SpanCS：面向跨语言代码生成的片段级语码转换

朱庆福[1], 周士祺[1], 王硕[2], 张致铭[1], 王昊钰[3], 陈麒光[1], 车万翔[1*]
[1]哈尔滨工业大学
[2]清华大学
[3]北京邮电大学
{qfzhu, sqzhou, zmzhang, qgchen, wxche}@ir.hit.edu.cn
wangshuo.thu@gmail.com
wanghaoyu666@bupt.edu.cn

摘要

跨语言代码生成旨在将英语到代码的生成能力迁移至其他自然语言。翻译-训练（Translate-Train）和语码转换（Code-Switching）是实现跨语言迁移的两类经典数据增广方法，两者优势互补但尚未有效结合。为此，本文提出了一种面向跨语言代码生成的片段级语码转换（SpanCS）方法。首先，该方法利用语码转换框架关联源语言上下文与目标语言片段，以促进多种语言的交互和对齐。其次，该方法利用翻译-训练方法从完整的源语言翻译中提取目标语言片段，以保证增广数据与原始数据间的语义一致性。为了公平地评价多种自然语言之间代码生成的性能差异，本文通过人工翻译与校验，基于HumanEval构建了包含10种自然语言的多语言代码生成评测基准MHumanEval。该基准上的三个主干模型的实验结果表明，SpanCS在跨语言代码生成任务上一致优于前人的数据增广方法。

关键词： 跨语言；代码生成；语码转换

SpanCS: Span-Level Code-Switching for Cross-Lingual Code Generation

Qingfu Zhu[1], Shiqi Zhou[1], Shuo Wang[2], Zhiming Zhang[1], Haoyu Wang[3], Qiguang Chen[1], Wanxiang Che[1*]
[1]Harbin Institute of Technology
[2]Tsinghua University
[3]Beijing University of Posts and Telecommunications
{qfzhu, sqzhou, zmzhang, qgchen, wxche}@ir.hit.edu.cn
wangshuo.thu@gmail.com
wanghaoyu666@bupt.edu.cn

Abstract

Cross-lingual code generation aims to transfer the ability of generating code from English to other natural languages (NLs). Translate-train and Code-switching are two common data augmentation (DA) approaches for cross-lingual transfer, which complement each other but have not been effectively combined. To this end, we propose a span-level code-switching (SpanCS) approach for cross-lingual code generation. First, it leverages the code-switching framework to correlate source language context and target language span to model the interaction and alignment among multiple languages. Second, it utilizes the translate-train approach to extract target language span from a complete source language translation, ensuring the semantic consistency between augmented data and original data. To fairly evaluate the discrepancy of code generation across multiple NLs, we construct MHumanEval, a multilingual code generation

* 通讯作者

benchmark that includes 10 NLs, based on HumanEval via manual translation and verification. Experiments on the benchmark across three backbones show that SpanCS consistently outperforms conventional DA approaches for cross-lingual code generation.

Keywords: cross-lingual , code generation , code-switching

1 引言

代码生成（Code Generation）任务旨在根据给定的自然语言描述生成相应的编程语言代码 (Chen et al., 2021; Li et al., 2022)。近年来，代码大模型技术蓬勃发展，带动了更多种类的编程语言数据向大模型的汇集 (Li et al., 2023; Roziere et al., 2023; Nijkamp et al., 2023; Daya Guo, 2024)，进而推动了代码生成任务由单编程语言（如Python）向多编程语言的扩展 (Feng et al., 2020; Zheng et al., 2023b)。与此同时，由于全球95%人口的母语为非英语的自然语言 (Guo, 2018)，进一步将代码生成任务扩展至多自然语言同样至关重要。然而，由于缺乏非英语到代码的平行训练数据与评测基准，目前多自然语言代码生成的研究仍处于起步阶段。

跨语言微调可以有效缓解多语言任务缺少带标签[1]的目标语言数据的问题，其微调带标签的源语言数据，利用大模型的语言迁移能力，将微调源语言获得的能力迁移至目标语言 (Zheng et al., 2021)。将跨语言微调应用于代码生成任务，即跨语言代码生成[2]，即可通过微调英语到代码数据实现多自然语言到代码的生成。在此基础上，将带标签的源语言（通常为英语）微调数据通过数据增广拓展至目标语言，可以进一步提升跨语言迁移的效果。常见的数据增广方法包括翻译-训练（Translate-Train） (Singh et al., 2019; Chai et al., 2022)以及语码转换（Code-Switching） (Liu et al., 2020; Qin et al., 2021)。

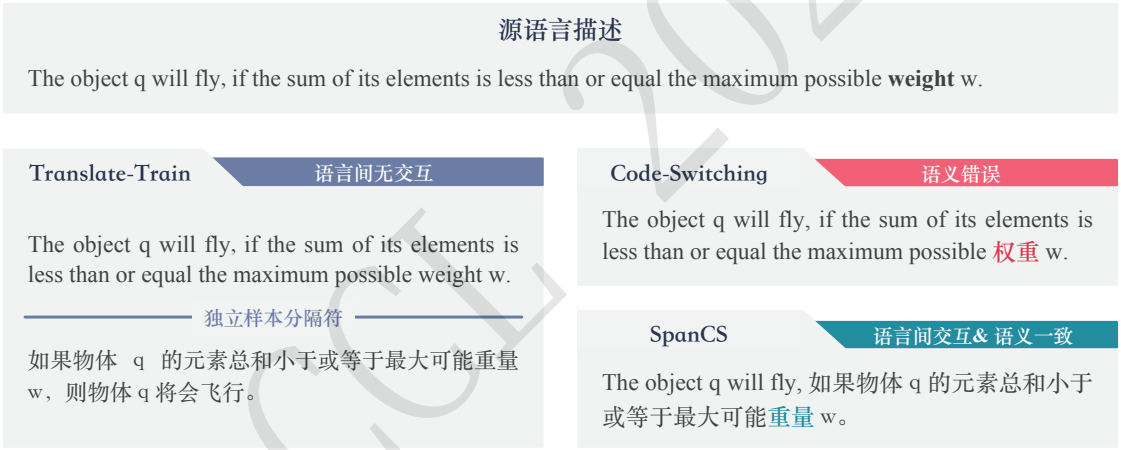

Figure 1: 翻译-训练（Translate-Train）、语码转换（Code-Switching）、片段级语码转换（SpanCS）方法示例（为避免歧义下文使用英文名称指代具体模型，中文名称泛指技术范式）。SpanCS中的“重量”为该上下文环境下源语言英语单词“weight”的正确中文翻译，表示质量大小。Code-Switching中的“权重”为错误中文翻译，表示重要程度。

