OKIEM EKSPERTA

Helion

Programowanie wspomagane AI

Automatyzacja pracy programisty dzięki ChatGPT i GitHub Copilot

Christoffer Noring, Anjali Jain,
Marina Fernandez, Ayşe Mutlu, Ajit Jaokar

packt

Tytuł oryginału: AI-Assisted Programming for Web and Machine Learning: Improve your development workflow with ChatGPT and GitHub Copilot

Tłumaczenie: Grzegorz Werner

ISBN: 978-83-289-2425-3

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/prwsai
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Printed in Poland.

Spis treści

Rozdział 2

Strategia podpowiedzi

Wprowadzenie

W poprzednim rozdziale omówiliśmy historyczny kontekst rozwoju AI z biegiem lat i przejście od **przetwarzania języka naturalnego** (ang. *natural language processing*, **NLP**) do **dużych modeli językowych** (ang. *large language model*, **LLM**) oraz wyjaśniliśmy, że to właśnie LLM-y są modelami uczenia maszynowego, na których opierają się asystenty AI. Aby używać tych asystentów, przekazuje im się dane wejściowe w postaci podpowiedzi (ang. *prompts*) pisanych w języku naturalnym. Żeby jednak skutecznie „podpowiadać" asystentom i osiągać żądane cele, trzeba przyjąć pewną strategię, i o tym właśnie będzie mowa w niniejszym rozdziale.

Efektywne „podpowiadanie" jest znane w branży jako „strategia podpowiedzi" albo „inżynieria podpowiedzi". Nie jest to praktyka inżynierska w typowym sensie tego słowa, ale raczej forma sztuki, w której użytkownicy asystentów AI odkryli wzorce oraz praktyki, które wydają się dobrze działać. My, autorzy tej książki, rozwijamy te odkryte praktyki i chcemy opisać nasze ustalenia w dwóch dziedzinach: tworzenia aplikacji internetowych typu full-stack oraz nauki o danych. Książka jest przeznaczona dla twórców aplikacji internetowych oraz specjalistów data science i wyjaśnia, jak mogą rozwiązywać problemy w swojej dziedzinie z użyciem asystenta AI.

Ten rozdział jest centralnym elementem książki, w tym sensie, że zaprezentowane tu podejście zostanie zilustrowane przykładami w pozostałych rozdziałach. W związku z tym jest rodzajem przewodnika, do którego możesz wielokrotnie wracać, objaśniającego teorię i założenia wykorzystywane w przyszłych rozdziałach, które rozwiązują konkretne problemy w dziedzinie data science i tworzenia aplikacji internetowych.

W tym rozdziale:

- Przedstawimy strategię rozwiązywania problemów z użyciem podpowiedzi i weryfikowania rozwiązania.
- Zilustrujemy strategię przykładami z dziedzin data science i tworzenia aplikacji internetowych.
- Wskażemy kilka podstawowych zasad pisania podpowiedzi.

Na jakim jesteś etapie?

Jako praktyk nauki o danych i (lub) tworzenia aplikacji internetowych typu full-stack znasz swoje rzemiosło. Oznacza to, że nieobce są Ci narzędzia i techniki rozwiązywania problemów. W tym momencie przyglądasz się asystentowi AI i zdajesz sobie sprawę, że jest on sterowany językiem naturalnym, tzw. podpowiedziami. Możesz jednak nie wiedzieć, że nie sprowadza się to do prostego pisania podpowiedzi i otrzymywania odpowiedzi. Asystent AI został wytrenowany na dużym korpusie tekstu, więc jest dość elastyczny w kwestii tego, na jakie tematy może generować tekst i jak reaguje na podpowiedzi. Ze względu na tę elastyczność powinieneś zrozumieć, jak pisać podpowiedzi, które będą działać skutecznie i wydajnie.

Wskazówki dotyczące efektywnego podpowiadania

Podpowiedzi są danymi wejściowymi dla narzędzi AI. W zależności od tego, co chcesz osiągnąć, musisz dostosować podpowiedzi do konkretnego scenariusza. Dlatego sposób „podpowiadania" ma znaczenie. Jeśli na przykład Twoja podpowiedź jest zbyt ogólnikowa, nie uzyskasz żądanego efektu. Albo przypuśćmy, że używasz podpowiedzi do wygenerowania firmowego sloganu; nie wykorzystałbyś tej samej podpowiedzi w celu wygenerowania kodu aplikacji. Natomiast w dyscyplinie takie jak data science jest ważne, żeby wykonywać zadania w pewnej kolejności, więc Twoja podpowiedź powinna odzwierciedlać to, co chcesz zrobić, a w razie potrzeby wskazywać etapy wymagane do wykonania zadania.

Do sukcesu potrzebujesz odpowiedniego podejścia, ogólnej strategii, którą będziesz mógł wykorzystać do efektywnej pracy z asystentami AI. Ponadto strategia ta powinna być na tyle konkretna, żeby obejmowała „najlepsze praktyki" w wybranych dziedzinach problemowych. Jak wspomniano wcześniej w tym rozdziale, opracowaliśmy strategie podpowiedzi specjalnie z myślą o tworzeniu aplikacji internetowych i o data science.

Na ogólniejszym poziomie sugerujemy ogólne wytyczne rozwiązywania problemów z pomocą asystentów AI; uważamy, że są one użyteczne bez względu na dziedzinę, której dotyczą Twoje podpowiedzi:

1. **Przeanalizuj problem, żeby go w pełni zrozumieć**. Może to obejmować kilka etapów, takich jak poniższe:
   - **Zrozum problem**. Niezależnie od problemu, trzeba zrozumieć, czym on jest, a czym nie jest. Na przykład czy budujemy model uczenia maszynowego do przewidywania sprzedaży, czy stronę internetową do śledzenia stanów magazynowych? Są to dwa różne problemy, które wymagają różnych podejść.
   - **Zidentyfikuj części**. Problem jest zwykle skomplikowany i składa się z wielu części, które wymagają osobnego rozwiązania. Jeśli na przykład budujemy model uczenia maszynowego do przewidywania sprzedaży, musimy określić dane, model, trening i ewaluację. Każdą z tych części można podzielić

na mniejsze części itd. Kiedy znajdziesz właściwy poziom szczegółowości dla swojego problemu, możesz zacząć rozwiązywać go przez pisanie podpowiedzi.

