

Porównanie SI w tematach o programowaniu

Autor: Rydzewski Damian 1EM

Wprowadzenie:

ChatGPT i Bard są to zaawansowane modele językowe odpowiednio stworzone przez odpowiednio firmy OpenAI oraz Google. Działają one w formie webowego chatbota, gdzie można wpisać pytanie i wręcz natychmiast dostać odpowiedź. Lecz trzeba podkreślić, że sztuczna inteligencja wedle definicji jest to pewnego rodzaju samoświadomy program a zarówno bard oraz ChatGPT nie posiadają żadnej świadomości, ich zadaniem jest tylko tworzyć najbardziej zbliżone odpowiedzi do tych ludzkich. W tym referacie skupię się na porównaniu tych dwóch najbardziej popularnych modeli językowych, na podstawie różnych pytań związanych z programowaniem i postaram się opisać cechy najważniejsze konkurentów.

Pytanie 1: Jakie są główne różnice między językami programowania kompilowanymi a interpretowanymi?

Bard pisze o wiele bardziej zwięźle sam tekst przypomina bardziej swego rodzaju notatkę. Tymczasem ChatGPT produkuje coś w stylu rozprawki bądź referatu. Gdzie obydwie odpowiedzi są bardzo podobne, różnią się właśnie stylem.

Materiał:

| Pytanie | Odpowiedź | |
|--|--|---|
| | Google Bard | OpenAI ChatGPT (GPT-3.5) |
| Jakie są główne różnice między językami programowania kompilowanymi a interpretowanymi ? | <p>Główną różnicą między językami programowania kompilowanymi a interpretowanymi jest sposób, w jaki kod jest przetwarzany przez komputer. W językach kompilowanych kod źródłowy jest przetłumaczony na kod maszynowy, który jest następnie wykonywany przez procesor. W językach interpretowanych kod źródłowy jest tłumaczony na kod maszynowy w czasie rzeczywistym, podczas wykonywania programu.</p> <p>Oto kilka innych różnic między językami kompilowanymi a interpretowanymi:</p> <ul style="list-style-type: none">• Czas kompilacji i wykonania: Języki kompilowane są kompilowane tylko raz, a następnie mogą być uruchamiane bezpośrednio. Języki interpretowane są interpretowane w czasie rzeczywistym, co może prowadzić do wolniejszego czasu wykonywania.• Błędy: Błędy w składni w językach kompilowanych są wykrywane podczas kompilacji, przed uruchomieniem programu. Błędy w składni w językach interpretowanych są wykrywane w czasie wykonywania, co może prowadzić do nieprzewidywalnego zachowania programu.• Mobilność: Języki kompilowane są zazwyczaj bardziej mobilne niż języki interpretowane, ponieważ kod maszynowy można uruchomić na dowolnym | <p>Języki programowania można podzielić na dwie główne kategorie ze względu na sposób ich wykonywania: kompilowane i interpretowane. Oto główne różnice między nimi:</p> <ol style="list-style-type: none">1. Proces wykonania kodu:<ul style="list-style-type: none">- Kompilowane: W językach programowania kompilowanych kod źródłowy jest najpierw przetwarzany przez kompilator, który tłumaczy go na kod maszynowy lub kod pośredni (np. bytecode). Ten przetworzony kod jest później wykonywany przez interpreter lub maszynę wirtualną.- Interpretowane: W językach programowania interpretowanych kod źródłowy jest przetwarzany przez interpreter linia po linii w czasie rzeczywistym, bez konieczności wcześniejszego kompilowania na kod maszynowy.2. Portowalność:<ul style="list-style-type: none">- Kompilowane: Programy napisane w językach kompilowanych mogą być mniej przenośne, ponieważ kod maszynowy jest specyficzny dla danej platformy. Musisz skompilować kod źródłowy osobno dla różnych platform.- Interpretowane: Języki interpretowane są często bardziej przenośne, ponieważ interpreter działa na różnych platformach. Możesz uruchamiać ten sam kod źródłowy na różnych systemach operacyjnych bez konieczności zmiany. |