具体而言，翻译-训练方法通过将带标签数据中的源语言翻译为目标语言，实现对目标语言的数据增广（如图 1所示，简洁起见，图中仅展示了各增广方法对自然语言的处理，省略了所有方法的标签，即对应的代码）。然而，翻译-训练方法中源语言样本与目标语言样本在训练过程中独立学习，缺少不同语言间的交互与对齐，因此限制了源语言到目标语言的迁移效率 (Zheng et al., 2021)。相对地，语码转换方法将源语言中的单词随机替换为目标语言中的翻译，从而有效地在语言模型建模过程中关联了不同语言。例如语码转换样本“除了1和该数本身以外不再有其他的因数的数被称为prime number”，根据中文关于素数的解释预测英文概念“prime number”的学习过程，显式地将两者在语义层面进行对齐，进而促进了两种语言的对

[1]对于代码生成任务而言，标签即为与自然语言描述对应的代码。
[2]本文中，“跨语言”和“多语言”的语言指**自然语言**，而非编程语言。

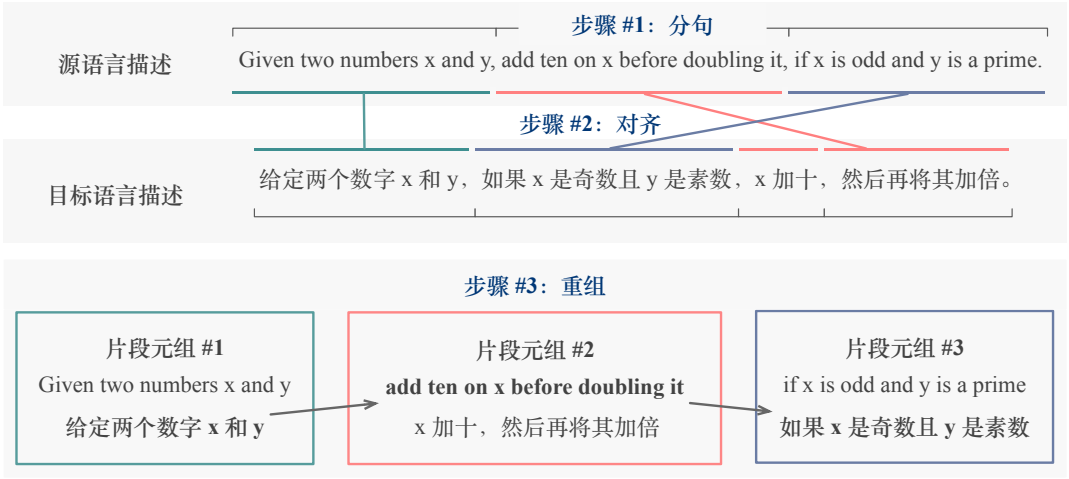

Figure 2: SpanCS的三个步骤的示例。**分句**：将源语言描述翻译成目标语言描述，然后将两者分割成子句。**对齐**：将两种语言的子句按片段对齐，其中，片段由一系列连续的子句构成。**重组**：从每个对齐的片段元组中选择一个源语言片段或目标语言片段，重组成语码转换描述。

齐。然而，当被替换的源语言单词存在多个翻译时，错误的翻译将与源语言上下文产生语义冲突（如图 1所示）。幸运的是，翻译-训练方法可以在翻译过程中通过考虑上下文获取准确的目标语言翻译，因此，基于翻译-训练增强语码转换可以自然地融合两类方法的优势，强化源语言与目标语言交互的同时保持语义一致性，构建高效高质量的增广数据。

为了融合翻译-训练与语码转换的优势，本文提出了一种片段级语码转换（**Span**-level **C**ode-**S**witching, SpanCS）方法，基于源语言的目标语言翻译选择性地替换源语言中的片段。如图 2所示，SpanCS首先将<源语言描述-代码>数据中的源语言描述翻译为目标语言描述，并将两者分割成子句。其次，以片段（存在语义等价匹配的最短连续子句）为单位对齐源语言与目标语言的子句。最后，在代码大模型的指导下，按序从每个对齐的<源语言片段-目标语言片段>元组中选择其一，组合成语码转换的增广样本。

直观上，SpanCS不仅在单个样本内显式地关联了源语言与目标语言，有效建模了语言间的交互与对齐关系，而且以上下文作为参考信息完成源语言片段到目标语言片段的转换，保证了转换前后的语义一致性。此外，为了使语码转换过程可以更好地适配下游代码生成任务，重组步骤在代码大模型的指导下展开，意味着SpanCS可以提供面向具体任务的定制化接口。

跨语言代码生成的效果可以通过多语言评测基准进行评价。然而，包含多种自然语言描述的现有代码生成评测基准，如MCoNaLa (Wang et al., 2023a)和ODEX (Wang et al., 2022)，语言之间并非完全对齐，即某种语言的一个样本在其他语言上可能不存在语义等价的平行样本。因此，一种语言的评测结果优于另一种语言，既可能是由于模型处理两种语言的能力不同，也可能是由于两种语言的评测样本难度不同，进而导致难以公平地评价模型在不同语言上的性能差异。为此，本文将HumanEval (Chen et al., 2021)的英语描述扩展至九种其他自然语言，构建了一个名为MHumanEval的新评测基准。

本文的贡献总结如下：

- 本文结合翻译-训练与语码转换的优势，提出了一种新的跨语言数据增广方法SpanCS，可以将英语代码生成能力迁移至多种自然语言。
- 本文构建了多语言代码生成评测基准MHumanEval，涵盖共10种自然语言（包括英语），用于公平地评价模型跨语言代码生成能力。
- 在三种主干代码大模型上的实验结果表明，SpanCS一致优于前人的跨语言数据增广方法。

