

Tipps für zielführende Prompts bei der Entwicklung einer App für eine programmierbare Smartwatch

Large Language Models (LLM), die euch beim Programmieren helfen sollen, erzeugen eine Ausgabe (euer Programm), indem sie eine Eingabe (Prompt) durch komplexe Berechnungen mit vorberechneten Parametern verarbeiten (das Modell). Dies ist keine allgemeine Einleitung in das Thema „Programmieren mit LLM-Unterstützung“, sondern enthält konkrete Tipps, die sich konkret im vorgestellten Projekt beim Arbeiten mit LLMs und programmierbaren Smartwatches als hilfreich erwiesen haben.

- Der Prompt sollte detailliert sein. Wenn dem LLM Informationen fehlen, wird es das wahrscheinlichste annehmen, um trotzdem eine Ausgabe zu erzeugen. Oft, aber nicht immer, sind diese Annahmen sogar zweckmäßig. Dennoch erzeugt ein detaillierter Prompt mit viel höherer Wahrscheinlichkeit das gewünschte Programm.
 - Der erste Prompt wird nur selten zum gewünschten Ergebnis führen. Hier ist eine Liste mit Fragen, die helfen können, einen geeigneten Prompt zum Bugfixing zu schreiben. Nicht in jedem Fall müssen alle Fragen genutzt werden.
1. Grobe Einordnung, was verbessert werden soll (z.B. Bedienung im Screen1).
 2. Wie ist das aktuelle Verhalten des Programms? Gibt es Fehlermeldungen und wenn ja, welche?
 3. Was soll passieren? Hier kommt es auf sprachliche Genauigkeit an. „Die App soll stoppen“ ist missverständlich. Dies könnte meinen, dass die App geschlossen werden soll, der Startzustand wiederhergestellt werden soll, eine Auswertung oder Abbruchmeldung ausgegeben werden oder nur ein Timer gestoppt werden soll.
 4. Wie genau soll das Verhalten ausgelöst werden?
 - Was ist der Auslöser: Touch (wo?), Button (wie lange?), Messwert (welcher Wert? ein einzelner Messwert oder ein Mittelwert o.ä.), Zeit (wann?),....
 5. Wann soll es funktionieren?
 - Soll das Verhalten nur auf einem bestimmten Screen, oder unter bestimmten Bedingungen auftauchen, während die App geöffnet ist oder auch im Hintergrund?
 6. Wie soll es aussehen?
 - Welcher Text oder welches Symbol soll gezeigt werden?
 - Gebt konkrete Werte z.B. für die Position und Farben.
 - Nutzt auch gerne Zeichnungen. Diese werden gelegentlich missverstanden, können aber in Kombination mit mehreren Korrekturprompts hilfreich sein.

Beispiel: „Auf dem Stoppuhrscreen läuft der Timer immer weiter, obwohl der Button 1 gedrückt wird. Der Timer sollte aber beim Drücken von Button 1 stoppen und die gestoppte Zeit auf dem Auswertungsscreen angezeigt werden.“

Weitere Tipps

- In unserem LLM tauchten bestimmte Fehler immer wieder auf. Zum Beispiel war die Farbe, die offensichtlich erzeugt werden sollte, invertiert. Das heißt, obwohl das Modell in den Kommentaren ausgab, es handle sich um schwarzen Text, war der Text weiß und auf dem weißen Hintergrund nicht zu sehen. Dieser scheinbar simple Fehler ließ sich auch durch Korrekturprompts kaum beheben.
- Große Probleme hatte das LLM damit, Bedienelemente sinnvoll auf dem Screen anzuordnen. Oft wurde der vorhandene Platz nicht gut genutzt oder Elemente waren „abgeschnitten“. Hier war es zweckmäßig, die Zahlenwerte mit den Koordinaten der Elemente manuell anzupassen.
- Wurde eine Funktion falsch verwendet, in dem zum Beispiel zwei Argumente eingegeben wurden, aber nur eines akzeptiert wird, dann genügte es oft, die Fehlermeldung und die Dokumentation dieser Funktion in den Korrekturprompt einzufügen.
- LLM sind sehr gut darin, Annahmen zu treffen, wenn der Nutzer keine genauen Angaben macht, da sie so entwickelt wurden, möglichst immer eine Antwort zu erzeugen, die den Nutzer zufriedenstellt. Der Nutzer ist schließlich ein Kunde und möchte vielleicht nicht gesagt bekommen, dass sein Prompt ungenau formuliert ist. Oft geschieht es dann, dass man versucht, dem LLM diese Annahmen wieder auszureden und dabei gar nicht mehr dazu kommt, zu beschreiben, wie man es eigentlich haben wollte. Besser ist es, den Chat direkt mit einem möglichst genauen und ausführlichen Prompt zu beginnen. Manchmal kann es daher von Vorteil sein, in einem neuen Chat zu beginnen, wenn man merkt, dass man sich verrannt hat.