一、问题描述

这个 Project 主要实现对一种低资源语言——壮语的中文翻译，壮语属于壮侗语系，是由中国语言学家李方桂先生提出并命名的。据世界民族语言网（Ethnologue）的报告显示，壮侗语系共有 91 种语言和方言，主要分布在东起中国广东、广西、海南，西至印度阿萨姆，北达四川金沙江，南抵泰国南部的区域。现有语言模型直接翻译壮语的能力很有限，本任务旨在设计一些巧妙的方法，利用提供的壮语语法书和词典等工具，实现对所给测试数据中壮语尽可能准确的中文翻译。

第一个任务根据提供的壮语语法书，以及测试数据中所给的特定语境下一些关键词的中文释义，进行中文翻译；第二个任务根据提供的壮语语法书、壮语词典、壮语-中文平行语料库，直接翻译测试数据中较为复杂的壮语。

二、解决方法

1、方法概述

两个任务我都使用了 LLM API 的方式来完成，调用了 qwen-max 模型。在每个任务中，针对不同测试样例的具体情况，从壮语语法书、壮语字典、壮语中文平行语料中挑选不同的信息，构建不同的 prompt，然后将构建的 prompt 输入 LLM，将模型输出的回答写入答案文件。

2、方法细节

(1) Sub-Task 1

在这个任务中，所给的参考材料只有壮语语法书和每一个测试样例自带的关键词释义。在加载好壮语语法书、测试数据后，对每一个样例构建 prompt，包括从壮语语法书中选取的语法要点以及对应的例句、样例本身自带的关联词释义、要翻译的壮语句子，对目标任务的确切描述。然后把构建好的 prompt 传入 LLM，即可得到翻译结果。最后得到的结果中不可避免的会存在模型输出的多余信息（如礼貌用语，注释等），所以要对翻译的结果进行后处理，主要使用了正则化匹配的方法，去除结果中包含的释义内容，使最后的结果只包含中文翻译。

其中，从壮语语法书中选择每一个测试样例对应的语法要点我经过了特殊设计。考虑到传

入整个语法书会造成大量的 token 消耗，并且可能会使 LLM 无法注意到真正有用的关键信息，于是我选择部分选择其中的语法要点传入 LLM。具体地，选择的语法要点由两部分构成，一部分是通过简单的词匹配选择的，即选择壮语语法书中对应例句与该测试样例句子中公共词最多的 20 个语法要点；另一部分是基于 TF-IDF 找到壮语语法书中对应例句与该测试样例句子的余弦相似性最高的语法要点，这个相似性分数又由两部分组成，一部分是例句的壮语单词、关键词组成的文档与测试样例中壮语句子的余弦相似度；另一部分是例句中关键词的中文释义与测试样例关联词的中文释义的 Jaccard 相似度，二者 2:1 进行加权，最后按得分从高到低取前 70 个语法要点。为了避免两部分语法要点中有重复使得 LLM 过分地重视某一语法规则的作用，最后还要进行去重处理。

(2) Sub-Task 2

在这个任务中，所给的参考材料是壮语语法书、壮语字典、壮语中文平行语料，但是测试样例中不再包含关联词释义，并且测试的句子更难更复杂。在加载好数据后，对每一个样例构建 prompt，包括从壮语词典中查找测试样例句子中每一个单词的释义、从壮语语法书中选取的语法要点以及对应的例句、从壮语中文平行语料库中选取的句子的壮语，中文以及来源信息、要翻译的壮语句子、对目标任务的确切描述。然后把构建好的 prompt 传入 LLM，即可得到翻译结果。最后得到的中文翻译结果相比 Sub-Task 1 包含更复杂的干扰信息，需要更多正则化匹配规则来做后处理，使最后得到结果只包含中文翻译。