2 相关工作

2.1 跨语言代码生成

在缺少带标签多语言数据的条件下，面向多种自然语言进行代码生成的关键是迁移源语言

（通常是英语）上习得的能力至其他语言。语码转换将源语言描述的部分单词替换为目标语言的翻译，提供了一种数据高效的解决方案 (Qin et al., 2021; Liu et al., 2020)。其数据高效特性主要体现在以下两方面，首先，将源语言描述替换为语码转换描述不必引入额外的训练样本（相对地，翻译-训练方法为每个样本引入了一条新的目标语言训练样本）；其次，语码转换的单个样本内包含多种语言，通过语言模型显式建模了语言间的交互，更高效地实现了语言间的知识迁移（相对地，翻译-训练方法中语义等价的不同语言的样本被视为独立样本分别单独训练，语言间无交互关系，知识迁移效率较低）。语码转换中的单词替换通常借助双语词典完成，因此当源语言单词在目标语言中存在多个翻译时，替换过程将存在较为明显的不确定性。为解决该问题，可以训练一个转写模型根据源语言描述直接生成其语码转换后的版本 (Tarunesh et al., 2021; Gautam et al., 2021)。本文提出的SpanCS与传统的语码转换方法的不同之处在于，它可以参考上下文信息解决词典替换的不确定性问题，并且不需要英语到语码转换的平行训练数据来学习转写模型。值得注意的是，翻译-训练与语码转换两类方法均存在曝光偏置，即训练格式不完全等价于推理格式。该现象存在的主要原因如下：一方面，训练过程引入其他语言，尤其是富资源语言（如英语），可以借助大模型的迁移学习能力提升目标语言的性能。另一方面，考虑到实际应用中为每一种语言单独开发部署一个专用模型成本较高，因此训练一个多语言模型更符合实际应用场景。

SpanCS的另一个相关工作是ERNIE-Code (Chai et al., 2022)，其在预训练过程中引入了大量的平行机器翻译数据，以构建多语言代码大模型。SpanCS与之不同之处在于，ERNIE-Code是一个**预训练**基础模型，多语言代码生成能力源自海量的<英语-代码>形式的代码生成数据与<英语-非英语>形式的机器翻译数据，两类数据以英文为枢轴实现隐式对齐。SpanCS是一种跨语言**微调**方法，旨在增广出<语码转换-代码>形式的样本，显式构造非英语到代码的对齐数据，以期通过微调进一步提升代码大模型已有的多语言代码生成能力。

2.2 多语言代码生成评测基准

一个涵盖多种语言的全面的评测基准对于公平评价多语言能力至关重要。在代码生成领域，Wang et al. (2023b)通过持续收集和标注西班牙语、日语和俄语的<自然语言-代码>平行数据，扩展了以英语为中心的CoNaLa评测基准 (Yin et al., 2018)。在此基础上，Wang et al. (2022)进一步补充了单元测试用例，使得评价可以从功能正确性的角度展开，更符合人类在软件开发过程中的验证代码质量的真实评价过程。

上述基准与本文提出的MHumanEval的区别在于：首先，MHumanEval在不同语言之间是平行的，即不同语言中的自然语言描述在语义上完全等价，由此可以在控制内容难度相同的条件下分析由语言差异导致的性能变化。其次，作为HumanEval (Chen et al., 2021)的一种扩展，MHumanEval原则上可以方便地桥接其他基于HumanEval的扩展工作。例如，将多自然语言的MHumanEval与多编程语言的HumanEval-X组合，即可构建出多自然语言到多编程语言的更为全面的多语言代码生成基准。

3 方法

本节将详细介绍SpanCS，一种用于跨语言代码生成任务的数据增广方法。如图 2所示，首先SpanCS将源语言描述翻译为目标语言描述，并分别将两者切分为子句（§3.2）。然后，将源语言描述和目标语言描述在片段级别上对齐，生成一系列<源语言片段-目标语言片段>形式的元组。此处的片段被定义为源/目标语言描述中的最短连续子句，且在目标/源语言描述中存在语义等价的对应片段（§3.3）。最后，通过在每个对齐的元组中选择一个源语言片段或目标语言片段进行重组，即可构建一个新的语码转换描述（§3.4）。

3.1 符号标记

本文使用大写字母表示随机变量（例如，V），使用小写字母表示随机变量的索引（例如i），使用粗体字母表示向量（例如：$\boldsymbol{V} = \{V_1, ..., V_n\}$）。以此，源语言描述和目标语言描述分别表示为$\boldsymbol{S} = \{S_1, ..., S_m\}$和$\boldsymbol{T} = \{T_1, ..., T_n\}$，其中$S_i$和$T_i$分别表示它们的第$i$个子句。[3] 片段的起止位置由子句的索引表示，例如，$\boldsymbol{S}$中从第i个子句开始到第j个子句结束的片段（包含起止索引）表示为$\boldsymbol{S}_{i:j}$。

[3]在分句步骤之前子句的划分是不可见的。

Translate the text below in {Chinese}, be careful not to translate variable names, function names, code, database schema, graph and HTML.Your translation should not include a prompt, such as: "This is my translation" or "Translation as follows." Here is text:
{Given two numbers x and y, add ten on x before doubling it, if x is odd and y is a prime.}

Figure 3: 将源语言描述翻译为目标语言描述的提示和输入示例。红色和绿色字体分别表示目标语言和待翻译的源语言描述样本。

3.2 分句

给定源语言描述$\boldsymbol{S}$，SpanCS将其作为一个整体翻译为目标语言描述$\boldsymbol{T}$以确保$\boldsymbol{S}$和$\boldsymbol{T}$之间的语义一致性。通常，描述中可能包含若干程序相关的成分，例如函数名和单元测试用例等，这些成分在翻译过程中应保持不变，否则将破坏后续生成的代码的正确性。例如，函数名的更改将破坏生成的函数与其调用程序之间预定义的接口。然而，机器翻译模型很难区分程序成分与纯文本。为此，本文精心设计了相应的指令，提示GPT-3.5 Turbo完成这项任务，具体指令信息如图 3所示。

在此之后，SpanCS将源语言描述和目标语言描述分别划分为子句，划分过程可以形式化表示为$\text{SEG}(\boldsymbol{S}) = \{S_1, ..., S_m\}$，$\text{SEG}(\boldsymbol{T}) = \{T_1, ..., T_n\}$，其中SEG表示分句函数。具体而言，分句的粒度应尽可能地细致，以支持后续精准的片段对齐和多样的片段级语码转换。为实现该目标，本文尽可能多地收集了常用的分隔符，并通过启发式正则实现子句划分。

3.3 对齐

语码转换中被替换的源语言成分应当与替换后的目标语言成分语义对齐。然而，由于翻译后的目标语言可能具有与源语言不同的子句数量和顺序，因此对齐难以在子句级别进行。如图2所示，源语言的三个子句被翻译成目标语言的四个子句，且源语言的第二个子句同时对应目标语言的第三和第四个子句。为应对上述问题，SpanCS的对齐将在片段（Span）级别进行。对齐过程可形式化表示为将$\{S_1, ..., S_m\}$和$\{T_1, ..., T_n\}$划分为一系列对齐的片段元组$\{(\boldsymbol{S}_{1:i}, \boldsymbol{T}_{1:p}), ..., (\boldsymbol{S}_{j:m}, \boldsymbol{T}_{q:n})\}$，其中$\boldsymbol{S}_{i:j}$和$\boldsymbol{T}_{p:q}$是语义等价的最短连续子句序列。

