2021). A number of methods have been proposed to mine intent-snippet pairs for the purpose of code search, summarization, or generation. While our work falls in the line of mining from SO (Wong et al., 2013; Iyer et al., 2016; Yao et al., 2018; Yin et al., 2018b), other work also attempts to exploit other data sources such as API documentation (Chatterjee et al., 2009; Movshovitz-Attias and Cohen, 2013; Xu et al., 2020), code comments (Wong et al., 2015), specialized sites (Quirk et al., 2015), and developer communications (Panichella et al., 2012). One prior methodology to automatically collect large-scale parallel data is using heuristics to extract intent-snippet pairs (Chatterjee et al., 2009; Wong et al., 2013; Zagalsky et al., 2012), but this often results in compromised data quality (Xu et al., 2020). Our work resorts to a manual annotation strategy that often yields accurately aligned intent-snippet pairs.

Multilingual Learning While the bulk of code-related tasks have their NL components in English, program developers native in other languages cannot enjoy the advances in code intelligence techniques, leading to the current lacunae in multilingual learning. Our work intends to mitigate this gap by facilitating NL-to-Code generation in multiple languages beyond English. To enable language understanding across multiple languages, a number of works propose to train language models with corpus in multiple languages (Devlin, 2018; Liu et al., 2020; Conneau et al., 2020; Xue et al., 2021). In addition to multilingual training, other data augmentation techniques commonly used in machine translation (MT), such as back-translation (Edunov et al., 2018), monolingual (Sennrich et al., 2016; Siddhant et al., 2020) or generalized data augmentation (Xia et al., 2019), also inspired our experiments. However, these techniques have rarely been utilized for NL-conditioned code generation. We present preliminary attempts in the experiments.