- **Podziel problem na mniejsze części**. W razie potrzeby podziel problem na mniejsze, łatwiejsze do opanowania części.
- **Zidentyfikuj i zrozum dane**. Zwłaszcza w uczeniu maszynowym bardzo istotne jest zidentyfikowanie zbioru danych, z którym będziesz pracować, jego zawartości i jego struktury. W tworzeniu aplikacji internetowych dane również odgrywają kluczową rolę, ale celem jest zwykle zagwarantowanie, że będziesz mógł odczytywać, zapisywać i prezentować dane w sposób użyteczny dla użytkownika.

2. **Generuj podpowiedzi na odpowiednim poziomie**. Kiedy w pełni zrozumiesz problem, powinieneś mieć listę zadań i dla każdego z nich móc utworzyć i wykonać podpowiedź, która rozwiązuje to zadanie.
3. **Zweryfikuj rozwiązanie**. Tak jak podczas pracy bez asystentów AI, walidacja odgrywa kluczową rolę w budowaniu systemów lub aplikacji. Tradycyjnie oznaczało to pisanie testów, testowanie różnych komponentów oraz pozwalanie użytkownikom na wypróbowanie różnych części. Pod tym względem używanie podpowiedzi niczym się nie różni. Efektem ubocznym korzystania z LLM-ów jest to, że mogą one generować tekst, który nie jest istotny dla problemu albo rozwiązuje problem w nieoptymalny sposób. Ponieważ kod jest generowany na podstawie Twoich podpowiedzi, weryfikacja poprawności i istotności rozwiązania staje się jeszcze ważniejsza.

W kolejnych podrozdziałach opiszemy szczegółową strategię dla obu wybranych dziedzin problemowych: tworzenia aplikacji internetowych typu full-stack oraz nauki o danych.

Techniki podpowiedzi

Istnieją pewne techniki, które okazują się szczególnie użyteczne podczas instruowania asystenta AI, ponieważ pozwalają w pełni wykorzystać jego możliwości. Ogólna rada jest taka, żeby zapewnić asystentowi wystarczający kontekst, ale co to dokładnie oznacza, można pokazać bardziej konkretnie przez omówienie poniższych wzorców:

- Wzorzec zadanie – działanie – wskazówka.
- Wzorzec persona – instrukcja – treść.
- Podpowiedź eksploracyjna.

Wzorzec zadanie – działanie – wskazówka (ang. Task-Action-Guideline, TAG)

Aby wyjaśnić, jak działa ten wzorzec, przeanalizujmy jego części i zbadajmy przykładową podpowiedź wraz z przykładową odpowiedzią.

Intencja. Poinstruować asystenta AI, jakie zadanie ma rozwiązać, jakie ma podjąć działania i według jakich wskazówek.

- **Zadanie**. Jasno zdefiniuj zadanie, które ma wykonać asystent AI.
- **Działanie**. Określ etapy lub działania, które powinien podjąć asystent w celu wykonania zadania.
- **Wskazówka**. Określ reguły, ograniczenia lub wytyczne, których powinien przestrzegać asystent podczas wykonywania zadania.

Przykład:

[Podpowiedź]

Zadanie: Wygeneruj streszczenie najnowszego artykułu prasowego o zmianie klimatu.

Działanie:

1. Wyszukaj najnowsze artykuły prasowe o zmianie klimatu.
2. Wybierz najbardziej istotny i wiarygodny artykuł.
3. Podsumuj kluczowe tematy artykułu w 3 – 4 zdaniach.

Wskazówka:

1. Zadbaj o to, żeby streszczenie było zwięzłe i dokładne.
2. Unikaj technicznego żargonu; posługuj się prostym, przystępnym językiem.
3. Przytocz źródło artykułu na końcu streszczenia.

[Koniec podpowiedzi]

[Odpowiedź]

Streszczenie: W niedawnym artykule BBC News pisze o alarmującym wzroście globalnych temperatur w minionej dekadzie. Naukowcy ostrzegają, że jeśli bieżący trend się nie zmieni, mogą czekać nas bezprecedensowe wydarzenia klimatyczne. Artykuł podkreśla pilną potrzebę międzynarodowej współpracy na rzecz walki ze zmianami klimatu. [Źródło: BBC News]

Ten wzorzec pomaga zagwarantować, że asystent AI rozumie zadanie, zna kroki, które musi przedsięwziąć, i przestrzega określonych wytycznych, co prowadzi do dokładniejszych i użyteczniejszych wyników.

[Koniec odpowiedzi]

Oto jak mogłaby wyglądać przykładowa podpowiedź dotycząca tworzenia aplikacji internetowej:

[Podpowiedź]

Zadanie: Opracuj RESTful API dla prostej aplikacji blogowej z użyciem Node.js i Express.

Działanie:

1. Skonfiguruj nowy projekt Node.js.
2. Utwórz serwer Express.
3. Zdefiniuj trasy dla operacji CRUD na postach na blogu.
4. Zaimplementuj oprogramowanie pośrednie do parsowania JSON i obsługi błędów.

5. Połącz się z bazą danych MongoDB z użyciem Mongoose.
6. Napisz testy dla punktów końcowych API.

Wskazówka:

1. Zadbaj o to, aby konfiguracja projektu była prosta i klarowna.
2. Podaj zwięzłe i dokładne przykłady kodu.
3. Unikaj używania nadmiernie złożonej terminologii; posługuj się prostym, przystępnym językiem.
4. Dołącz do kodu komentarze wyjaśniające każdą czynność

[Koniec podpowiedzi]

Wzorzec persona – instrukcja – kontekst (ang. Persona-Instruction-Context, PIC)

Opiszmy ten wzorzec w ten sam sposób co poprzedni — zdefiniujmy go i zilustrujmy przykładową podpowiedzią oraz przykładową odpowiedzią:

Intencja. Poinstruować asystenta AI, jaką personę powinien przyjąć i jakie instrukcje wykonać, oraz zapewnić wystarczający kontekst do wygenerowania poprawnej odpowiedzi.