具体而言，SpanCS采用贪心算法实现片段对齐以确保对齐效率。[4] 首先，选择$\boldsymbol{S}$和$\boldsymbol{T}$的第一个子句作为一个候选片段元组: $\boldsymbol{C} = (\boldsymbol{S}_{1:1}, \boldsymbol{T}_{1:1})$. 其次，将$\boldsymbol{S}$和$\boldsymbol{T}$的第二个子句分别扩展到当前候选片段元组中，以获得两个包含更多子句的新候选$\boldsymbol{C}_s = (\boldsymbol{S}_{1:2}, \boldsymbol{T}_{1:1})$和$\boldsymbol{C}_t = (\boldsymbol{S}_{1:1}, \boldsymbol{T}_{1:2})$。然后，通过LaBSE度量 (Feng et al., 2022)计算每个候选内源语言片段与目标语言片段之间语义相似度。如果$\boldsymbol{C}$具有最高相似度，则表示长度最短的候选$(\boldsymbol{S}_{1:1}, \boldsymbol{T}_{1:1})$已经在语义上对齐，因此可以被接收为一个正式的片段元组。否则，$\boldsymbol{C}_s$和$\boldsymbol{C}_t$之间具有较高相似度的元组将作为新的$\boldsymbol{C}$，并重复上述扩展和比较过程，直到扩展额外的子句不能进一步提高相似度为止。

在上述对齐算法中，部分$\boldsymbol{S}$或$\boldsymbol{T}$末尾的子句可能无法成功对齐，因为将它们扩展到最后一个片段元组后不能带来进一步的相似度提升。尽管如此，SpanCS仍然将其附加到最后一个片段元组，以确保所有子句都存在对齐对象，以便在下一步中进行语码转换。

3.4 重组

给定对齐后的片段元组构成的序列，重组可以形式化表示为：按顺序从每个元组中选择源语言片段或目标语言片段，将所有选中片段拼接构成语码转换描述。因此，对于包含L个片段元组的样本，搜索空间将为2^L，通过枚举所有可能全局搜索最优解的方案时间复杂度过高，无法达到实际应用的需求。为了缓解上述问题，本文确定了最优语码转换描述的两个关键特征，以启发式查找一个近似解。首先，最优语码转换描述应当是连贯的，即每个子句与该子句的上下文保持连贯。其次，作为服务于特定任务（如代码生成）的中间步骤，最优语码转换描述应同时是任务相关的，即通过合理地选择和组织语码，辅助提升下游具体任务的性能。

基于困惑度的重组 为了将上述两个因素引入搜索过程，本文设计了一种基于困惑度的（perplexity, PPL）的重组机制。它采用束搜索（beam search）算法在代码大模型的指导下

[4]本文同时尝试了更复杂的动态规划算法SentAlign (Steingrimsson et al., 2023)，详见§5.5。

查找具有最小困惑度的不同语言片段的组合。在搜索的每一步中，分别将下一个片段元组的源语言片段和目标语片段扩展到束中的所有候选上，然后保留按困惑度排序的top-K候选，其中K表示束的大小。

直观上，大规模数据上训练的大模型所计算困惑度可以有效地评价连贯性。同时，代码大模型的引入使得重组过程可以更为精准地识别和处理程序片段。该方法的一个潜在问题是困惑度指标可能更倾向于单语描述而非语码转换描述。但基于以下两点考虑，本文并未对该倾向性进行干预。首先，从语码转换结果上看，约42%的样本产生了语码转换，因此该倾向性并未对语码转换的出现产生严重的影响。其次，本文认为该倾向性可以一定程度上降低语义错误的语码转换出现的概率。该倾向性的存在，使得仅当目标语言片段相对于源语言片段在困惑度上存在优势、且该优势足以抵消单一语言的倾向性时，才可以进行语码转换，因此可以屏蔽掉不确定性大的转换，仅保留高质量的转换。

基于先验的加权 在基于困惑度的重组的基础上，本文进一步引入了基于先验的加权机制来平衡特定语言的倾向。实际上，由于各语言的token数量在词表中的占比是不均衡的，大模型计算的困惑度在不同语言之间存在偏差。为消除该偏差，本文以源语言为枢轴对所有目标语言的困惑度进行了加权。具体而言，首先从源语言和目标语言训练数据中随机抽取100个样本，分别计算它们的困惑度，其中源语言结果记为P_s，目标语言结果记为P_t。然后，计算P_s/P_t作为每个目标语言片段在束搜索过程中的先验，以此平衡大模型在语言之间的倾向性。

4 多语言代码生成基准MHumanEval

平行的多语言评测基准对于定量对比各种语言的性能至关重要。然而，目前代码生成领域尚未有这样的基准被提出，严重阻碍了该领域的发展。为此，本文将HumanEval (Chen et al., 2021)的英语（en）描述扩展到其他九种语言构建了MHumanEval。参考 Qin et al. (2023)的选择，本文的多语言包括孟加拉语（bn）、德语（de）、西班牙语（es）、法语（fr）、日语（ja）、俄语（ru）、斯瓦希里语（sw）和泰卢固语（te）以及汉语（zh）。

具体而言，本文采用专业的商业服务[5]完成MHumanEval源语言描述到目标语言描述的翻译。与常规翻译任务不同，代码生成任务的描述中包含一些程序片段，例如函数名等，这些片段在翻译中应保持不变。为此，本文在翻译之前对该类片段进行了人工标注。在翻译过程中，每一个样本先后由一名译员翻译和一位编辑审校。为了保证构建数据的质量，所有选定的译员和编辑均具备专业认证资质（例如德语、西班牙语、泰语专业八级，日语国际一级等）和三年以上的工作经验，累计翻译量超过200万字。成本方面，每个源语言（英语）单词翻译成目标语言的平均成本为0.065美元。

5 实验

5.1 数据

SpanCS将代码生成微调数据中的自然语言描述增广到多种其他自然语言。本文实验中的微调数据为evol-codealpaca-v1，[6] 它是evol-instruct (Luo et al., 2023)的一个开源复现，在Code Alpaca (Chaudhary, 2023)的基础上借助自指令技术扩展而来。完整的数据集包含111,272个样本，增广至N种目标语言后，对于翻译-训练等方法而言微调样本将同步扩展N倍。为了进行公平的比较和控制微调成本，本文随机选取了10,000个样本展开实验，并将发布该选定数据集以供后续研究参考和对比。

