Equivariant Adaptation of Large Pretrained Models: A Study on the NLC2CMD Competition

Abstract

This paper presents an investigation into the challenges of adapting pretrained models, specifically in the context of the NLC2CMD competition.

1 Introduction

This paper addresses the critical need for effective methods to translate natural language descriptions into executable command-line instructions. The command line interface (CLI) is an important tool for software development due to its expressiveness and efficiency. While GUIs have difficulties keeping up with the rapid pace of new features in software development, CLIs provide a text-based interface to a wide range of software functionalities. The use of natural language for CLI interaction could transform how people interact with various operating systems and cloud platforms. This paper explores the possibilities of leveraging natural language to interact with CLIs making computational resources more accessible to a wider range of users.

2 Task Description

The primary objective of the NLC2CMD task is to transform a natural language (NL) description of a command-line action into its corresponding Bash command. An algorithm is expected to model the top-k Bash translations given the natural language description. This can be represented mathematically as:

$A_{nlc} \in \{p \mid p = (c, \delta)\}; |A(nlc)| < k$

Each prediction from the model includes a set of Bash commands along with a confidence score, δ, ranging from 0.0 to 1.0. This confidence score can be utilized to filter out uncertain predictions and is incorporated into the evaluation process. The default confidence is set to 1.0, indicating full confidence in the model's prediction.

3 Competition Overview

The competition occurred between July and November of 2020, encompassing training, validation, and testing phases. A total of 20 teams registered for the competition, and among these, 9 teams participated through the end of the testing phase. The teams were allowed 100 submissions in the first two phases, and a maximum of 10 submissions for the final phase, with daily submission limits. The EvalAI platform was used for hosting the competition.

4 Data

4.1 NL2Bash

The NL2Bash dataset was utilized, consisting of around 10,000 pairs of natural language descriptions paired with corresponding command line syntax.

.

4.2 Tellina Query Log

Around 1000 natural language utterances recorded from user interactions with the Tellina system was collected. Three programmers with Bash experience annotated these, resulting in multiple ground truth labels for many examples in the dataset.

4.3 NLC2CMD Data Collection Track

A parallel data-collection track was included in the competition, collecting natural language to bash command pairs through a web interface on the competition website. 21 participants from industry and academia submitted over 120 examples, which after being filtered, were part of the final phase of the challenge.

4.4 Data partitions and pipeline

The data was filtered for each data sample through a Bash parser to ensure that only valid Bash commands were included. Any text that was not a valid Bash command or used utilities not in the Ubuntu 18.04 LTS command set was removed. For training, participants were provided with a filtered version of the NL2Bash dataset, as well as man pages for Ubuntu 18.04 LTS. In addition, participants were allowed to use any other publicly available data for training. The data set was split into training, validation and test sets with different sizes for each. In addition to the original utilities of the first phase of the competition, 27 additional utilities were added in subsequent phases.

5 Metrics

The submissions to the NLC2CMD competition were assessed based on two primary metrics: accuracy and energy consumption. This approach was utilized to better evaluate submitted solutions.

5.1 Accuracy

This section discusses the metrics used to evaluate the task of translating natural language to Bash code. Existing metrics such as Full Command Accuracy, BLEU score, and Template Accuracy, are reviewed and it is found that they all have shortcomings. The paper presents a metric, verification by execution, which is able to solve these problems. Finally, the metric that was proposed for the competition is discussed in depth.

5.1.1 Existing Metrics

Full Command Accuracy is a metric that measures the exact match between a generated code and a reference code. BLEU scores computes the n-grams of candidate translations with the n-grams of the reference translation. Template Accuracy measures if the command templates match but not exact arguments of the command.

5.1.2 Verification by Execution

Because Bash is a Turing complete language, the equivalence of two commands is undecidable. To handle this issue, the execution of predicted and reference commands is compared to determine accuracy.

5.1.3 NLC2CMD Metric

This paper presents a metric that ignores the arguments in the predicted commands, considers the order of utilities in piped commands and penalizes excess flags.