- **Persona**. Zdefiniuj personę albo cechy charakteru, które powinna odzwierciedlać odpowiedź asystenta.
- **Instrukcja**. Określ zadania lub działania, które powinien wykonać asystent.
- **Kontekst**. Podaj istotne informacje ogólne lub kontekst, aby pomóc asystentowi zrozumieć sytuację i wygenerować poprawną odpowiedź.

Przykład:

[Podpowiedź]

Persona. Zastosuj personę przyjaznego i znającego się na rzeczy przewodnika turystycznego.

Instrukcja:

1. Poleć trzy atrakcje w Paryżu.
2. Podaj krótki opis każdej atrakcji.
3. Zasugeruj najlepszą porę odwiedzenia każdej atrakcji.

Kontekst. Użytkownik planuje swoją pierwszą wycieczkę do Paryża i jest zainteresowany zarówno zabytkami, jak i lokalną kulturą.

[Koniec podpowiedzi]

[Odpowiedź]

Jako Twój przyjazny i znający się na rzeczy przewodnik chciałbym polecić Ci trzy obowiązkowe atrakcje w Paryżu:

1. **Wieża Eiffla**. Z tego symbolu Paryża roztacza się zapierający dech w piersiach widok na całe miasto. Najlepiej odwiedzić ją wczesnym rankiem lub późnym wieczorem, aby uniknąć tłumów i cieszyć się wschodem albo zachodem słońca.

2. **Luwr**. Luwr to skarbiec historii i kultury, w którym znajdują się tysiące dzieł sztuki, w tym słynna „Mona Lisa". Zaplanuj zwiedzanie na środek tygodnia w godzinach porannych, aby uniknąć największego ruchu.
3. **Montmartre**. Historyczna, tętniąca ulicznym życiem dzielnica artystów. Odwiedź ją późnym popołudniem, aby eksplorować urocze uliczki albo niespiesznie zjeść posiłek w lokalnym bistro.

Miłej wizyty w Paryżu!

[Koniec odpowiedzi]

Wzorzec ten gwarantuje, że AI przybierze właściwą personę, wykona określone instrukcje i użyje dostarczonego kontekstu, aby wygenerować istotną i pomocną podpowiedź.

Oto, jak mogłaby wyglądać taka podpowiedź podczas tworzenia aplikacji internetowej:

[Podpowiedź]

Persona. Zastosuj personę przyjaznego i znającego się na rzeczy mentora programisty.

Instrukcja:

1. Skonfiguruj nowy projekt Node.js.
2. Utwórz serwer Express.
3. Zdefiniuj trasy dla operacji CRUD na postach na blogu.
4. Zaimplementuj oprogramowanie pośrednie do parsowania JSON i obsługi błędów.
5. Połącz się z bazą danych MongoDB z użyciem Mongoose.
6. Napisz testy dla punktów końcowych API.

Kontekst. Użytkownik jest początkującym programistą aplikacji internetowych i chce zbudować swój pierwszy interfejs RESTful API dla aplikacji blogowej. Potrzebuje klarownych, szczegółowych instrukcji i przykładów kodu.

[Koniec podpowiedzi]

Wzorzec podpowiedzi eksploracyjnych

Możesz znaleźć się w sytuacji, w której nie budujesz projektu od początku do końca albo chcesz zbudować tylko jego mniejszą część, a następnie ocenić odpowiedź. W takim przypadku Twoje odpowiedzi będą miały naturę bardziej eksploracyjną i mogą wyglądać tak, jak pokazano poniżej:

[Podpowiedź]

Oczyść dane.

[Koniec podpowiedzi]

Zakładamy tu, że mamy otwarty notatnik z kodem, który pobrał już dane.

A oto przykład związany z tworzeniem aplikacji internetowej:

[Podpowiedź]

Dodaj CSS do tej listy produktów.

[Koniec podpowiedzi]

W tym wzorcu podpowiedzi są zwykle dużo krótsze, mają kontekst (wywodzący się z istniejącego kodu albo przekazany w inny sposób), a programista rzadko wybiega myślą poza bieżący etap.

Wzorzec nauka – improwizacja – informacje zwrotne – ewaluacja (ang. Learn-Improvise-Feedback-Evaluate, LIFE)

Ten wzorzec, podobnie jak TAG i PIC, pomaga sformułować problem i zapewnia dobre rozwiązanie wyjściowe, które możesz potem dopracować.

- **Nauka**. Podkreśl znaczenie zrozumienia danych z użyciem różnych technik analitycznych, od podstawowej statystyki do złożonych korelacji i analizy czasowej.
- **Improwizacja**. Zaadaptuj analizę na podstawie wstępnych ustaleń. Na przykład jeśli niektóre kategorie produktów wykazują nietypowe trendy, pogłęb analizę w tych obszarach.
- **Informacje zwrotne**:
  - Udostępnij kod i wyniki modelu, aby zapewnić efektywną naukę i zrozumienie.
  - Uwzględnij rady i krytykę, aby dopracować model i podejście.
  - Podaj błędy, aby umożliwić zdiagnozowanie i rozwiązanie problemów.
- **Ewaluacja**. Wykonaj kod podany przez ChatGPT, aby zweryfikować jego dokładność i poprawność.

Przykładowa podpowiedź w tym wzorcu mogłaby wyglądać tak:

[Podpowiedź]

Tytuł projektu: Budowanie strony wyników wyszukiwania internetowego. Musisz zaproponować etapy i kod.

