INSTITUT FÜR INFORMATIK
Datenbanken und Informationssysteme

Universitätsstr. 1 D–40225 Düsseldorf

Heinrich Heine
Universität
Düsseldorf

Context-Aware CLI autocompletion

Roger Sellin

Bachelorarbeit

Date of issue:	7. june 2023
Date of submission:	7. September 2023
Reviewers:	Dr. Konrad Völkel
	Prof. Dr. Stefan Conrad

Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Düsseldorf, den 7. September 2023

Roger Sellin

Abstract

Stochastic Autocompletion systems for the terminal are nothing new, but the recent development of large-language-models(LLMs) allows for new approaches. These LLMs integrate world knowledge, leading to a potential spillover effect that can enhance autocompletion systems.

While the completion of CLI (Command Line Interface) commands is possible, the intriguing question is whether using the path of command execution and the files contained in the current directory as additional context can improve the accuracy of completion predictions.

For instance, the likelihood of using a git command in a Git repository is higher than in a downloads folder. Furthermore, a command beginning with 'git commit' is more likely to be succeeded by 'git push' than 'git pull'.

The specific prompt can influence the outcome strongly. But an indepth look into this would exceed the scope of this thesis. So the variety is only tested with a few examples.

In terms of implementation, a local solution for autocompletion is preferred. This allows the system to operate effectively on encapsulated systems without needing internet access. Additionally, the computational power required should be considered. An autocompletion system containing an LLM that necessitates a high-end computer, although technically possible, would nullify its practicality.

Contents

1 The Model **1**

1.1 rejected models 1

1.2 why not LLama? 1

1.3 Alpaka 1

2 tecnical background **2**

3 setup **3**

3.1 tests with gpt2 4

3.2 test with alpaca-7b 4

4 possible extensions **5**

5 Sources **5**

List of Figures **6**

List of Tables **6**

1 The Model

The Training of an LLM from scratch would be to cost/time intensive for this thesis so the use of a pretrained LLM is far more practical.

1.1 rejected models

1.1.1 BERT

While BERT and BERT based model are small in size their word based approach limits its capability to complete words. Which makes it inadequit to for the task. This can be mitigated as loong as the word is known to the tokenizer.

1.1.2 GPT-3.5/GPT-4

While gpt based models as of now are considered to be the best LLMs. Their size and needed computational power to operate them make the impractibel for consumergrade computers. This can be cirkumvented by the use of the opeaiAPI but this is not free of charge and needs to sent data over the internet which prohibits its use with sensitive data and cannot be used on capsueld systems.

1.2 why not LLama?

(i need to answere this question)

1.3 Alpaka

Stanfords Alpacka Model based on Metas LLama Model semes to be a good candidate. We will use the Alpaca-7B version because it is the smalest.

2 tecnical background

Since the training of LLMs would take to much time and be very expensive.Therfore not feasable. We just have to use pretrained models.

The models we use here are all transformerbased LLMs the reason for this is that the transformerarchitektur is the is the most powerful known architectur for language models and the defacto standart for all leading models

The final limitation given the task of autocomletion is that we need a tokenbased model. A word based model would not suffice because these models are not abel to autocomplete words just sentences. The ability to complete words is substancial for the task at hand.

3 setup

i used a Miniconda 3.1 enviroment

- `transformers-4.30.2-pyhd8ed1ab_1.conda`
- `tokenizers-0.13.3-py38h7d131c9_0.conda`

list of commands used

I used the following commands to autocomplete: "sudo apt","sudo apt up","sudo apt in","ls","py","pyt","pyth","pytho","git","git i","git in","git ini","git co","git comm"

It contains apt, python and git commands

list prompt variations

file$_{c}ontexts$ = $[$"$Therearethefollowingfilesinthecurrentdirectory$: ","$Files$: ","$Thesefilesareinthisdirectory$: "$]$

premises = ["You are an autocomplete function. ","This is a linux terminal. ","This is a linux terminal command. ","This is an autocomplete function. ",""]

order = ["Autocomplete the following linux terminal command and provide no further explanation for the command: "]

A number of prompt combinations have been tested and for simplicity we decided to choose the variation with the best outcome

the "You are an autocomplete function. " and "This is an autocomplete function. " tend to provide significantly worse results than other premises. There are less likely to produce a terminal command, but a text about said command.

"This is a linux terminal command. " provides better output but tends to append an explanation of the command. "This is a linux terminal. " tends to provide the best solutions.

Path and file in the current directory are not specified in the final prompt because these are defined by the context.

The file contexts show no significant differences, so "There are the following files in the current directory: "is chosen to pick one for simplicity.

So the final prompt is: "This is a linux terminal. There are the following files in the current directory: <files>,Path: <path>, Autocomplete the following linux terminal command and provide no further explanation for the command: <command>".

3.1 tests with gpt2

The unfinetuned gpt2 was chosen for its small size of round 550mb. Which makes it runnable on a low resource machine without any further processing.

We used it with the Huggingface API since it is easy to use and the most popular API for such tasks. The "This is a linux terminal. There are the following files in the current directory: <files>,Path: <path>, Autocomplete the following linux terminal command and provide no further explanation for the command: <command>" prompt as decided on in the previous section was used.

The model produced no valid commands while some of the git completions resembled some git commands, none of them were valid.

While the speed was sufficient, its failure to produce valid commands suggests that

it is probably better to use a different model.

I tested two scenarios with the gpt2 model first the apt based commands that worked insofar that it predicted the most commonly used apt commands update and install also with a lot of followup text, however most apt based commands are independent from their context insofar that they don't rely on the path they are executed in. It also has to be noted that the model produced more text appended to the command

The git commands had way worse results. While trying to use the git init command given the "git ini" was not able to predict the "t" of "init" correctly, and the "git c" wasn't able to predict anything near "git commit" not even the "git comm" could predict "commit" most of the time even then not without a lot of unwanted text appended. The prepending of "Autocomplete the following terminal command:" seems to produce less unwanted text but still no usable output.

3.2 test with alpaca-7b

While alpaca gives far better results, there are limits.

A test without any prefix showed only nonsensical results.

The apt commands often get completed into a text about apt based commands. While the ls command doesn't get interpreted as a command, presumably because it is short.

The python based commands mostly get completed into nonsensical text. Sometimes with python commands with more letters of the word "python" gets interpreted as a type of snake, which is not surprising considering the training data probably contains more text about snakes than the programming language. The git based commands again get completed into nonsensical text.

The added prefix "This is a linux terminal. There are the following files in the current directory:,Path:, Autocomplete the following linux terminal command and provide no further explanation for the command: <command>" shows an improvement, with apt and ls commands, with the notable outlier "py" who gets completed to the misspelled "pyhton" following a text about python.

The git based commands get completed in to nonfunctional git commands and text about

these commands.

If we add a path and files in the current directory as context, we still cannot produce valid commands.

Noteworthy here is that sometimes the python commands dont get any completion at all.

This suggests that the base model is not able to do it in a sufficient way. Which makes methods like Lora not viable to counter its slow speed and big size. Since it would lower its accuracy.

Therefore fine-tuning the model is the next logical step.

4 possible extensions

It could be interesting to investigate if and how previous commands influence the outcome. Commands from the history could be added to the prompt. Although since this could be a high amount of tokens, the computational power needed therefore would be higher and the token limit of the model could be exceeded. Therefore, the commands need to be filtered.

A different model could also be tested. While this was written, LLama2 was released by Meta. It is a successor of LLama which alpaca is based on. It was not used solely for the reason that it came out late in the writing of this thesis.

5 Sources

List of Figures

List of Tables