其中，从壮语语法书中选择每一个测试样例对应的语法要点和 Sub-Task 1 的方式类似但不完全相同，只基于 TF-IDF 找到壮语语法书中对应例句的壮语单词，关键词组成的文档与该测试样例壮语句子的余弦相似度最高的 60 个语法要点。从壮语字典中查找测试样例壮语句子中每一个单词的所有释义，全写入 prompt 中供 LLM 参考。壮语中文平行语料库中句子很多，无法全部写入 prompt 中，于是我也采用 TF-IDF 的方法寻找语料库中壮语句子与测试样例余弦相似度最高的 140 个参考资料，将他们的壮语和中文以及来源都写入 prompt 并提醒 LLM 在翻译时注意与相似句子平行语料的来源语境和用词习惯相近。

3、代码实现

在每个文件夹中，main.py 文件主要处理数据的读入，加载每个测试样例，调用方法构建 prompt，进行翻译，最后将结果和构建的 prompt 分别储存。utils.py 文件中包含了一系列的方法，用于读入数据、选择参考材料、构建 propmt、翻译句子、对翻译结果后处理。

三、实验结果

1、评测得分

提交至 Kaggle 上的 Leaderboard 测评，Sub-Task 1 结果为 0.62911，Sub-Task 2 结果为 0.27198。(Figure 1,Figure 2)