Cel: Utwórz dynamiczną i interaktywną stronę wyników wyszukiwania, która wyświetla i filtruje wyniki wyszukiwania na podstawie zapytań użytkowników. Etapy:

- **Nauka**. Zrozum znaczenie efektywnego wyświetlania wyników wyszukiwania. Przeanalizuj dane dotyczące interakcji użytkownika, aby zidentyfikować kluczowe wskaźniki, takie jak klikalność, średni czas spędzony na stronie wyników i ocena satysfakcji użytkownika. Użyj różnych technik analitycznych, od podstawowych statystyk, po złożone korelacje i analizę czasową, aby uzyskać wgląd w zachowania i preferencje użytkowników.
- **Improwizacja**. Na podstawie wstępnych ustaleń dostosuj stronę wyników wyszukiwania, aby wyróżnić najbardziej istotne wyniki. Na przykład, jeśli użytkownicy często klikają określone typy wyników, nadaj im priorytet w wyświetlaniu. Wdróż funkcje, takie jak filtry, opcje sortowania i paginacja, aby poprawić wrażenia użytkownika.

- **Informacje zwrotne**. Udostępnij swój kod i stronę wyników wyszukiwania innym programistom lub mentorom w celu zasięgnięcia opinii. Pomoże to zapewnić skuteczną naukę i zrozumienie. Uwzględnij ich rady i krytykę, aby udoskonalić stronę wyników wyszukiwania i poprawić jej funkcjonalność. Podaj błędy: dokumentuj wszelkie błędy napotkane podczas tworzenia oprogramowania, takie jak problemy z pobieraniem lub wyświetlaniem danych. Rozwiąż te problemy, ucząc się podczas tego procesu, aby zwiększyć swoje umiejętności.
- **Ewaluacja**. Wykonaj kod dostarczony przez ChatGPT, aby upewnić się, że jest dokładny i poprawny. Sprawdź stronę wyników wyszukiwania, porównując jej działanie z rzeczywistymi danymi i opiniami użytkowników. Wprowadź niezbędne zmiany, aby poprawić dokładność i satysfakcję użytkowników.

[Koniec podpowiedzi]

Odpowiedź asystenta mogłaby wyglądać tak:

[Odpowiedź]

Świetnie! Przeanalizujmy etapy i podajmy kilka fragmentów kodu, aby pomóc Ci w zbudowaniu dynamicznej i interaktywnej strony wyników wyszukiwania.

- **Tytuł projektu**: Budowanie strony wyników wyszukiwania internetowego.
- **Cel**: Utworzenie dynamicznej i interaktywnej strony wyników wyszukiwania, która efektywnie wyświetla i filtruje wyniki na podstawie zapytań użytkowników.

Etapy:

Nauka:

- **Zrozum znaczenie efektywnego wyświetlania wyników wyszukiwania**.
  - Przenalizuj dane dotyczące interakcji z użytkownikami, aby zidentyfikować kluczowe wskaźniki, takie jak CTR, średni czas spędzany na stronach wyszukiwania i oceny satysfakcji użytkowników.
  - Użyj różnych technik analitycznych, od podstawowych statystyk, po złożone korelacje i analizę czasową, aby uzyskać wgląd w zachowania i preferencje użytkowników.

Improwizacja:

- **Na podstawie wstępnych ustaleń dostosuj stronę wyników wyszukiwania**.
  - Wyróżnij najbardziej istotne wyniki na podstawie danych o interakcjach z użytkownikami.
  - Zaimplementuj takie funkcje jak filtry, opcje sortowania i paginacja, aby poprawić wrażenia użytkownika.

Informacje zwrotne:

- **Udostępnij swój kod i stronę wyników wyszukiwania innym programistom lub mentorom w celu zasięgnięcia opinii**.
  - Uwzględnij ich rady i krytykę, aby udoskonalić stronę wyników wyszukiwania i poprawić jej funkcjonalność.

Podaj błędy:

- **Dokumentuj wszelkie błędy napotkane podczas tworzenia oprogramowania**.
  - Rozwiąż te problemy, ucząc się podczas tego procesu, aby zwiększyć swoje umiejętności.

Ewaluacja:

- **Wykonaj kod dostarczony przez ChatGPT, aby upewnić się, że jest dokładny i poprawny**.
  - Zweryfikuj stronę wyników wyszukiwania, porównując jej działanie z rzeczywistymi danymi i opiniami użytkowników. Wprowadź niezbędne zmiany, aby poprawić dokładność i satysfakcję użytkowników.

Fragmenty kodu:

Struktura HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Search Results Page</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="search-container">
        <input type="text" id="search-input" placeholder="Search...">
        <button onclick="performSearch()">Search</button>
    </div>
    <div class="results-container" id="results-container"></div>
    <script src="script.js"></script>
</body>
</html>
```

Obstylowanie CSS:

```
/* styles.css */
body {
    font-family: Arial, sans-serif;
}

.search-container {
    margin: 20px;
}

.results-container {
    margin: 20px;
}

.result-item {
    border: 1px solid #ccc;
    padding: 10px;
    margin-bottom: 10px;
}
```

Funkcje JavaScript:

```
// script.js
function performSearch() {
    const query = document.getElementById('search-input').value;
    const resultsContainer = document.getElementById('results-container');
    resultsContainer.innerHTML = ''; // Usuwanie poprzednich wyników

    // Symulacja wyników wyszukiwania
    const results = [
        { title: 'Result 1', description: 'Description for result 1' },
        { title: 'Result 2', description: 'Description for result 2' },
        { title: 'Result 3', description: 'Description for result 3' },
    ];