5.2 基线方法

本文将SpanCS和零样本以及跨语言微调两种范式下的基线进行了比较。具体基线如下：

- **Base**：以零样本的方式直接提示基础大模型根据目标语言描述生成代码。
- **Cross-Lingual**：在<源语言描述-代码>微调数据上训练，利用大模型的迁移能力提升目标语言上能力。

[5]https://f.youdao.com/
[6]https://huggingface.co/datasets/theblackcat102/evol-codealpaca-v1

- **Translate-Train**：将<源语言描述-代码>数据翻译成<目标语言描述-代码>数据，并将两者混合用于微调的跨语言数据增广方法 (Hu et al., 2020)。公平起见，该方法的目标语言描述使用了SpanCS分句步骤中的GPT-3.5 Turbo的翻译结果（详见§3.2）。

- **Code-Switching**：另一种跨语言数据增广方法，以一定的概率（本文中为0.7）随机选取源语言描述中的单词替换为目标语言翻译。

值得注意的是，本文的基线方法中未列入基于转写的语码转换方法 (Tarunesh et al., 2021; Gautam et al., 2021)，因为用于构建该类方法的语码转换训练数据对于大多数目标语言而言在现实中很难收集。

5.3 训练与推理设置

所有基线方法和SpanCS都是模型无关的，因此可以适用于各种代码大模型。为了提供全面的效果对比，本文选取了代码生成领域应用最广泛的三种代码大模型作为主干模型开展实验，包括：CodeGen2.5-7B-mono (Nijkamp et al., 2023)、CodeLlama-7B-python (Roziere et al., 2023)和DeepSeekCoder-7B-base (Daya Guo, 2024)。简洁起见，默认情况下，下文将省略代表具体参数与版本的后缀。

主干模型	方法	bn	de	en	es	fr	ja	ru	sw	te	zh	Avg.
CodeGen	Base	6.7	9.1	10.9	10.3	11.5	10.9	9.1	9.7	7.9	11.5	9.8
	Cross	24.3	37.1	46.3	45.7	39.6	36.5	**42.0**	27.4	24.3	34.1	35.7
	Trans	**28.0**	**40.2**	45.1	43.2	40.8	35.9	38.4	**29.2**	23.1	**42.0**	36.6
	CS	**28.0**	37.1	**47.5**	45.7	37.1	38.4	40.2	26.2	**26.8**	36.5	36.4
	SpanCS	**28.0**	39.0	46.9	**46.3**	**45.7**	**42.0**	41.4	26.8	25.6	38.4	**38.0**
CodeLlama	Base	23.7	31.1	42.6	34.1	34.1	33.5	35.9	26.2	21.9	34.7	31.8
	Cross	30.4	43.9	50.0	**54.2**	47.5	39.6	41.4	**31.7**	**29.2**	**45.1**	41.3
	Trans	**34.7**	42.0	50.6	50.6	50.0	38.4	46.3	26.2	25.6	43.2	40.8
	CS	31.7	**47.5**	50.6	51.2	46.3	40.2	46.3	26.2	**29.2**	42.6	41.2
	SpanCS	34.1	42.0	**56.0**	53.6	**53.0**	**42.0**	**46.9**	30.4	**29.2**	**45.1**	**43.2**
DeepSeek	Base	20.1	34.7	44.5	34.7	32.3	42.0	39.0	29.2	23.1	47.5	34.7
	Cross	39.6	61.5	62.8	62.8	**67.0**	**57.9**	59.7	32.9	34.7	59.7	53.9
	Trans	**40.8**	**63.4**	64.0	62.1	65.8	54.8	59.1	**42.0**	32.9	59.7	54.5
	CS	38.4	**63.4**	60.9	61.5	66.4	57.3	59.7	39.6	**37.1**	57.9	54.2
	SpanCS	**40.8**	61.5	**67.0**	**66.4**	65.8	**57.9**	**66.4**	39.6	32.9	**62.8**	**56.1**

Table 1: Cross-Lingual (Cross), Translate-Train (Trans), Code-Switching (CS)和SpanCS在三个代码主干大模型（7B）下的pass@1结果。最后一列(Avg.) 表示九种语言的平均性能。

实验中三个主干模型下的所有方法（基线方法和SpanCS）均采用了统一的训练程序。参考目前监督微调范式下的SOTA方法evol-instruct (Luo et al., 2023)的设置，本文的batch大小设置为512，最大长度设置为2048。所有参数由AdamW (Loshchilov and Hutter, 2017)优化器进行更新，其中学习率为1e-3，warmup步数为1,000，训练过程持续三个epoch。推理过程通过贪心策略解码输出代码，并通过pass@1 (Chen et al., 2021)指标对结果进行评价。在SpanCS方法中，基于困惑度的重组的束的大小设置为16，源语言为英语，目标语言为MHumanEval中的其他九种自然语言。

5.4 实验结果

各方法在MHumanEval评测基准上的pass@1结果如表1所示。三个主干模型的性能排名从高到低依次为DeepSeekCoder、CodeLlama、CodeGen2.5，大体上呈现出与各主干模型训练数据量相一致的趋势。在每个主干模型中，Base方法由于缺少针对指令跟随能力以及跨语言能力的微调训练，因此其效果显著弱于其他方法。Cross-Lingual方法在<英语-代码>数据上进行了微调，补齐了指令遵循能力，因此相比于Base方法有了极为显著的提升，但跨语言能力的学习仅限于预训练阶段，仍有较大的提升空间。Code-Switching与Cross-Lingual相比效果相当或略有优势，证明了在单个样本中通过语码转换建模语言间交互关系的有效性，该范式同时也是SpanCS所基于的范式。Translate-Train的稳定性相对

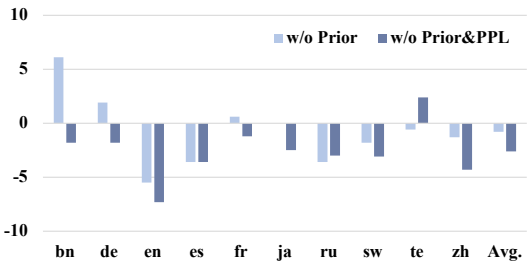

Figure 4: SpanCS去除基于先验的加权(w/o Prior) 与去除整体基于困惑度的重组(w/o Prior&PPL)的消解实验。每个柱状图代表消解前与消解后的pass@1差值。

