Large Language Models for 5-Digit OECD Health Sector Classification

David Zhu, Michelle Zhou, AidData

June 17, 2023

ABSTRACT

Recent advancements in AI have showcased remarkable potential of Large Language Models (LLMs) in various natural language tasks, such as reading comprehension and categorizations. This paper delves into the application of LLMs in the health industry, specifically, in 5-digit encoding task conforming to the OECD's sector classification codes. We will introduce our framework for performing sector specific 5-digit encoding task based on few-shot learning with automatic selection of in-context examples. Our pipeline incorporates data processing, annotation selection, vector database establishment, prompt engineering, model deployment and testing. We have employed two primary tools in our approach. The first, fast Vote-k (Hongjin et al., 2021), efficiently selects diverse and representative examples for annotation purpose, largely reducing annotation costs. The second, dynamic retrieval algorithm selects in-context examples (Liu et al.) that provide most semantically aligned examples for each data piece. Comparing to random selection of in-context examples, our implementation of dynamic example retrieval improved the model accuracy from 88% to 97%.

1 Introduction

1.1 Background

AidData's TUFF (Tracking Underreported Financial Flows) team aims to provide comprehensive information about aid and credit from official sector donors and lenders who do not publish comprehensive or detailed information about their overseas activities according to existing global reporting systems, such as the OECD's Creditor Reporting System and the International Aid Transparency Initiative (IATI). Sector classification based on OECD's guidelines are represented by 3 and

5 digit purpose codes. These purpose codes provide fine-grained information that aims to answer the question: "which specific area of the recipient's economic or social structure is the transfer intended to foster?"

Currently, within AidData's workflow, 3-digit (CRS) purpose codes are recorded for each project. However, 3-digit codes contain less fine-grained sectorial information compared to 5-digit codes. Having 5-digit purpose codes adds a new variable to the TUFF dataset, which would provide more detailed sectoral information for each project on the record and support more accurate sectoral research & analysis.

1.2 Our Approach

In our work, we frame the task of OECD 5-digit coding as a sequence-to-sequence conditional generation problem. We rely on the strong in-context learning and instruction-following capabilities of large language models and prompt the LLM with task instructions that include the OECD sector guidelines, as well as in-context examples in the form of input-output pairs in order to facilitate the model to generate the desired representation of the 5-digit code. The dataset we use contains 481 hand-coded and verified historical projects (1965-1999) in the health sector. Our best results are achieved by implementing semantic similarity-based in-context example selection, with an average accuracy of over 96% (k=5,8,10), and best accuracy of 96.84% (k=10).

Currently, our method has only been applied and experimented in the health sector, partially due to the lack of labeled datasets for benchmarking in other sectors. Future work direction includes exploring standardized procedures to extend our prompt-based framework in other sectors, as well as further exploring the reliability and sensitivity of this framework. Particularly considering how the model reacts to longer, more diverse project description texts, as well as more complex instructions for categorization.

2 Problem Statement

The creation of 5-digit codes for recorded projects involves the following steps: For each project recorded using the TUFF methodology, a project title and description are developed by research assistants, providing detailed information about the project. Additionally, a 3-digit CRS code is assigned to each project. These 3 and 5 digit codes are formatted hierarchically, meaning an additional 2 digits are added to each 3-digit sector code to represent more specific sectorial information, thereby forming a 5-digit sectorial code.

Given the detailed information for each project and the 3-digit code, the coder reviews the relevant information and determines which specific area of the recipient's economic or social structure the transfer is intended to foster, according to the OECD guidelines.

Previously, this process was conducted manually by research assistants, posing a significant challenge due to high costs and time consumption. The utilization of machine learning and large language models for encoding will not entirely eliminate the need for human verification, but it will significantly reduce the time, effort, and organization required to encode projects on a larger scale.