    results.forEach(result => {
        const resultItem = document.createElement('div');
        resultItem.className = 'result-item';
        resultItem.innerHTML =
`<h3>${result.title}</h3><p>${result.description}</p>`;
        resultsContainer.appendChild(resultItem);
    });
}
```

Dodatkowe funkcje:

- **Filtry i sortowanie**. Dodaj listy rozwijane lub pola wyboru do filtrowania i sortowania wyników.
- **Paginacja**. Zaimplementuj paginację, aby obsługiwać duże zbiory wyników.
- **Obsługa błędów**. Dodaj obsługę błędów, aby uwzględnić problemy z pobieraniem lub wyświetlaniem danych.

Powinno Ci to dać solidne podstawy do zbudowania strony wyników wyszukiwania internetowego. Daj mi znać, jeśli potrzebujesz dalszej pomocy lub bardziej szczegółowych przykładów kodu!

[Koniec odpowiedzi]

Który wzorzec wybrać?

Ogólna odpowiedź jest taka, że wybór wzorca zależy od okoliczności. Wzorce TAG i PIC wybierasz wtedy, gdy masz jasno zdefiniowany problem, który chcesz rozwiązać od początku do końca, i liczysz na to, że asystent wykona za Ciebie „brudną robotę”. Wzorzec eksploracyjny przydaje się wtedy, kiedy pracujesz nad istniejącą bazą kodu albo chcesz wypróbować różne podejścia. Ogólnie radzimy wypróbować wszystkie trzy wzorce i sprawdzić, co najlepiej pasuje do Twojego podejścia i problemu.

Strategia podpowiedzi w tworzeniu aplikacji internetowych

Wykorzystajmy wskazówki z poprzedniego podrozdziału i na ich podstawie krok po kroku opracujmy strategię podpowiedzi.

Analiza problemu „system internetowy do zarządzania zapasami"

Używamy prawdziwego przykładu zarządzania zapasami w aplikacji internetowej, żeby pokazać ogólne podejście do problemu. „Zarządzanie" zapasami wymaga odczytywania i zapisywania danych. Najprawdopodobniej w Twoim systemie/aplikacji będą różne role, od administratorów, po zwykłych użytkowników. Być może będziesz musiał również rozważyć, jak ten system pasuje do innych systemów, czy powinieneś na przykład zintegrować go z innymi systemami i z jakich części będzie on składać się w takim przypadku.

Dziedzina wydaje się dość prosta, więc przejdźmy do ustalenia, z jakich części składa się problem.

Na ogólnym poziomie rozumiemy, co system powinien robić. Aby jednak rozwiązać problem, musimy podzielić go na mniejsze części, za które w aplikacjach internetowych zwykle odpowiadają następujące komponenty:

- **Frontend**. Frontend to część systemu, z którą użytkownik wchodzi w interakcję. Frontend odpowiada za prezentowanie danych użytkownikowi i przyjmowanie od niego danych wejściowych.
- **Backend**. Ta część systemu komunikuje się z frontendem. Backend odpowiada za odczytywanie i zapisywanie danych w bazie danych. W bardziej złożonym systemie mogą istnieć różne frontendy i różne aplikacje, które komunikują się z backendem.
- **Baza danych**. Baza danych to część systemu, która przechowuje dane. Jest to magazyn danych, na przykład relacyjna baza danych, taka jak MySQL lub PostgreSQL. Baza danych odpowiada za przechowywanie danych w sposób, który zapewnia wysoką wydajność oraz łatwość odczytu i zapisu.
- **Raportowanie**. Aplikacja często ma część raportującą do prezentowania spostrzeżeń. Pobiera ona dane z magazynu danych i może wymagać przekształcenia danych do celów raportowania.

Dalszy podział frontendu na funkcje

Taki ogólny przegląd jest użyteczny, ale zwykle nie wystarczy, żebyśmy mogli zacząć pisać podpowiedzi. Musimy podzielić problem na mniejsze części, zwykle według funkcji. Dalszy podział frontendu na funkcje może wyglądać tak:

- **Logowanie**. Użytkownik musi móc zalogować się w systemie.
- **Wylogowywanie**. Użytkownik musi móc wylogować się z systemu.

- **Przeglądanie zapasów**. Użytkownik musi móc przeglądać zapasy.
- **Dodawanie zapasów**. Użytkownik musi móc dodawać zapasy.
- **Usuwanie zapasów**. Użytkownik musi móc usuwać zapasy.
- **Aktualizowanie zapasów**. Użytkownik musi móc aktualizować zapasy.

Generowanie podpowiedzi dla każdej funkcji

W tym momencie problem jest sformułowany na tyle szczegółowo, że możemy zacząć pisać podpowiedzi. Weźmy pierwszą funkcję, logowanie, i zastanówmy się, jak skonstruować podpowiedź.

Mógłbyś użyć narzędzia takiego jak ChatGPT lub GitHub Copilot (więcej na ten temat powiemy w następnym rozdziale) i zacząć od następującej podpowiedzi:

[Podpowiedź]

Utwórz stronę logowania, która pozwala użytkownikowi zalogować się w systemie.

[Koniec podpowiedzi]

Choć mogłoby to zadziałać, pomijasz mnóstwo kontekstu, m.in. czego dokładnie Ci potrzeba, a czego nie, jakich technologii używasz, jaki planujesz interfejs użytkownika itd.

Spróbujmy ulepszyć podpowiedź, dodając więcej kontekstu:

[Podpowiedź]

Utwórz stronę logowania z polami na nazwę użytkownika i hasło. Powinna ona zawierać łącze do tworzenia konta użytkownika oraz przycisk logowania, być wyśrodkowana pionowo i poziomo oraz działać dobrze na telefonach komórkowych i tabletach. Powinna być napisana w React i używać biblioteki Material UI.

[Koniec podpowiedzi]

Podstawowe zasady — „strategia podpowiedzi" w tworzeniu aplikacji internetowych