You are an impartial rater and need to rate the pairs. Below are segmented sentences from two different languages and the pairings of these small sentences. The paired sentences are required to be semantically aligned as much as possible. The pairing can be many-to-many, but the fine-grainedness needs to be as high as possible. The score range is 0-10. The higher the score, the better the alignment of the paired sentences.The formate of score is [[rate]],for example:'Score:[[10]]'
{Given two ⋯}; {给定两个⋯}; {<Given two numbers x and y, 给定两个数字 x 和 y>, ⋯}

Figure 5: 基于GPT-4的对齐策略评价的提示词。待评价的样本按红色部分所示的格式输入，包含：完整的源语言描述、目标语言描述以及对齐后的片段元组。

较差，其在DeepSeekCoder和CodeLlama主干模型上优于Cross-Lingual和Code-Switching，但在CodeGen2.5主干模型上弱于其他方法。本文认为这是由于不同主干模型的多语言能力存在差异，在低资源语言场景下（例如sw与te），Cross-Lingual与Translate-Train在CodeLlama上的差异显著大于在CodeGen2.5、DeepSeekCoder上的差异。

相比之下，SpanCS的平均表现一致优于所有基线方法，且该提升具备统计显著性（t-test, p <0.05）。一方面，SpanCS进一步提升了Code-Switching的性能，证明了借助翻译-训练方法，以片段为单位进行语码转换保持语义一致的有效性。另一方面，相比于Translate-Train，SpanCS降低了CodeLlama主干模型在低资源语言（sw与te）场景下的不稳定性，表明SpanCS在跨语言迁移中更为有效。同时，上述迁移是通过语码转换范式实现的，该范式显式地在单个样本内建模了不同语言之间的交互。综上，SpanCS同时利用了翻译-训练和语码转换的优势，高质量地保证了增广前后的语义一致性、高效地建模了语言间的交互对齐关系，从而提高了整体性能。

5.5 实验分析

为了进一步了解SpanCS的有效性，本文以性能最好的DeepSeekCoder为测试平台进行了更为深入的分析。

消解实验 为了研究基于先验的加权和基于困惑度的重组对SpanCS整体效果的贡献，本文对相应模块进行了消解，图4展示了消解前与消解后的性能差值。首先，仅从基于困惑度的重组中去除基于先验的加权，结果记为w/o Prior。其次，将基于困惑度的重组（包含基于先验的加权）整体替换为一种随机策略，即随机选取一个源语言描述的片段替换为与其对齐的目标语言片段，结果记为w/o PPL。实验结果显示，消解前后的性能差值大多为负，表明这两个机制对最终性能均存在积极的贡献。其中，te与其他语言表现不一致的主要原因在于，大模型对少资源te语言处理能力较差。作为一个参考，表1中三个主干模型下五种方法的平均性能，te的结果是所有语言中的最低值。因此，基于困惑度的重组模块利用大模型计算te的困惑度时，可能存在更大的误差。

将源语言和目标语言的连续子句对齐成片段元组是SpanCS的关键步骤，为验证默认的贪心对齐策略的有效性，本文从九种目标语言中各抽取500个样本进行了评价。对齐效果的评价是一个较难的任务，待评价样本同时涉及多种语言与编程知识，因此人工评价需要精通各语言的编程人员才能胜任，现实中很难匹配到合适的标注者。因此本文参考前人工作(Zheng et

al., 2023a)，使用GPT-4开展评价。该工作表明，GPT-4在各类开放性任务的评价上具有与人类可比的表现与一致性。图5展示了用于评价的提示词，输入的每个待评价样本包含三个字段，分别为完整的源语言描述、目标语言描述以及对齐的片段元组，输出为0-10等级的评分。评价结果如表2所示，贪心策略和基于动态规划的SentAlign策略 (Steingrimsson et al., 2023)均获得了高于9.5分以上的评分，证明两者均有良好的对齐效果。相比之下，默认的贪心策略片段对齐效果（表2，Greedy vs. SentAlign）以及基于对齐结果的下游代码生成（表4，SpanCS vs. SpanCS+SentAlign）均略低于SentAlign，但对齐速度是SentAlign的4.7倍。在计算资源充足的条件下，基于SentAlign实现SpanCS的片段对齐可以进一步提升代码生成的性能。

方法	bn	de	es	fr	ja	ru	sw	te	zh	Avg.
Greedy	9.44	9.58	9.82	9.74	9.43	9.76	9.59	8.81	9.65	9.54
SentAlign	9.51	9.76	9.88	9.77	9.57	9.83	9.67	9.14	9.75	**9.65**

Table 2: SpanCS对齐步骤中基于贪心的策略（默认）与基于SentAlign动态规划的策略对齐效果。每种对齐策略的分数由GPT-4按0-10等级进行评分。

方法	en	es	ja	ru	Avg.
Cross-Lingual	50.3	44.4	**44.5**	**59.1**	49.6
Translate-Train	47.6	44.4	40.8	54.3	46.8
Code-Switching	50.3	45.5	40.8	58.3	48.7
SpanCS	**50.8**	**50.0**	41.4	57.1	**49.8**

Table 3: 各方法在ODEX评测基准上的pass@1结果。

方法	bn	de	en	es	fr	ja	ru	sw	te	zh	Avg.
Cross-Lingual	26.8	34.1	43.2	37.8	42.0	**34.1**	35.9	24.3	25.6	40.8	34.5
Translate-Train	**28.0**	34.1	42.0	38.4	39.0	**34.1**	**38.4**	26.8	**26.8**	38.4	34.6
Code-Switching	25.0	32.3	40.8	38.4	38.4	32.9	37.8	25.6	**26.8**	35.9	33.4
SpanCS	25.6	34.7	**47.5**	**42.6**	42.6	32.3	37.1	**28.6**	23.7	39.0	35.4
SpanCS+SentAlign	**28.0**	**39.6**	45.7	41.4	**43.9**	32.3	37.1	28.0	25.6	**42.0**	**36.4**

