

From Data Engineering to Prompt Engineering

Solving data preparation tasks with ChatGPT



Photo by [Ricardo Gomez Angel](#) on [Unsplash](#)

Data engineering makes up a large part of the data science process. In CRISP-DM this process stage is called “data preparation”. It comprises tasks such as data ingestion, data transformation and data quality assurance. In our article we solve typical data engineering tasks using ChatGPT and Python. By doing so, we explore the link between data engineering and the new discipline of prompt engineering.

Introduction

In May 2022, Stephen Wolfram and Lex Fridman gave an insightful talk titled “[Is programming dead?](#)”. They discussed whether high-level languages will still be used by developers in future. According to Wolfram, many programming tasks can be automated with large language models (LLMs). At the time of this writing, the most prominent example of such a model is [ChatGPT](#). Since its introduction in late 2022, it has generated astonishing results. Specifying an action to be performed by an LLM is referred to as “prompt engineering”. If Wolfram is right, at least part of software development will shift from writing code to writing prompts.

When it comes to data science, data preparation can be a time-consuming and tedious task. So why not try to automate it with an LLM? In the following sections we tackle different data engineering problems with ChatGPT and Python. Instead of writing the Python code ourselves, we used prompt engineering to generate it. Our experiment was conducted on 19 May 2023 based on the latest freely available ChatGPT version (GPT-3.5) at the time.

Creating a data frame

Let's start with a simple problem and create a Pandas data frame from a sample dataset. Table 1 contains national indicators as provided for instance by the [World Bank](#).

Continent	Country	Year	Category	Variable	Value
Asia	China	2022	Economy	GDP	18.1
Asia	China	2022	Demographics	Population	1412
North America	USA	2022	Economy	GDP	25.5
North America	USA	2022	Demographics	Population	333
Europe	Germany	2022	Economy	GDP	3.9
Europe	Germany	2022	Demographics	Population	84

Table 1: Dataset with national indicators

In order to create a data frame from the sample above, we started a new conversation with ChatGPT and posted the following prompt:

```
Create a Pandas table with the following data:  
Continent, Country, Year, Category, Variable, Value  
Asia, China, 2022, Economy, GDP, 18.1  
Asia, China, 2022, Demographics, Population, 1412  
North America, USA, 2022, Economy, GDP, 25.5  
North America, USA, 2022, Demographics, Population, 333  
Europe, Germany, 2022, Economy, GDP, 3.9  
Europe, Germany, 2022, Demographics, Population, 84
```

Prompt 1: Creating a data frame

In response, the model generated the Python code below:

```
import pandas as pd  
  
data = {  
    'Continent': ['Asia', 'Asia', 'North America', 'North America',  
                 'Europe', 'Europe'],  
    'Country': ['China', 'China', 'USA', 'USA', 'Germany', 'Germany'],  
    'Year': [2022, 2022, 2022, 2022, 2022, 2022],  
    'Category': ['Economy', 'Demographics', 'Economy', 'Demographics',  
                'Economy', 'Demographics'],  
    'Variable': ['GDP', 'Population', 'GDP', 'Population', 'GDP',  
                'Population'],  
    'Value': [18.1, 1412, 25.5, 333, 3.9, 84]  
}  
  
df = pd.DataFrame(data)  
print(df)
```

Response 1: Code for data frame creation

When running the script, we see that the output matches our specification:

```

      Continent Country Year Category Variable Value
0      Asia     China 2022   Economy     GDP    18.1
1      Asia     China 2022 Demographics Population 1412.0
2 North America     USA 2022   Economy     GDP    25.5
3 North America     USA 2022 Demographics Population 333.0
4      Europe   Germany 2022   Economy     GDP     3.9
5      Europe   Germany 2022 Demographics Population 84.0

```

Even though we used a sample for our experiment, the code generated in the next sections can in principle also handle bigger datasets.

Pivoting the data frame

For data science, the table structure we created with our first prompt is not optimal. Instead, a so-called “flat table” would be more convenient. In Table 1, the indicators “GDP” and “Population” are represented by two columns named “Variable” and “Value”. In contrast, a flat table would contain separate columns for each indicator, as shown in Table 2.