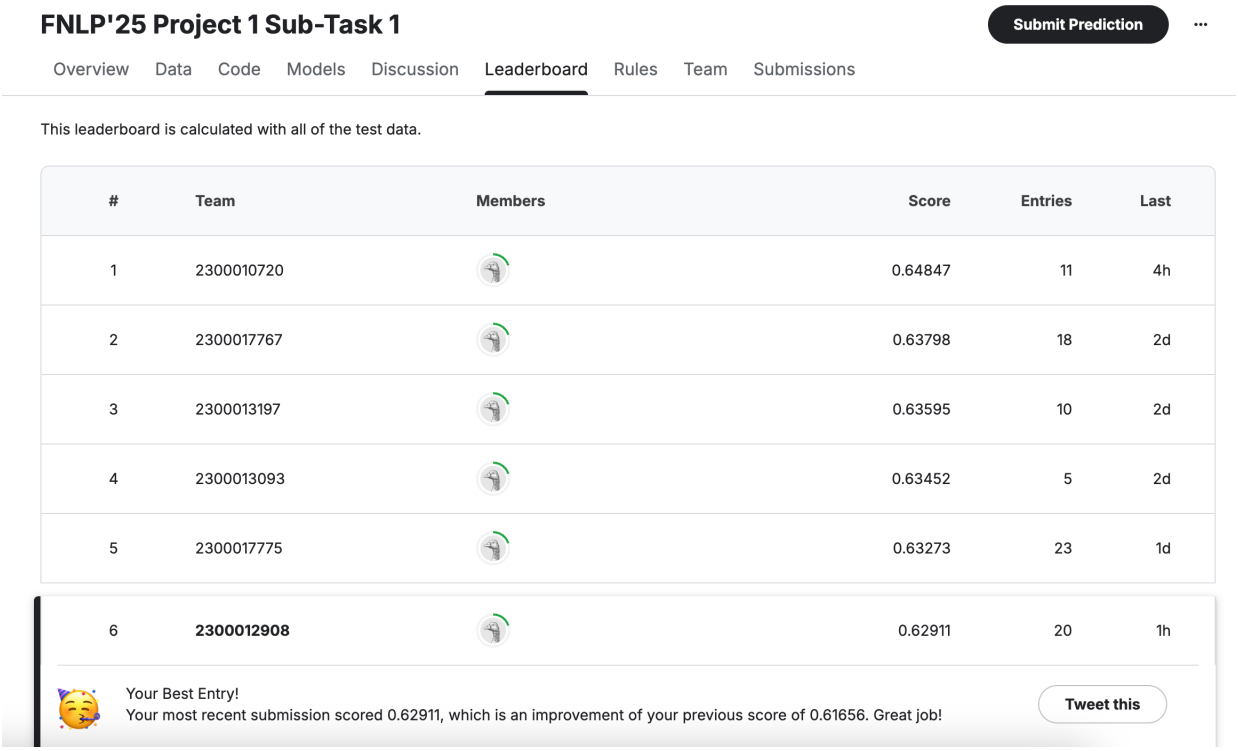

图 1: Sub-Task 1

2、有效性分析

选取了适当的 baseline 对我的方法进行有效性分析如下：

(1) Sub-Task 1

不使用壮语语法书，编写 prompt 传入 LLM 直接翻译测试样例的壮语句子 (详见每个 sub-task1 文件夹中的 zero-shot-experiment.py)，效果不好，评测得分 0.39204(Figure 3)。