Jak widziałeś w poprzednich przykładach, dzielimy problem na mniejsze, bardziej „poręczne" części i sugerujemy podpowiedzi umożliwiające zrealizowanie konkretnych funkcji. A zatem teraz, kiedy lepiej rozumiesz nasz przykład z „zarządzaniem zapasami", jak właściwie ma wyglądać nasza „strategia podpowiedzi"? Po pierwsze trzeba zauważyć, że strategia będzie zależna od kontekstu. Ponieważ zajmujemy się tworzeniem aplikacji internetowych, musimy używać słów kluczowych należących do tej dziedziny, a także bibliotek i architektur odpowiednich dla danego obszaru. Oto kilka wskazówek, które mogą okazać się pomocne:

- **Zapewnij kontekst — pola**. Ekran logowania może składać się tylko z pól nazwy użytkownika i hasła. Większość ekranów logowania ma więcej pól, na przykład pole potwierdzenia hasła, łącze do resetowania hasła, pole

do tworzenia nowego użytkownika itd. W zależności od potrzeb może będziesz musiał to określić bardzo szczegółowo.

- **Określ, jak to zrobić — projekt i opcje technologiczne**. Ekran logowania można zaprojektować na wiele różnych sposobów. Dziś często optymalizuje się go pod kątem różnych urządzeń, takich jak tablety, smartfony, duże ekrany itd. Jeśli chodzi o opcje technologiczne, w tworzeniu aplikacji internetowych jest ich pod dostatkiem, od platform takich jak React, Vue i Angular do prostego HTML-a i CSS-a. Określ opcje w zależności od potrzeb projektu.
- **Iteruj**. Różne narzędzia reagują inaczej na tę samą podpowiedź. W książce tej pokażemy Ci, jak używać takich narzędzi jak GitHub Copilot i ChatGPT. Każde narzędzie ma swoje wady i zalety i może oferować różne wyniki. Wypróbuj różne wersje podpowiedzi, dodając separatory, jak przecinki i dwukropki, spróbuj też przeformułować podpowiedź.
- **Bądź świadomy kontekstu**. Kiedy używasz narzędzi ChatGPT i GitHub Copilot, robisz to w istniejącym kontekście. W przypadku ChatGPT oznacza to prowadzenie bieżącej konwersacji z podpowiedziami i odpowiedziami, z kolei GitHub Copilot widzi nie tylko to, co napisałeś w otwartym pliku, ale również całą Twoją przestrzeń roboczą (jeśli mu na to pozwolisz). Asystent bada ten kontekst, żeby zdecydować, co wygenerować w odpowiedzi. Musisz być świadomy tego kontekstu, a jeśli nie otrzymujesz pożądanych odpowiedzi, w ChatGPT rozpocznij nową konwersację, w Copilocie zamknij otwarte pliki i zacznij pisać w pustym pliku itd.

Strategia podpowiedzi w data science

Przeprowadźmy teraz eksperyment myślowy podobny do tego, który omówiliśmy w kontekście tworzenia aplikacji internetowych, ale dotyczący nauki o danych. Użyjemy tych samych wskazówek — „przeanalizuj problem" i „wygeneruj podpowiedzi" — i podobnie jak w poprzednim podrozdziale wyciągniemy pewne ogólne wnioski dotyczące strategii podpowiedzi w data science.

Analiza problemu „przewidywanie sprzedaży"

Przypuśćmy, że budujemy model uczenia maszynowego do przewidywania sprzedaży. Na ogólnym poziomie rozumiemy, co system powinien robić. Aby jednak rozwiązać problem, musimy podzielić go na mniejsze części, za które w nauce o danych zwykle odpowiadają następujące komponenty:

- **Dane**. Dane to część systemu, która przechowuje informacje. Dane mogą pochodzić z wielu miejsc, takich jak bazy danych, internetowe punkty końcowe, statyczne pliki itd.
- **Model**. Model jest odpowiedzialny za uczenie się z danych i generowanie jak najdokładniejszych przewidywań. Do przewidywania potrzebne są dane wejściowe, które w wyniku dają wyjściową prognozę.

- **Trening**. Część systemu, która odpowiada za trenowanie modelu. Zwykle części danych używa się do treningu, a część traktuje jako próbki.
- **Ewaluacja**. Aby upewnić się, że model działa zgodnie z oczekiwaniami, musisz poddać go ewaluacji. Ewaluacja polega na wzięciu danych i modelu i obliczeniu wskaźnika, który określa, jak dobrze działa model.
- **Wizualizacja**. Część systemu, która pomaga w dokonywaniu cennych spostrzeżeń biznesowych dzięki wykresom. Jest to bardzo ważna część, ponieważ to ona jest najbardziej widoczna dla użytkowników biznesowych.

Dalszy podział na funkcje i etapy

W tym momencie jesteśmy na zbyt ogólnym poziomie, żebyśmy mogli zacząć pisać podpowiedzi. Musimy podzielić problem na mniejsze części, badając każdy etap.

- **Dane**. Zarządzanie danymi obejmuje wiele czynności, w tym gromadzenie danych, oczyszczanie ich i przekształcanie. Oto jeden z możliwych podziałów:
  1. **Zgromadź dane**. Dane trzeba skądś zebrać. Może to być baza danych, internetowy punkt końcowy, statyczny plik itd.
  2. **Oczyść dane**. Dane trzeba oczyścić. Oczyszczanie polega na usuwaniu nieistotnych danych, usuwaniu duplikatów itd.
  3. **Przekształć dane**. Dane wymagają przekształcenia. Przekształcenie oznacza zmianę danych w format użyteczny dla modelu.
- **Trening**. Podobnie jak zarządzanie danymi, trening składa się z wielu etapów. Oto propozycja podziału:
  1. **Rozdziel dane**. Dane trzeba rozdzielić na zbiór treningowy i ewaluacyjny. Dane treningowe służą do trenowania modelu, a dane ewaluacyjne — do oceny działania modelu.
