

To initially test the capabilities of the local LLM, a simple programming task was selected to compare the lines of text in two files. The contents of the two text files were as follows:

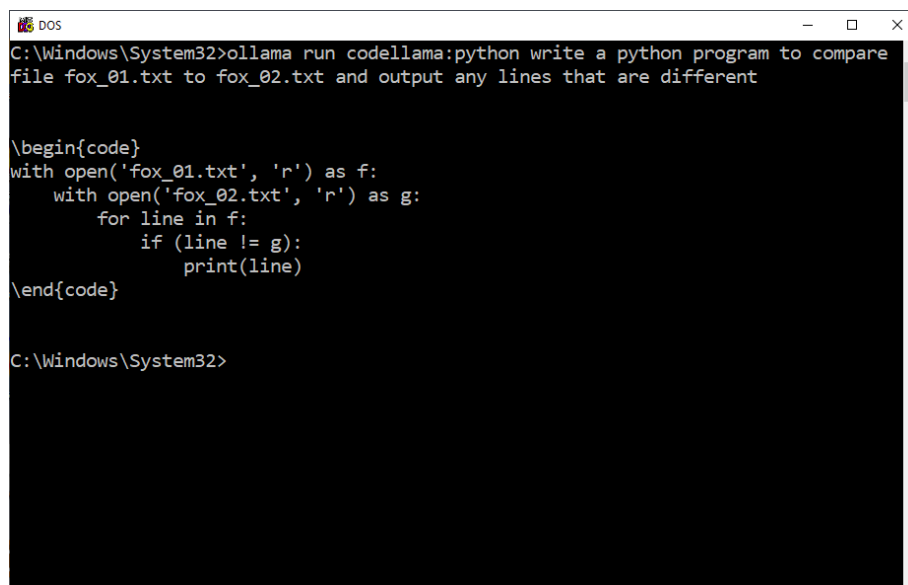
fox_01.txt

The quick brown fox
jumped over the lazy dog

fox_02.txt

The quick brown fox
jumped over the extremely lazy dog

Specifically, the command entered into the terminal was: *ollama run codellama:python write a python program to compare file fox_01.txt to fox_02.txt and output any lines that are different.* Processing this command took only two minutes and resulted in the Python code shown in Figure 2.



```
C:\Windows\System32>ollama run codellama:python write a python program to compare
file fox_01.txt to fox_02.txt and output any lines that are different

\begin{code}
with open('fox_01.txt', 'r') as f:
    with open('fox_02.txt', 'r') as g:
        for line in f:
            if (line != g):
                print(line)
\end{code}

C:\Windows\System32>
```

Figure 2. Python code generated by the local LLM

Copying and pasting the Python code generated into a text file named fox.py allowed this file to be opened and run in the PyCharm integrated development environment. The results are displayed in Figure 3.



```
Run Difffox x
C:\Users\toaster\PycharmProjects\FoxDiff\.venv\Scripts\python.exe D:\Difffox.py
jumped over the lazy dog
Process finished with exit code 0
```

Figure 3. Output from the Python program generated by the LLM

The results were totally consistent with the Python code that had been generated although not entirely so with the precise nature of the task set for the LLM model. The code specifies that if a line in file_01.txt is not present in file_02.txt then this line is to be displayed. Hence we see “jumped over the lazy dog”, the line in the first file that is different to the line “jumped over the extremely lazy dog” that is contained in the second text file. However, the command to generate the code was “...output any lines that are different” [note the plural “lines”], implying that any lines in either file that are different should be output.

Indeed, when the same programming task was repeated in ChatGPT, a much more comprehensive program was returned (see Figure 4). This code opens both files and copies their contents into arrays. The program then loops through the lines of the two files, using the maximum number of lines in either array as a boundary condition.. If a line in one file is identified as being different from that in the other file then both associated lines are displayed. So, for this program, both of the lines “jumped over the lazy dog” and “jumped over the extremely lazy dog” are included in the output (Figure 5).

```
python Copy code

def compare_files(file1, file2):
    with open(file1, 'r') as f1, open(file2, 'r') as f2:
        lines1 = f1.readlines()
        lines2 = f2.readlines()

    # Find the maximum number of lines in the files
    max_lines = max(len(lines1), len(lines2))

    # Compare lines
    for i in range(max_lines):
        line1 = lines1[i].strip() if i < len(lines1) else ''
        line2 = lines2[i].strip() if i < len(lines2) else ''

        if line1 != line2:
            print(f"Difference at line {i + 1}:")
            print(f"File 1: {line1}")
            print(f"File 2: {line2}")
            print()

# Replace 'fox_01.txt' and 'fox_02.txt' with your file paths if needed
compare_files('fox_01.txt', 'fox_02.txt')
```

Figure 4. Python program generated by ChatGPT

```
Run DiffFox_ChatGPT x
C:\Users\toaster\PycharmProjects\FoxDiff\.venv\Scripts\python.exe D:\DiffFox_ChatGPT.py
Difference at line 2:
File 1: jumped over the lazy dog
File 2: jumped over the extremely lazy dog
Process finished with exit code 0
```

Figure 5. Output from the ChatGPT Python program

So, what are my conclusions from this test of a Python-specific large language model running on a PC? Firstly, Ollama is really easy to install and similarly is easy to use once the terminal commands are understood. Typing *ollama help* in the terminal produces a list of the available commands but there are also many how-to tutorials and on-line videos available as additional resources. In addition, running a LLM locally is slower than using the web-based counterparts but even the small 7B model is capable of producing useful code.

Further testing of the model identified some instances of “semi-hallucination” where the module suggested importing a Python library but failed to provide any code to support its use, and also indicated the alternative use of the “diff” command that is only available for Linux and not for Windows.

I didn’t attempt to produce a complicated Python program that would likely require a very detailed and complex set of instructions. In fact, this might be better accomplished by generating specific code snippets using the local LLM and subsequently combining these into a complete program. Nevertheless, the current generation of local LLM’s shows promise and is most probably the precursor to a much faster and more powerful system at some point in the future.

Bottom Line:

Ollama (Open Source)
Version 0.3.9
<https://ollama.com>