$S_i^F(C_{pred}, C_{ref}) = 2 * \frac{|F(U(C_{pred})_i) \cap F(U(C_{ref})_i)|}{|F(U(C_{pred})_i) \cup F(U(C_{ref})_i)|}$

$S(p) = max_{C_{ref}} \frac{1}{T} \sum_{i=1}^{T} I[U(C_{pred})_i == U(C_{ref})_i] * S_i^F(C_{pred}, C_{ref})$

The overall score is then computed as follows:

$Score(A(nlc)) = \{ m\ ax_{p \in A(nlc)} S(p), if \exists p \in A(nlc) such that S(p) > 0$
$avg_{p \in A(nlc)} S(p), otherwise$

This metric encourages the correct utilities and their flags, weighted by the algorithm's reported confidence. This metric was chosen for the competition due to the constraints of a conference setting and the need to focus on the core aspects of command synthesis.

5.2 Energy Efficiency

This section discusses the metric of energy efficiency of models, and its relevance in the current research environment. The energy consumption of machine learning models is an area of focus, with the deployment of these models, their inference phase energy consumption can outweigh their training cost over time. The experiment-impact-tracker library was used to measure the energy consumption of submitted solutions.

6 Competing Solutions

The final leaderboard of the NLC2CMD competition consisted of 6 teams/entries, along with 2 baselines. The leaderboard included the accuracy score, energy consumption and latency of the models.

Table 1: Final leaderboard for the NLC2CMD competition, showing the accuracy score for the final (test) phase, along with the energy consumed and latency for every invocation.

Team Name	Accuracy Score	Energy (Joules)	Latency (sec)
Magnum	0.532	0.484	0.709
Hubris	0.513	12.037	14.870
Jb	0.499	2.604	3.142
AICore	0.489	0.252	0.423
AINixCLAISimple	0.429	N.A.	0.010
coinse-team	0.163	343.577	0.452
Tellina	0.138	2.969	3.242

6.1 TF/IDF and Proposed New Baselines

The team AINixCLAISimple developed several simple baselines for the task. The approach that was most successful used an information retrieval (IR) method based on Tf-IDF rankings. Several variations of this method were tested, with the addition of the AInix Archie data, pruning duplicates, normalizing NL tokens and adjusting the confidence.

Table 2: Results from simple IR baselines. Additions to the raw predictor are retained cumulatively top- to-bottom.

IR-Baseline Variation	Accuracy Score
Tf-IDF Raw	0.361
+ AInix Data	0.404
+ Prune Duplicates	0.413
+ Normalize NL	0.429
+ Adjust Conf.	0.472

6.2 Transformer with Beam Search

Team Magnum reached an accuracy score of 0.532 using an ensemble of 5 separately-trained transformer models. Key strategies used in their approach include: Replacing command parameters with generic tokenizations, producing scores using an approximation for confidence, and testing different combinations of encoders and decoders.

6.3 Fine-tuned GPT-2 in Ensemble

The team Hubris fine-tuned pre-trained transformer models, specifically, the GPT-2 architecture. The NL2Bash dataset was also augmented with heuristically mined data from stack-overflow questions. Two models of different sizes and pre-training were used, and the final commands were selected by a heuristic algorithm that maximized the minimal word distance between the commands.

6.4 Multi-Step Pipelines

The multi-step approach involves combining two different models for two separate steps. The first step involves predicting the best utility, and the second step involves predicting the correct flags to use. This can be seen in the models of team jb and team coinse.

7 Discussion

This section summarizes lessons learned and discussions with participants during the competition.

7.1 Metrics Revision

This section discusses suggested alternatives for accuracy and energy measurements.

7.1.1 Suggested Alternatives for Accuracy Measurement

Some suggestions for future metrics include: a metric that measures semantic match instead of exact command matching; restricting the range of commands covered; a metric that measures mean reciprocal rank; a metric that measures session scores over multiple interactions instead of one; using adaptability of algorithms; making fast retraining available; and calibration of penalties. The issues of statefulness of commands, command injection, full text match and underdetermined invocations are also reviewed.

7.1.2 Suggested Alternatives for Energy Measurement

The issues with power measurement, such as reducing computation to lower peak consumption are discussed. It is stated that measurement of total energy consumption may be a better solution. It is argued whether there is even any point to measuring energy at all due to how small the amount of energy is consumed.

7.2 Other Enhancements

Other enhancements include communication of explanations to users by converting commands back to natural language, and conversational interfaces to allow for more context for the system.

8 Conclusion

In this paper, the NLC2CMD competition is discussed, including the methodology, data used and the metrics of the competition. Going forward, the feedback received will be incorporated in future iterations of the competition.