3 Dataset and Model Selection

The labeled data set currently in use is a product of a collaboration between AidData and Ignite Lab in the summer of 2022. This data set, which covers historical Chinese development finance health projects, contains 481 projects that document China's foreign health-sector aid history in Africa before 2000. For each project, research assistants (including myself) manually produced the 5-digit code, which was then verified by staff members during the quality assurance process. The dataset columns are as follows:

Column names	**Description**
project_id	Internal id for identification
flow	Financial flow type
title	Title of the project
year	Year of the project
year_uncertain	certainty of which the project occurred in "year" column
description	detailed description of the project
all_recipients	recipient country of the project
official_sources_count	number of official sources on record
non_official sources	list of non-official sources

Table 1: Description of the data set

For this project, only the "title" and "description" columns are used for 5-digit coding with large language models.

Due to recent advancements in the field of AI, transformer-based Large Language Models (LLMs) have become more powerful, achieving human-level or superior performance in various NLP tasks. Our work aims to semi-automate the process of 5-digit coding in various sectors using publicly available language models. In this work, we treat the process of 5-digit coding as a sequence-to-sequence conditional text generation problem. We use the recently released, open-source model FLAN-UL2 published by Google to achieve strong results in our experiments. FLAN-UL2 is a pre-trained and instruction-tuned 20B parameter model with an encoder-decoder transformer architecture. It currently ranks as the best open-source model on comprehensive NLP benchmarks such as Big-Bench hard and MMLU. We provide more details about our methodology below.

4 Methodology

We leverage the strong in-context learning and instruction-following capabilities of FLAN-UL2 to facilitate the task of 5-digit coding. We frame this task as a reading-comprehension and text-categorization problem in the task instruction, concatenating in-context examples in the form of input-output pairs.

We experimented with 0-shot and few-shot prompting, achieving an initial accuracy of 88% with 0-shot prompting, 92% (k=5) and 95% (k=7) with hand-selected in-context examples. However, the process of manually selecting in-context examples proved to be suboptimal because it requires an iterative, trial and error process. Furthermore, the selection of effective examples relies heavily on intuition. Motivated by the need to eliminate the manual selection process and increase accuracy and robustness, we implemented similarity-based in-context example selection (Liu et al), which largely further improved the accuracy to 97%.