除此之外，我还做了其他的实验证明我方法的有效性。

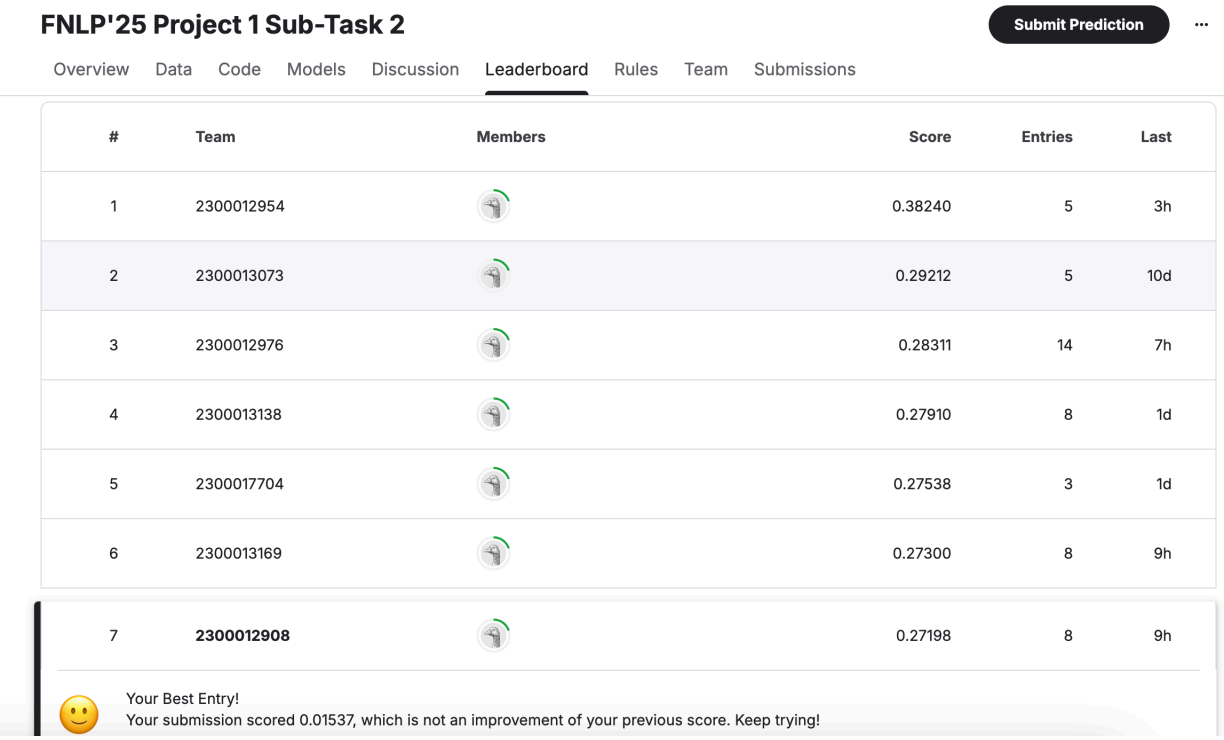

图 2: Sub-Task 2

图 3: Sub-Task 1 zero-shot-experiment

实验一

在选择较少数量语法要点（第一部分 2，第二部分 4），得到的效果也不是很好，评测得分 0.59925(Figure 4)。

图 4: Sub-Task 1 Experiment 1

实验二

只选择第一部分，即只用简单统计公共单词的方法选择语法要点，效果不好，评测得分 0.57924(Figure 5)。

图 5: Sub-Task 1 Experiment 2

实验三

只选择第二部分，即只使用 Tf-IDF 相似度选择的语法要点，效果不好，评测得分 0.39806(Figure 6)。

图 6: Sub-Task 1 Experiment 3

实验四

随着加入 prompt 中语法要点数量的增加，效果逐渐变好，但是也不能加的太多，加的太多（第一部分 30，第二部分 100），效果反而变差，评测得分 0.61433（Figure 7）。

图 7: Sub-Task 1 Experiment 4

实验一到四只需在代码中注释或更改参数即可，不需要额外编写代码，最终确定在第一部分选 20 条，第二部分选 70 条语法要点加入 prompt 中得到的效果最好（Figure 1）。

(2) Sub-Task 2

不使用壮语语法书、壮语词典、壮语中文平行语料，编写 prompt 传入 LLM 直接翻译测试样例的壮语句子 (详见每个 subtask2 文件夹中的 zero-shot-experiment.py)，效果不好，评测得分 0.01537(Figure 8)。

图 8: Sub-Task 2 zero-shot-experiment

同样的，随着选择的语法要点、平行语料数量增多，翻译效果逐渐变好。

语法要点选择 4 条，平行语料选择 6 条，评测结果 0.20438；语法要点选择 50 条，平行语料选择 100 条，评测结果 0.24074(Figure 9)。但在选择数量太多时，编写的 prompt 太大，会超过所使用 LLM 允许的最大输入长度，最终尝试得到在语法要点选择 60 条，平行语料选择 140 条时，效果最好，评测结果是 0.27198(Figure 2)。

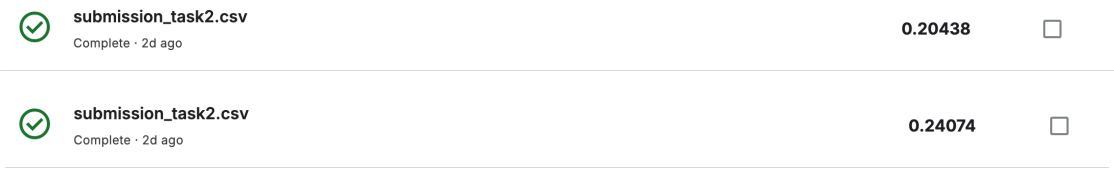

图 9: Sub-Task 2 Experiments

3、实验结果分析

(1) Sub-Task 1

在这个任务中，可以看出加入 prompt 中的语法要点越多，翻译效果越好，但是增加超过一定数量后效果反而变差。对此，我的理解是，一开始语法要点数量不多的时候，每多加入一条，都会给 LLM 提供新的有用信息，从而让它更有可能翻译出较为准确、通顺、合理的句子；但是一旦加入的数量过多，LLM 受限于对对上下文的注意能力，可能无法充分利用所有提供的语法要点，并且甚至遗忘一开始提供的较为重要的语法（因为我是按照相似度从大到小加入 prompt 的），导致翻译效果很差，此外太多的语法要点也可能使得模型为了符合一些不那么重要和相关的语法翻译出不正确的句子。

最后得到的翻译结果中，有那么一些样例无论怎么更改参数都翻译的不可能正确，并且几乎每次都错的一样，比如：”baeznaengz -> 坐车次”、”Rumzrumz fwnfwn gou cungj bae. -> 大风大雨我们都去了。” 等很多不那么通顺，甚至不合理的句子，我猜测是提供的参考材料中没有非常相关的句子，导致每次都只能让模型凭空想象。