Table 4: 各方法在DeepSeekCoder-1.3B-Base主干模型下的pass@1结果。

泛化性 为验证SpanCS的泛化性，表3展示各方法在ODEX评测基准上的结果。Translate-Train的平均性能显著弱于其他方法，主要是因为其在日语（ja）和俄语（ru）中的表现较差，这与MHumanEval基准上的结果相一致。相比之下，SpanCS获得了最好的平均性能，并显著提高了西班牙语（es）的性能，表明其可以更好地将在源语言中学习的知识迁移到其他语言。为进一步验证SpanCS在不同量级模型上的性能，本文进一步对比了各方法在DeepSeekCoder-1.3B-Base[7]上的结果。如表4所示，SpanCS在1B量级的模型上获得了最优平均性能，证明其在模型量级维度上同样具备泛化性。

训练效率 图6展示了SpanCS和Translate-Train（DeepSeekCoder上除了SpanCS之外的最佳方法）的pass@1指标随着训练过程的变化。SpanCS以1.3倍的速度达到了Translate-Train的最佳性能，同时，在相同训练量下性能超过Translate-Train 3%，证明其具有更优的训练效率。

值得注意的是，由于英语在所有语言中处于枢轴位置，因此每种语言在消解后未解决的困惑度偏差和翻译错误（如图8中SpanCS-Filtered）会级联作用于英语，导致英语降幅最为明显。尽管如此，作为枢轴的中心位置同时也为英语提供了更多的机会来整合其他语言的改进。

可视化 跨语言研究的核心是不同语言相互对齐，以此将源语言知识迁移至目标语言。为了直观地理解对齐的效果，本文在图7中通过PCA (Wold et al., 1987)方法展示了对各种语言的隐层状态的可视化效果。参照Qin et al. (2021)的可视化设置，本文从MHumanEval中随机选取

[7]除此处分析实验外，其余实验均在7B量级模型上完成，模型细节详见§5.3。

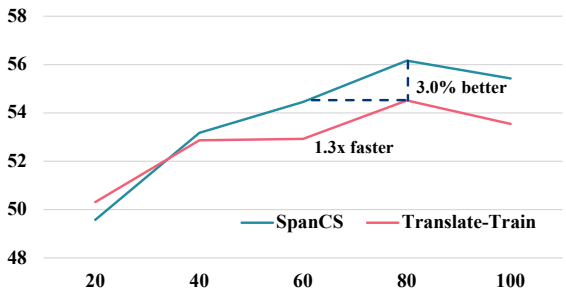

Figure 6: Translate-Train与SpanCS训练效率。横、纵坐标分别表示训练步数与pass@1结果。

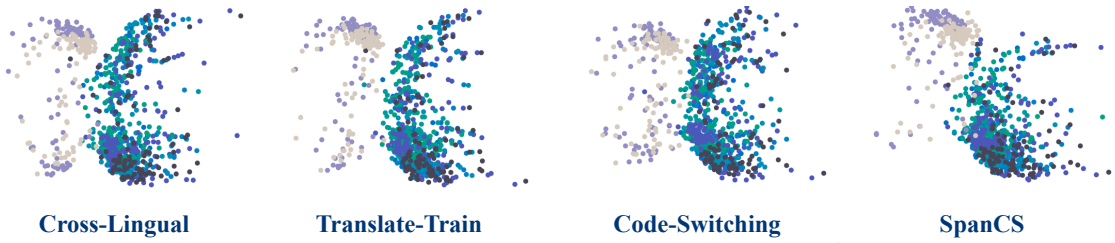

Figure 7: 基于PCA的隐层状态可视化。不同语言由颜色区分。基线方法中的各种语言相对松散地分布在隐层空间的四个角落，SpanCS更为集中地分布在隐层空间的对角线上，语言间的对齐效果更优。

	案例 #1	案例 #2
源语言描述	Referring to the dictionary labeled as 'B', ..., It should be able to surgically expunge those key-value pairs wherein the values are marked less than 10. // Dictionary B B = {'one': 1, 'ten': 10, 'twenty': 20, 'zero': 0}	Implement a Python program to parse the given XML data … Adding to it, the XML data includes … <subject>Maths</subject>… The program should also handle cases where a subject list may be empty
Code-Switching	Referring -ই ঊ 字典 étiqueté as 'B', ..., It উচিৎ అవ్వు способный -ರು chirurgicalement తీసివేయు those key-value জোড়া wherein ঊ valeurs are marqué কম than 10. // Dictionary B B = {'one': 1, 'ten': 10, 'twenty': 20, 'zero': 0}	Implement ઢ Python программа -ই parse করে donnés XML data … Adding kufika it, el XML 資料 includes … <subject>Maths</subject>… The программа উচিৎ কেও ಕಡಿಯಂ casos где ঊ тема kuorodesha সোমবার kuwa empty.
SpanCS-Filtered	Referring to the dictionary labeled as 'B', ..., Inapaswa kuondoa kirafiki key-value ambapo thamani zimehakikiwa kuwa chini ya 10. // Kamusi B B = {'moja': 1, 'kumi': 10, 'ishirini': 20, 'sifuri': 0}	একটি পাইথন প্রোগ্রাম বিন্যাস করুন যাতে দেওয়া এক্সএমএল তথ্যটি … এর আরও অংশগুলি সংযোজিত হচ্ছে একটি … <subject>গণিত</subject>… The program should also handle cases where a subject list may be empty
SpanCS	Kwa kurejelea kamusi iliyoandikwa kama 'B', ..., Inapaswa kuondoa kirafiki key-value ambapo thamani zimehakikiwa kuwa chini ya 10. // Dictionary B B = {'one': 1, 'ten': 10, 'twenty': 20, 'zero': 0}	একটি পাইথন প্রোগ্রাম বিন্যাস করুন যাতে দেওয়া এক্সএমএল তথ্যটি … এর আরও অংশগুলি সংযোজিত হচ্ছে একটি … <subject>Maths</subject>… The program should also handle cases where a subject list may be empty

Figure 8: Code-Switching, SpanCS, 以及SpanCS在重组步骤中过滤掉的样本示例。红色字体表示Code-Switching中语义错误的替换，例如案例#1中英语短语“be able to”的“be”被替换为了泰卢固语中代表“成为”的单词，案例#2中英语短语“adding to”的“to”被替换为了斯瓦西里语中代表“到达”的单词。绿色字体代表应当保留但被误翻译的程序成分，例如，案例#1中Python词典中的键（key），案例#2中XML数据标签内部的内容。