Continent	Country	Year	GDP	Population
Asia	China	2022	18.1	1412
North America	USA	2022	25.5	333
Europe	Germany	2022	3.9	84

Table 2: Flat table with national indicators

Converting the format of Table 1 into the format of Table 2 is called “pivoting”. Let us try to solve this task with ChatGPT. To this end, we posted a second prompt in our conversation:

```

Transform the table into the following format:
Continent, Country, Year, GDP, Population
Asia, China, 2022, 18.1, 1412
North America, USA, 2022, 25.5, 333
Europe, Germany, 2022, 3.9, 84

```

Prompt 2: Pivoting the data frame

The above message is based on an example of the required data format. Alternatively, we could have formulated our request in a purely verbal way. As with classical programming, there are alternative solutions to achieve a goal in prompt engineering. Finding the optimal way will be an exciting task for the near future. As a result of our prompt, ChatGPT adjusted the Python code as follows:

```

import pandas as pd

data = {
    'Continent': ['Asia', 'Asia', 'North America', 'North America',

```

```

    'Europe', 'Europe'],
    'Country': ['China', 'China', 'USA', 'USA', 'Germany', 'Germany'],
    'Year': [2022, 2022, 2022, 2022, 2022, 2022],
    'Category': ['Economy', 'Demographics', 'Economy', 'Demographics',
    'Economy', 'Demographics'],
    'Variable': ['GDP', 'Population', 'GDP', 'Population', 'GDP',
    'Population'],
    'Value': [18.1, 1412, 25.5, 333, 3.9, 84]
}

df = pd.DataFrame(data)

# Pivot the table
pivot_df = df.pivot_table(index=['Continent', 'Country', 'Year'],
                           columns='Category', values='Value').reset_index()

# Rename the columns
pivot_df.columns.name = None
pivot_df.columns = [column if column != 'Variable' else None for column in
                    pivot_df.columns]

print(pivot_df)

```

Response 2: Pivoting code

However, running the second version of the script reveals that the output is not what we had requested:

	Continent	Country	Year	Demographics	Economy
0	Asia	China	2022	1412.0	18.1
1	Europe	Germany	2022	84.0	3.9
2	North America	USA	2022	333.0	25.5

Instead of the expected columns “GDP” and “Population”, the table contains two columns named “Demographics” and “Economy”. To fix this problem, we posted the next prompt:

```

The columns are not correct, you should have used
the variables GDP and population

```

Prompt 3: Fix for the pivoting code

In response, the model corrected the mistake and generated another version of the Python code:

```

import pandas as pd

data = {
    'Continent': ['Asia', 'Asia', 'North America', 'North America',
    'Europe', 'Europe'],
    'Country': ['China', 'China', 'USA', 'USA', 'Germany', 'Germany'],
    'Year': [2022, 2022, 2022, 2022, 2022, 2022],
    'Category': ['Economy', 'Demographics', 'Economy', 'Demographics',
    'Economy', 'Demographics'],
    'Variable': ['GDP', 'Population', 'GDP', 'Population', 'GDP',
    'Population'],

```

```

    'Population'],
    'Value': [18.1, 1412, 25.5, 333, 3.9, 84]
}

df = pd.DataFrame(data)

# Pivot the table
pivot_df = df.pivot_table(index=['Continent', 'Country', 'Year'],
                           columns='Variable', values='Value').reset_index()

# Rename the columns
pivot_df.columns.name = None
pivot_df.columns = ['Continent', 'Country', 'Year', 'GDP', 'Population']

print(pivot_df)

```

Response 3: Fixed pivoting code

Running the script reveals that the pivoting was implemented correctly this time:

	Continent	Country	Year	GDP	Population
0	Asia	China	2022	18.1	1412.0
1	Europe	Germany	2022	3.9	84.0
2	North America	USA	2022	25.5	333.0

Enriching the data frame

Another typical data engineering task is to enrich a dataset with additional information. For this purpose, we asked ChatGPT to add [ISO country codes](#) to our data by posting the prompt below:

```
For each country, in the table above, add its ISO code
```

Prompt 4: Adding country codes

As a result, the model adjusted the response and returned the following code snippet:

```

import pandas as pd

data = {
    'Continent': ['Asia', 'Asia', 'North America', 'North America',
                 'Europe', 'Europe'],
    'Country': ['China', 'China', 'USA', 'USA', 'Germany', 'Germany'],
    'Year': [2022, 2022, 2022, 2022, 2022, 2022],
    'Category': ['Economy', 'Demographics', 'Economy', 'Demographics',
                 'Economy', 'Demographics'],
    'Variable': ['GDP', 'Population', 'GDP', 'Population', 'GDP',
                 'Population'],
    'Value': [18.1, 1412, 25.5, 333, 3.9, 84]
}

iso_codes = {
    'China': 'CN',
    'USA': 'US',

```

```

    'Germany': 'DE'
}

df = pd.DataFrame(data)

# Add ISO code column
df['ISO Code'] = df['Country'].map(iso_codes)

# Pivot the table
pivot_df = df.pivot_table(index=['Continent', 'Country',
                                'ISO Code', 'Year'],
                          columns='Variable', values='Value').reset_index()

# Rename the columns
pivot_df.columns.name = None
pivot_df.columns = ['Continent', 'Country', 'ISO Code',
                   'Year', 'GDP', 'Population']

print(pivot_df)

```

Response 4: Data frame with country codes

Running the script shows that a new column with ISO codes had been added to the data frame.