  2. **Wytrenuj model**. Model trzeba wytrenować. Podczas treningu model uczy się na danych treningowych.
- **Ewaluacja**. Ewaluacja jest zwykle pojedynczym etapem, ale można podzielić ją na mniejsze części.

Generowanie podpowiedzi dla każdego etapu

Zauważ, że podział problemu w data science wygląda nieco inaczej niż w tworzeniu aplikacji internetowych. Zamiast funkcji takich jak *Dodaj zapasy* mamy funkcje takie jak *Zgromadź dane*.

Jesteśmy jednak na odpowiednim poziomie, żeby skonstruować podpowiedź, więc użyjmy funkcji *Zgromadź dane* jako naszego przykładu.

[Podpowiedź]

Zgromadź dane z pliku *data.xls* i wczytaj je do `DataFrame` z użyciem biblioteki Pandas.

[Koniec podpowiedzi]

Powyższa podpowiedź jest jednocześnie ogólna i konkretna. Jest ogólna w tym sensie, że nakazuje „zgromadzić dane", a konkretna w tym, że określa, jakiej biblioteki, a nawet jakiej struktury danych (`DataFrame`) należy użyć. Bardzo możliwe, że zadziałałaby tu również prostsza podpowiedź, na przykład:

[Podpowiedź]

Zgromadź dane z *data.xls*.

[Koniec podpowiedzi]

Może to zależeć od tego, czy Twoim narzędziem jest ChatGPT, czy GitHub Copilot.

Podstawowe zasady — „strategia podpowiedzi" w tworzeniu aplikacji internetowych

Poniżej podajemy kilka zasad podobnych do tych, które omówiliśmy w przykładzie z tworzeniem aplikacji internetowej:

- **Zapewnij kontekst — nazwa pliku**. Pliki CSV mogą mieć dowolne nazwy. Ważne jest określenie nazwy pliku.
- **Określ, jak to zrobić — biblioteki**. Istnieje wiele sposobów wczytywania pliku CSV, a choć biblioteka Pandas jest popularną opcją, trzeba podać jej nazwę. Dostępne są też inne biblioteki i możesz potrzebować na przykład rozwiązania dla języków Java, C# lub Rust, w których biblioteki nazywają się inaczej.
- **Iteruj**. Warto wypróbować różne wersje podpowiedzi, dodając separatory, jak przecinki i dwukropki, a także przeformułować podpowiedź.
- **Bądź świadomy kontekstu**. Również tutaj kontekst ma duże znaczenie; jeśli pracujesz w notatniku, GitHub Copilot będzie miał dostęp do poprzednich komórek, ChatGPT będzie miał dostęp do historii konwersacji itd.

Jak wynika z powyższych wskazówek, strategia jest bardzo podobna jak w tworzeniu aplikacji internetowych. Wymieniamy te same punkty — „Zapewnij kontekst", „Określ, jak to zrobić", „Iteruj" i „Bądź świadomy kontekstu". Różnica tkwi w szczegółach. Istnieje jednak alternatywna strategia, która sprawdza się w nauce o danych, a są to długie podpowiedzi. Choć w tym przykładzie podzieliliśmy problem data science na etapy, nie musisz pisać oddzielnej podpowiedzi dla każdego etapu. Innym rozwiązaniem problemu byłoby podanie wszystkich żądanych zadań w jednej dużej podpowiedzi. Podpowiedź mogłaby zatem wyglądać tak:

[Podpowiedź]

Chcesz przewidywać sprzedaż na podstawie pliku *data.xls*. Użyj Pythona i biblioteki Pandas. Oto etapy, które powinieneś wykonać:

- Zgromadź dane.
- Oczyść dane.
- Przekształć dane.
- Podziel dane.

- Wytrenuj model.
- Przeprowadź ewaluację.

[Koniec podpowiedzi]

W dalszych rozdziałach poświęconych nauce o danych i uczeniu maszynowemu będziemy podawać przykłady zarówno z mniejszymi, jak i większymi podpowiedziami. Sam zdecyduj, które podejście bardziej Ci odpowiada.

Walidacja rozwiązania

Najważniejszą częścią tej strategii jest weryfikacja dokładności i upewnienie się, że tekst i kod utworzony przez AI są poprawne. Istnieją dwa ogólne podejścia do weryfikacji wyników:

- **Weryfikacja z użyciem podpowiedzi**. Pierwsze podejście polega na użyciu podpowiedzi do weryfikacji wyniku. Oznacza to pisanie podpowiedzi pytających o poprawność konkretnych wyników. Może to być dobra strategia na początku procesu weryfikacji. W ten sposób szukasz sytuacji, w której odpowiedzi asystenta AI nie są konsekwentne.
- **Klasyczna weryfikacja**. Drugim podejściem jest użycie klasycznych technik weryfikacji. Techniki te zależą od dziedziny problemu. Na ogólnym poziomie sprowadza się to jednak do testowania kodu, porównywania wyników oraz sprawdzania ich poprawności z wykorzystaniem własnej wiedzy (i wiedzy współpracowników).

Narzędzie AI tak naprawdę nie wie, co robi. Odpowiedzi, których udziela, zależą od jego korpusu treningowego. Powinieneś zawsze o tym pamiętać i polegać na swojej wiedzy podczas weryfikacji wyników.

W następnych punktach zbadamy różne podejścia do ręcznej i klasycznej weryfikacji.

Weryfikacja z użyciem podpowiedzi

Podpowiedzi możesz używać zarówno do generowania wyników, które przybliżają Cię do rozwiązania problemu, jak i do ich weryfikacji. Wróćmy do przykładu, w którym budowaliśmy ekran logowania. Napisaliśmy w nim następującą podpowiedź:

[Podpowiedź]

Utwórz stronę logowania z polami na nazwę użytkownika i hasło. Powinna ona zawierać łącze do tworzenia konta użytkownika oraz przycisk logowania, być wyśrodkowana pionowo i poziomo oraz działać dobrze na telefonach komórkowych i tabletach. Powinna być napisana w React i używać biblioteki Material UI.

[Koniec podpowiedzi]

Aby zweryfikować wynik, możemy napisać następującą podpowiedź:

[Podpowiedź]

Co robi poniższy kod?