了100个样本，针对模型中间层（16/32）的隐层状态平均值进行可视化。图中的每一个点代表

一种语言的一个样本，通过颜色对语言进行区分。

在可视化结果中，更好的对齐效果对应着更为集中的分布，即不同语言互相对齐时，彼此在隐层空间的重叠程度应更高。Cross-Lingual方法在未使用任何显式对齐信息的情况下，不同语言的表示分布在隐层空间的四个角落。Translate-Train的源语言描述和目标语言描述是独立训练的，可视为一种隐式对齐，因此对齐能力有限，对齐效果与Cross-Lingual相比并无显著差异。Code-Switching与其他基线方法之间的区别同样并不显著。其分布边缘甚至略微呈现出发散的趋势。本文认为这是由于其存在语义不一致的替换错误，使得对齐过程产生了混淆。相比之下，SpanCS的各种语言更集中地分布在隐层空间的对角线上，证明了其更优的对齐能力。

案例分析 图8展示了Code-Switching、SpanCS以及SpanCS中在基于的困惑度的重组步骤中被过滤掉的样本（记为SpanCS-Filtered）。在缺乏上下文信息的情况下，Code-Switching的替换很容易发生错误。例如，在案例#1中，短语“be able to”的“able”被泰卢固语中代表“成为”的翻译所替换。Translate-Train和SpanCS方法中基于大模型的翻译过程，即便显式地设置了相应的指令要求，应该保留的程序片段仍存在着被翻译的风险，例如案例#1和#2中SpanCS-Filtered的词典的键和XML数据。尽管如此，SpanCS可以在后续基于困惑度的重组阶段借助代码大模型的编程知识屏蔽掉上述存在误翻译现象的候选。

6 结论

本文提出了一种面向跨语言代码生成的片段级语码转换方法SpanCS。与现有方法相比，SpanCS有效地将目标语言片段与其源语言上下文相关联，同时保持了两者之间的语义一致性。为了公平地评价多种语言之间代码生成的性能差异，本文构建了一个涵盖共10种语言的MHumanEval评测基准。在三个主干模型上的实验表明，SpanCS的平均性能显著优于所有基线方法。此外，SpanCS更有效地促进了源语言上习得的能力向其他语言的迁移，同时具备良好的训练效率。

致谢

本文研究受到国家自然科学基金(No. 62236004，No. 62206078)等项目资助。

参考文献

Yekun Chai, Shuohuan Wang, Chao Pang, Yu Sun, Hao Tian, and Hua Wu. 2022. Ernie-code: Beyond english-centric cross-lingual pretraining for programming languages. *arXiv preprint arXiv:2212.06742*.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca.

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

Dejian Yang Zhenda Xie Kai Dong Wentao Zhang Guanting Chen Xiao Bi Y. Wu Y.K. Li Fuli Luo Yingfei Xiong Wenfeng Liang Daya Guo, Qihao Zhu. 2024. Deepseek-coder: When the large language model meets programming – the rise of code intelligence.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A pre-trained model for programming and natural languages. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online, November. Association for Computational Linguistics.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland, May. Association for Computational Linguistics.

Devansh Gautam, Prashant Kodali, Kshitij Gupta, Anmol Goel, Manish Shrivastava, and Ponnurangam Kumaraguru. 2021. CoMeT: Towards code-mixed translation using parallel monolingual sentences. In Thamar Solorio, Shuguang Chen, Alan W. Black, Mona Diab, Sunayana Sitaram, Victor Soto, Emre Yilmaz, and Anirudh Srinivasan, editors, *Proceedings of the Fifth Workshop on Computational Approaches to Linguistic Code-Switching*, pages 47–55, Online, June. Association for Computational Linguistics.

Philip J Guo. 2018. Non-native english speakers learning computer programming: Barriers, desires, and design opportunities. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–14.

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 generalization. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4411–4421.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *arXiv preprint arXiv:2203.07814*.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Zihan Liu, Genta Indra Winata, Zhaojiang Lin, Peng Xu, and Pascale Fung. 2020. Attention-informed mixed-language training for zero-shot cross-lingual task-oriented dialogue systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8433–8440.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.

Erik Nijkamp, Hiroaki Hayashi, Caiming Xiong, Silvio Savarese, and Yingbo Zhou. 2023. Codegen2: Lessons for training llms on programming and natural languages. *ICLR*.

Libo Qin, Minheng Ni, Yue Zhang, and Wanxiang Che. 2021. Cosda-ml: multi-lingual code-switching data augmentation for zero-shot cross-lingual nlp. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3853–3860.

Libo Qin, Qiguang Chen, Fuxuan Wei, Shijue Huang, and Wanxiang Che. 2023. Cross-lingual prompting: Improving zero-shot chain-of-thought reasoning across languages. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2695–2709, Singapore, December. Association for Computational Linguistics.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Jasdeep Singh, Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2019. Xlda: Cross-lingual data augmentation for natural language inference and question answering. *arXiv preprint arXiv:1905.11471*.

Steinthor Steingrimsson, Hrafn Loftsson, and Andy Way. 2023. SentAlign: Accurate and scalable sentence alignment. In Yansong Feng and Els Lefever, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 256–263, Singapore, December. Association for Computational Linguistics.

Ishan Tarunesh, Syamantak Kumar, and Preethi Jyothi. 2021. From machine translation to code-switching: Generating high-quality code-switched text. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3154–3169, Online, August. Association for Computational Linguistics.

Zhiruo Wang, Shuyan Zhou, Daniel Fried, and Graham Neubig. 2022. Execution-based evaluation for open-domain code generation. *arXiv preprint arXiv:2212.10481*.

Zhiruo Wang, Grace Cuenca, Shuyan Zhou, Frank F. Xu, and Graham Neubig. 2023a. MCoNaLa: A benchmark for code generation from multiple natural languages. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 265–273, Dubrovnik, Croatia, May. Association for Computational Linguistics.

Zhiruo Wang, Grace Cuenca, Shuyan Zhou, Frank F. Xu, and Graham Neubig. 2023b. MCoNaLa: A benchmark for code generation from multiple natural languages. In Andreas Vlachos and Isabelle Augenstein, editors, *Findings of the Association for Computational Linguistics: EACL 2023*, pages 265–273, Dubrovnik, Croatia, May. Association for Computational Linguistics.

Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. 2018. 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.

Bo Zheng, Li Dong, Shaohan Huang, Wenhui Wang, Zewen Chi, Saksham Singhal, Wanxiang Che, Ting Liu, Xia Song, and Furu Wei. 2021. Consistency regularization for cross-lingual fine-tuning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3403–3417, Online, August. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023a. Judging llm-as-a-judge with mt-bench and chatbot arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Lei Shen, Zihan Wang, Andi Wang, Yang Li, et al. 2023b. Codegeex: A pre-trained model for code generation with multilingual benchmarking on humaneval-x. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5673–5684.