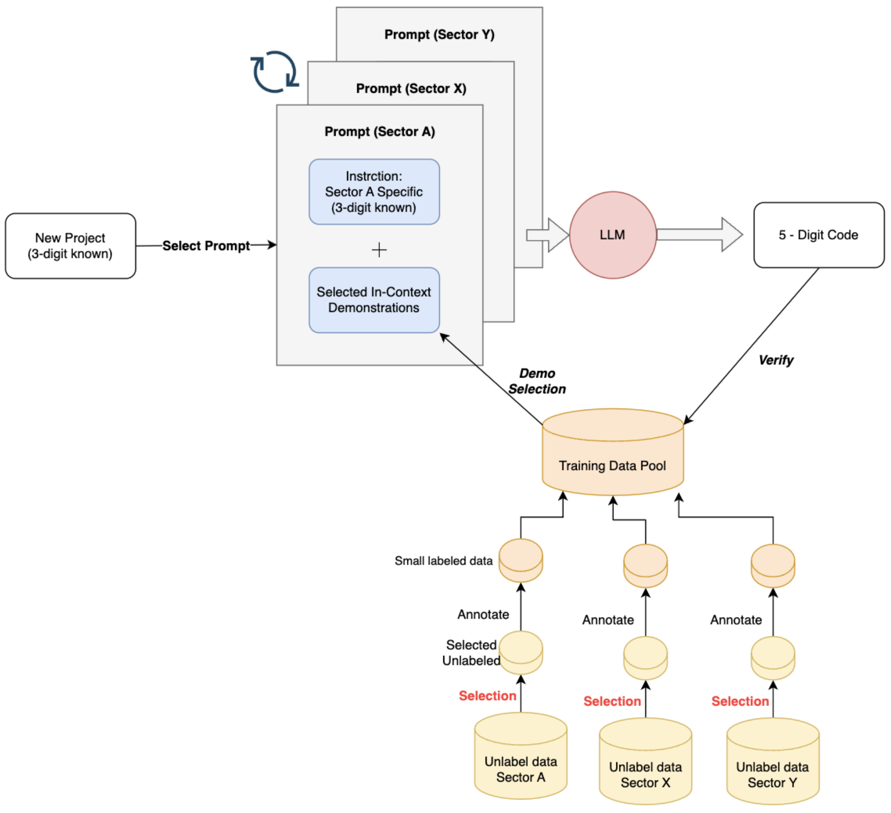

Figure 1: Flowchart of our framework for 5 digit coding

4.1 In-context Example Selection

4.1.1 Manual Selection

For the manual example selection process, we began by including five random examples. After comparing the model's results with the human-labeled results, we found that the most miscoded projects by the model fell into the 12191 Medical services category (confusing "equipment" with "infrastructure"). We thus added two more examples for code 12191 involving such cases to the prompt, aiding the model in understanding this coding criterion. The model was able to learn to distinguish this detail with just two demonstrations. Finally, with seven examples included in the prompt, we achieved a 95% accuracy on the testing data.

4.1.2 Similarity-based Dynamic Retrieval

To optimize the example selection process,

We used the standard pre-trained sentence transformer from SBERT as the embedding model. The semantic similarity-based selection consistently improved overall accuracy (k = 5, 8, 10) compared to manual selection, with an average accuracy of over 96% (best 96.84% , k=10). Further, we tested the effect of permuting the order of in-context examples to account for recency bias. We observed little variance introduced by the reversal of example order. The number of in-context examples (k) is limited by the model context length of 2048 tokens.

There are several directions for the further development of this project. The first is to explore and develop a standardized procedure to expand and apply our prompt-based method to other sectors. A few challenges are presented when generalizing our method. Because our experiments conducted so far only involve projects in the health sector, there are only about 12 different 5-digit level criteria for consideration, all of which fall under the 120 and 121 3-digit CRS codes. In a more general scenario where projects span many different sectors (each falling under a specific 3-digit CRS code), there will be many more 5-digit criteria of varying lengths to consider under each 3-digit CRS code.

Theoretically, this challenge can be overcome by dynamically switching between different prompts (including instructions and demonstrations) for projects under each sector. Assuming each project has already been assigned a 3-digit CRS code, the number of 5-digit level criteria to consider can be reduced to only those under the 3-digit CRS code, similar to the case for the health sector in our experiments. Therefore, we propose to construct prompts for each sector (i.e., for each 3-digit CRS code, construct instructions that include only 5-digit criteria expanding upon the 3-digit code). Then at inference time, we feed the model the corresponding prompt based on the 3-digit code of individual projects. It is important to note that under our current method, there is no need to train specialized sub-models for each sector (i.e., for each prompt), since we did not modify the weights of the FLAN-UL2 model via fine-tuning and rely only on in-context learning at inference time to achieve our results. This means that we can use the same set of model weights for all specialized prompts. Another challenge is the lack of readily available labeled data in other sectors to construct a sufficiently large pool for semantic similarity-based demonstration selection. One critical aspect that contributed to the success of experiments in the health sector is the ability to utilize already labeled data for dynamic demonstrations. However, in-context learning is much more sample-efficient than traditional supervised learning. A practical way to overcome the lack of labeled data is to follow the vote-k selective annotation framework detailed by Su et al. This framework selects a few (¡100) diverse and representative unlabeled projects to manually annotate. Paired with semantic similarity retrieval, this framework achieves better results with small annotation budgets. We plan to implement and follow this framework for different sectors in order to construct small but effective demonstration selection pools for in-context learning, paired with sector-specific instruction prompts.

5 Cost Analysis

For our experiments, we downloaded the FLAN-UL2 model from the Hugingface Hub, which takes up about 42 Gb GPU memory space. We then deployed the model to a AWS Sagemaker g5.12xlarge instance for inference. For our experiments, including iterations of trial and error, takes around 12 15 hours of g5.12xlarge instance runtime, with an hourly rate of \$5.67. Addtionally, since our method involves only prompting/inferencing, there is no additional fine-tuning process that will require much more powerful hardware. For further comparison, a single iteration thorugh the training set of 481 projects, using our longest and highest-performing prompt (k=10), takes around 25-30 minutes inference time.