```
.login {
  <!-- brakuje kodu CSS do środkowania w poziomie i pionie oraz zapewniania responsywności -->
  @media (min-width: 768px) {
  <!-- brakuje kodu CSS do środkowania w poziomie i pionie oraz zapewniania responsywności -->
  }
}

<div class="login">
  <TextField id="username" label="Username" />
  <TextField id="password" label="Password" />
  <Button variant="contained" color="primary">
    Login
  </Button>
</div>
```

[Koniec podpowiedzi]

W rzeczywistości nie potrzebujesz podpowiedzi, żeby określić, co robi pokazany wyżej kod; widać, że brakuje w nim kodu CSS, który zapewniałby responsywność. Chodzi tu jednak o to, że przez pisanie podpowiedzi możesz skłonić asystenta AI do powiedzenia Ci, jak jego zdaniem działa kod.

Używanie podpowiedzi do zadawania pytań dotyczących danych wyjściowych jest dobrym pierwszym krokiem do weryfikacji wyników. Czasem to jednak nie wystarcza i musisz skorzystać z klasycznych technik weryfikacji, które omówimy w następnym punkcie.

Klasyczna weryfikacja

Sposób weryfikacji wyników zależy od dziedziny problemu. W tworzeniu aplikacji internetowych możesz użyć różnych narzędzi i technik, jak na przykład:

- **Testowanie**. Testy całego rozwiązania lub samego frontendu pozwalają upewnić się, że kod działa zgodnie z oczekiwaniami. Testy tego typy zwykle polegają na programistycznym symulowaniu interakcji użytkownika ze stroną internetową z użyciem narzędzi takich jak Selenium.
- **Testowanie ręczne**. Możesz ręcznie przetestować stronę internetową, otwierając ją w przeglądarce i wchodząc z nią w interakcje. Jest to dobre podejście na początku procesu weryfikacji. Oprócz badania interakcji możesz wizualnie zweryfikować stronę i upewnić się, że jej wygląd jest zgodny z Twoimi wymaganiami.
- **Recenzja kodu**. Możesz przejrzeć kod i sprawdzić, czy jest poprawny. Jest to dobre podejście na początku procesu weryfikacji. Pozwala ono zweryfikować wynik nie tylko Tobie, ale również Twoim współpracownikom.
- **Narzędzia**. Narzędzia mogą przetestować wiele różnych scenariuszy, takich jak dostępność, wydajność itd. Narzędzia te prawdopodobnie są już częścią Twojego procesu tworzenia aplikacji.

Kiedy realizujesz projekty data science, możesz wykorzystać wszystkie poprzednie podejścia, ale możesz też użyć innych. Do najpopularniejszych należą:

- **Testy jednostkowe**. Pozwalają upewnić się, że kod działa zgodnie z oczekiwaniami.
- **Testy integracyjne**. Również służą do sprawdzania, czy kod działa zgodnie z oczekiwaniami.
- **Walidacja wyników**. Ten typ walidacji polega na porównywaniu wyników analizy lub modelu ze znanymi wynikami lub wartościami odniesienia.
- **Walidacja krzyżowa**. W tym typie walidacji dzielisz dane na zbiór treningowy i ewaluacyjny, trenujesz model na danych treningowych i oceniasz go na danych walidacyjnych. Jest to dobre podejście na początku procesu weryfikacji.

Podsumowanie

W rozdziale tym przedstawiliśmy strategię rozwiązywania problemów z użyciem podpowiedzi i weryfikowania wyników.

Dowiedziałeś się, jak projekty aplikacji internetowych i data science można podzielić na mniejsze problemy, które można rozwiązać z wykorzystaniem podpowiedzi. Podaliśmy też pewne podstawowe zasady pisania podpowiedzi.

Na koniec wyjaśniliśmy, jak ocenić poprawność rozwiązania z użyciem podpowiedzi i klasycznych technik weryfikacji.

Mamy nadzieję, że wrócisz do tego rozdziału, kiedy będziesz musiał rozwiązać problem związany z tworzeniem aplikacji internetowych albo nauką o danych i będziesz szukał efektywnego podejścia.

Odpytywanie asystenta AI to coś więcej niż pisanie podpowiedzi i otrzymywanie odpowiedzi. W pozostałych rozdziałach książki zobaczysz, jak stosujemy omówione wyżej zasady w różnych dziedzinach w celu rozwiązywania problemów. Staraj się wpisywać przykładowe podpowiedzi w miarę lektury — dostosuj je do swoich potrzeb i sprawdź, co się stanie.

W następnym rozdziale dowiesz się więcej o dwóch preferowanych przez nas asystentach, GitHub Copilot i ChatGPT.

AI w programowaniu — twórz szybciej i skuteczniej!

Obecnie miliony użytkowników korzystają z dużych modeli językowych do generowania treści, analizy danych, pisania kodu i automatyzacji pracy. Narzędzia takie jak ChatGPT i GitHub Copilot pozwalają na zwiększenie efektywności i radzenie sobie ze skomplikowanymi wyzwaniami. Ułatwiają również tworzenie aplikacji na profesjonalnym poziomie.

Ta książka jest przeznaczona dla programistów, którzy chcą używać AI do optymalizacji procesu tworzenia oprogramowania. Znalazły się tu praktyczne informacje dotyczące budowy interfejsu użytkownika, backendu, tworzenia i optymalizacji kodu. Opisano, jak pisać interfejsy Web API, refaktoryzować kod i zwiększać jego wydajność za pomocą Copilota. Omówiono ponadto sposoby formułowania podpowiedzi dla przetwarzania danych, inżynierii cech, doboru modeli, ich trenowania, strojenia hiperparametrów i oceny jakości uczenia maszynowego. Nie zabrakło również zaawansowanych technik pracy z Copilotem i agentami programowymi, a także omówienia zasad wywoływania narzędzi AI.

W książce:

- budowa modeli uczenia maszynowego za pomocą GitHub Copilot i ChatGPT
- korzystanie z asystentów AI w całym cyklu tworzenia oprogramowania
- techniki inżynierii podpowiedzi w projektach data science
- tworzenie frontendu i backendu aplikacji internetowej za pomocą sztucznej inteligencji
- refaktoryzacja kodu i poprawa jego efektywności i czytelności
- optymalizacja przepływów pracy

Christoffer Noring pracuje w Microsofcie, posiada tytuł Google Developer Expert. Jest autorem książek i wykładowcą na Uniwersytecie Oksfordzkim. **Anjali Jain** jest wykładowcą w Oksfordzie i architektem danych w Metrobanku, gdzie rozwija oprogramowanie dla sektora finansowego. **Marina Fernandez** specjalizuje się w zarządzaniu ryzykiem finansowym. Współpracuje z zespołem akademickim Uniwersytetu Oksfordzkiego, posiada tytuły MCP i CSM. **Ayşe Mutlu** zajmuje się technologiami Azure AI i DevOps, tworzy i wdraża modele uczenia maszynowego i uczenia głębokiego. **Ajit Jaokar** jest specjalistą data science w Feynlabs, gdzie buduje prototypy AI dla złożonych aplikacji. Jest również kierownikiem kursu AI na Uniwersytecie Oksfordzkim.

Helion
helion.pl
HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-289-2425-3
9 788328 924253
Cena: 129,00 zł
<packt>