	Continent	Country	ISO Code	Year	GDP	Population
0	Asia	China	CN	2022	18.1	1412.0
1	Europe	Germany	DE	2022	3.9	84.0
2	North America	USA	US	2022	25.5	333.0

Deriving a column

Let us next derive a new column “GDP per capita” by dividing the GDP of a country by its population size. To do so, we have sent this prompt:

```
Derive a new column "GDP per capita" by dividing the GDP
of a country by its population size
```

Prompt 5: Deriving a new column

In response, the model revised the code as follows:

```

import pandas as pd

data = {
    'Continent': ['Asia', 'Asia', 'North America', 'North America',
                 'Europe', 'Europe'],
    'Country': ['China', 'China', 'USA', 'USA', 'Germany', 'Germany'],
    'Year': [2022, 2022, 2022, 2022, 2022, 2022],
    'Category': ['Economy', 'Demographics', 'Economy', 'Demographics',
                'Economy', 'Demographics'],
    'Variable': ['GDP', 'Population', 'GDP', 'Population', 'GDP',
                'Population'],
    'Value': [18.1, 1412, 25.5, 333, 3.9, 84]
}

```

```

}

iso_codes = {
    'China': 'CN',
    'USA': 'US',
    'Germany': 'DE'
}

df = pd.DataFrame(data)

# Add ISO code column
df['ISO Code'] = df['Country'].map(iso_codes)

# Pivot the table
pivot_df = df.pivot_table(index=['Continent', 'Country',
                                'ISO Code', 'Year'],
                          columns='Variable', values='Value').reset_index()

# Calculate GDP per capita
pivot_df['GDP per capita'] = pivot_df['GDP'] / pivot_df['Population']

# Rename the columns
pivot_df.columns.name = None
pivot_df.columns = ['Continent', 'Country', 'ISO Code', 'Year', 'GDP',
                    'Population', 'GDP per capita']

print(pivot_df)

```

Response 5: Data frame with new column

The output reveals that the script produces the expected result:

	Continent	Country	ISO Code	Year	GDP	Population	GDP per capita
0	Asia	China	CN	2022	18.1	1412.0	0.012819
1	Europe	Germany	DE	2022	3.9	84.0	0.046429
2	North America	USA	US	2022	25.5	333.0	0.076577

Analyzing the data

Once the data transformations are completed, the actual analysis can begin. Let us finally calculate the correlation coefficient between the GDP and the population of the countries. To this end, we posted a last prompt:

```
Use the table above to calculate the correlation coefficient between
the GDP and the population of the countries
```

Prompt 6: Analyzing correlation

As a result, the model returned the code fragment below:

```
import pandas as pd
```

```
data = {
    'Continent': ['Asia', 'Asia', 'North America', 'North America',
                 'Europe', 'Europe'],
    'Country': ['China', 'China', 'USA', 'USA', 'Germany', 'Germany'],
    'Year': [2022, 2022, 2022, 2022, 2022, 2022],
    'Category': ['Economy', 'Demographics', 'Economy', 'Demographics',
                 'Economy', 'Demographics'],
    'Variable': ['GDP', 'Population', 'GDP', 'Population', 'GDP',
                 'Population'],
    'Value': [18.1, 1412, 25.5, 333, 3.9, 84]
}

df = pd.DataFrame(data)

# Pivot the table
pivot_df = df.pivot_table(index=['Continent', 'Country', 'Year'],
                           columns='Variable', values='Value').reset_index()

# Calculate correlation coefficient
correlation_coefficient = pivot_df['GDP'].corr(pivot_df['Population'])

print("Correlation Coefficient: {:.3f}".format(correlation_coefficient))
```

Response 6: Code to analyze correlation

When running the script, we find a positive correlation between the two variables, which is not a surprise:

```
Correlation Coefficient: 0.350
```

Conclusion

Admittedly, our experiment was based on a simple dataset. Nevertheless, the results are remarkable. We performed several data engineering tasks without writing a single line of code. ChatGPT was not only able to correctly implement our prompts in most cases. But even when the model made a mistake, it was capable of reflecting and fixing it. As always in software development, generated code must be tested. Moreover, it may need refactoring and optimization. Using pylint is still a good idea in the age of AI. Summarizing, however, we have to agree with Wolfram: In future, a significant part of data engineering will shift from coding to prompt engineering. This new approach will not replace data engineers, but will make them more efficient.

All images unless otherwise noted are by the authors. In the PDF version, source code may have been reformatted for layout reasons.

About the authors

Christian Koch is an Enterprise Architect at BWI GmbH and Lecturer at the Nuremberg Institute of Technology Georg Simon Ohm.

Markus Stadi is a Senior Cloud Data Engineer at Dehn SE working in the field of Data Engineering, Data Science and Data Analytics for many years.

Lukas Berle is a Data Architect at TeamBank AG specialized in the design and implementation of robust data analytics architectures.