还有一个值得注意的点是，在加入 prompt 中语法要点的数量较少时，无论怎么修改别的

参数，几乎每次都会把五叔、六子、三姐等词语翻译成第五个叔叔、第六个儿子、第三个姐姐，这个现象在增加语法要点数量后得到有效缓解。我猜测因为这些词语是独立的称谓，在语法要点中相似度较高的句子应该是词汇表面意义上的相似度高，即姐姐，叔叔，儿子，还有这些数字等，真正具有逻辑上相似性，比如二哥等词语，在 TF-IDF 方法下衡量的相似度不那么高，所以只有增加语法要点数量，这些逻辑上更相似的语法要点才有可能被加入 prompt 从而让 LLM 正确翻译出这些称谓。同样的事情还发生在农历日期的翻译上，语法要点较少时，五月初五、二月初二、六月初六都会被错误翻译成正月初五、正月初二、正月初六，这一点同样在增加语法要点数量后得到改善，我猜测原因是语法书中含有正月的句子与他们在 TF-IDF 衡量的相似度高于与他们逻辑上真正相似的语法。

(2) Sub-Task 2

与 Sub-Task 1 中一样，加入 prompt 的参考材料越多，得到的翻译效果会更好。但是由于测试样例缺乏一些关联词的解释，取而代之的只能去壮语词典中寻找释义，这就无法确保找到的释义中 LLM 可以正确推理出要翻译的句子语境下要用哪一个释义，并且也可能存在这个语境下要使用字典中没有说明的引申义，加之要翻译的壮语句子更加复杂，所以与 Sub-Task 1 相比翻译结果较为糟糕和不稳定。

我观察了翻译结果，按理说测试样例应该是由几个部分组成，每个部分是一段小故事或一段话。但是我的翻译结果对某一段话的翻译可能并不是那么的连贯，换言之可以看出他们在讲的东西很相似，但是相邻的句子翻译结果没有很强的逻辑联系。造成这样情况的原因，我猜测是由于我是一个句子一个句子处理的，每个句子分别构建 prompt 然后使用 LLM 翻译，一句完成后再继续处理下一句，所以 LLM 每次都是重新开始翻译，这些本来相关的句子得到的翻译结果就很难在逻辑上还能强相关了。把一个系列的测试样例一起构建 prompt 进行翻译，可能是一个较好的解决方法，但是本次作业使用的 LLM 一次性能处理的 token 数量有限，一次性翻译 10 至 20 个句子（甚至更多），prompt 中还要再包含每个句子所需要的众多参考材料，LLM 无法处理这么长的 prompt，所以我没有验证这个思路的可行性。

(3) 我观察到其他有意思的事情

在相同参数设定下，每次运行得到的翻译结果也会有不一样，有时这种差异微小，有时候又会很显著。对于 Sub-Task 1 中的一些例子翻译可以很好说明情况：mbiengjsae 有时被翻译成西边，有时被翻译成东边；canghdoq 有时被翻译成打造箱子（动词），有时被翻译成木匠（名词）。虽然 LLM 每次回答具有随机性可以在一定程度上解释这一现象，但翻译结果出现词性上的差异，甚至出现反义词的情况，我认为可能说明了参考资料数量还不够，该方法在翻译时仍存在不稳定的隐患。

(4) 在实验中，我发现 Sub-Task 2 在某几次运行时会因文输出违反道德规范的词语或句子而直接报错，主要是第 159,，第 172，第 213 个测试样例，在最终确定的参数下，合计一共运行了 7 次代码，成功了 4 次，另外 3 次因为上述原因在中途报错退出。

四、参考资料

LLMs API 调用教学

Prompt Engineering 介绍

Prompt 设计要点

TF-IDF 高效文本检索

Jaccard Similarity

平行语料在翻译中的应用