| | | |
|--|---|---|
| | <p>komputerze z odpowiednim procesorem. Języki interpretowane wymagają obecności interpretera, który może nie być dostępny na wszystkich platformach.</p> <ul style="list-style-type: none">• Rozmiar kodu: Języki kompilowane generują kod maszynowy, który jest zazwyczaj mniejszy niż kod interpretowany.• Zaawansowane funkcje: Języki kompilowane zazwyczaj obsługują bardziej zaawansowane funkcje, takie jak optymalizacja kodu i zarządzanie pamięcią. <p>Ogólnie rzecz biorąc, języki kompilowane są szybsze i bardziej niezawodne niż języki interpretowane. Języki interpretowane są bardziej elastyczne i łatwiejsze w użyciu.</p> <p>Oto kilka przykładów popularnych języków programowania kompilowanych i interpretowanych:</p> <ul style="list-style-type: none">• Kompilowalne: C, C++, Java, C#, Go, Rust• Interpretowane: Python, JavaScript, PHP, R, Ruby <p>Wybór języka programowania zależy od konkretnego zastosowania. Jeśli zależy Ci na wydajności i niezawodności, wybierz język kompilowany. Jeśli zależy Ci na elastyczności i łatwości użycia, wybierz język interpretowany.</p> | <p>3. Wydajność:</p> <ul style="list-style-type: none">- Kompilowane: Zazwyczaj języki kompilowane oferują lepszą wydajność niż języki interpretowane, ponieważ kod maszynowy jest bardziej efektywny niż kod interpretowany. Programy kompilowane są zazwyczaj szybsze.- Interpretowane: Języki interpretowane mogą być wolniejsze, ponieważ każda linia kodu jest tłumaczona w czasie rzeczywistym przez interpreter. <p>4. Debugowanie:</p> <ul style="list-style-type: none">- Kompilowane: Debugowanie kodu kompilowanego może być bardziej trudne, ponieważ musisz dostarczyć specjalne narzędzia do analizy kodu maszynowego lub kodu pośredniego.- Interpretowane: Języki interpretowane często oferują łatwiejsze debugowanie, ponieważ można dostarczyć interaktywne środowisko programistyczne, które umożliwia analizę kodu źródłowego w czasie rzeczywistym. <p>5. Rozmiar plików wykonywalnych:</p> <ul style="list-style-type: none">- Kompilowane: Pliki wykonywalne generowane przez języki kompilowane są zazwyczaj mniejsze niż pliki wykonywalne generowane przez języki interpretowane, ponieważ nie zawierają kodu źródłowego ani interpretera. <p>Przykłady języków kompilowanych to C, C++, Rust, a przykłady języków interpretowanych to Python, Ruby, JavaScript (w kontekście przeglądarek internetowych). Warto jednak zaznaczyć, że istnieją również języki, które są hybrydą obu podejść, np. Java, która kompiluje kod źródłowy do bytecode'u, który jest później wykonywany przez maszynę wirtualną Java (JVM).</p> |
|--|---|---|

Pytanie 2: Czym jest zmienna w programowaniu i jakie są jej podstawowe cechy?

W tym przypadku powtarza się to samo co w poprzednim punkcie, ale tym razem dodatkowo wspomina o podziale zmiennych na lokalne i globalne o czym gpt najwyraźniej "zapomniał". Tymczasem OpenAI daje dodatkowy bardziej szczegółowy przykład.

Materiał:

| Pytanie | Odpowiedź | |
|--|--|---|
| | Google Bard | OpenAI ChatGPT (GPT-3.5) |
| Czym jest zmienna w programowaniu i jakie są jej podstawowe cechy? | <p>Zmienna w programowaniu to pojemnik na dane, który może być przypisany do różnych wartości w trakcie wykonywania programu. Zmienne są niezbędne do przechowywania danych w programach komputerowych, takich jak liczby, tekst, wartości logiczne i inne.</p> <p>Podstawowe cechy zmiennej to:</p> <ul style="list-style-type: none">• Nazwa: Każda zmienna musi mieć unikalną nazwę, która pozwala na jej identyfikację w programie. Nazwa powinna być jasna i opisowa, aby łatwo można było zrozumieć, co przechowuje zmienna.• Typ: Każda zmienna ma określony typ, który określa rodzaj danych, które może przechowywać. Typy zmiennych mogą obejmować liczby całkowite, liczby rzeczywiste, wartości logiczne, znaki, napisy i inne.• Wartość: Wartość zmiennej to dane, które są obecnie przechowywane w zmiennej. Wartość zmiennej może być zmieniana w trakcie wykonywania programu. <p>Zmienne można podzielić na dwa podstawowe typy:</p> <ul style="list-style-type: none">• Zmienna lokalna: Zmienna lokalna jest widoczna tylko w bloku kodu, w którym została zadeklarowana. Po wyjściu z bloku zmienna lokalna przestaje istnieć.• Zmienna globalna: Zmienna globalna jest widoczna w całym programie. Zmienna globalna może być używana z każdego miejsca w programie. | <p>W programowaniu zmienna to symboliczna nazwa lub etykieta przypisana do pewnej wartości lub miejsca w pamięci komputera. Zmienne są jednym z fundamentalnych conceptów w programowaniu i pełnią kluczową rolę w przechowywaniu i manipulacji danymi w programach komputerowych. Oto podstawowe cechy zmiennych:</p> <ol style="list-style-type: none">1. Nazwa (identyfikator):<ul style="list-style-type: none">- Każda zmienna musi mieć unikalną nazwę, która pozwala na odwoływanie się do niej w kodzie programu. Nazwy zmiennych zazwyczaj składają się z liter, cyfr i podkreślników, ale z reguły nie mogą zaczynać się od cyfry.2. Typ danych:<ul style="list-style-type: none">- Zmienna jest związana z określonym typem danych, który określa, jakie rodzaje wartości może przechowywać oraz jakie operacje można na niej wykonywać. Przykłady typów danych to liczby całkowite (integers), liczby zmiennoprzecinkowe (floats), tekst (string), logiczne (boolean), itp.3. Przypisanie wartości:<ul style="list-style-type: none">- Przed użyciem zmiennej, musisz jej przypisać wartość. Przypisanie wartości do zmiennej polega na zapisaniu danej wartości pod określoną nazwą zmiennej. Na przykład: <code>x = 42`</code>.4. Przechowywanie danych:<ul style="list-style-type: none">- Zmienne przechowują dane w pamięci komputera. Każda zmienna ma swoje miejsce w pamięci, w którym przechowywana jest jej wartość. |

Zmienne są ważną częścią programowania. Pozwalają one na przechowywanie danych w programach komputerowych i na ich zmianę w trakcie wykonywania programu.

5. Modyfikacja wartości:

- Zmienne można zmieniać (aktualizować) w trakcie działania programu, przypisując im nowe wartości. To pozwala na dynamiczną manipulację danymi w programie.

6. Zakres (scope):

- Zakres zmiennej określa, gdzie w kodzie programu jest widoczna i dostępna. Zmienne mogą być zmiennymi lokalnymi (widocznymi tylko w określonym zakresie) lub globalnymi (widocznymi w całym programie).

7. Inicjalizacja:

- Inicjalizacja zmiennej oznacza przypisanie jej początkowej wartości. Niektóre języki programowania wymagają explicitj inicjalizacji, podczas gdy inne domyślnie nadają zmiennym początkową wartość (np. zero lub null).

Przykład w języku Python:

```
```python
Deklaracja i inicjalizacja zmiennej 'x'
x = 10

Zmiana wartości zmiennej 'x'
x = x + 5

Wyświetlenie zawartości zmiennej 'x'
print(x) # Wypisze: 15
```
```

Wartości przechowywane w zmiennych mogą być liczbami, tekstem, obiektami, lub innymi danymi, w zależności od typu zmiennej. Zrozumienie i efektywne zarządzanie zmiennymi jest kluczowe w procesie tworzenia programów, ponieważ to one umożliwiają manipulację danymi i wykonywanie operacji na nich.

Pytanie 3: Jakie są najlepsze praktyki podczas pisania kodu?

Tu znów pojawia się ten sam motyw co w pytaniu numer 1 lecz ChatGPT tym razem wypisał wszystko bez większego składu a bard posiada bardziej czytelny podział na "Podstawowe cechy" i "dodatkowe wskazówki". Tu też model google popełnia błąd gdzie pisze "Pomognie" gdzie prawdopodobnie miało być *pomorze*.

Materiał:

| Pytanie | Odpowiedź | |
|---|---|--|
| | Google Bard | OpenAI ChatGPT (GPT-3.5) |
| Jakie są najlepsze praktyki podczas pisania kodu? | <p>Najlepsze praktyki podczas pisania kodu to zbiór zasad i wskazówek, które pomagają w tworzeniu czystego, łatwego w utrzymaniu i niezawodnego kodu. Dobre praktyki mogą się różnić w zależności od języka programowania i konkretnego zastosowania, ale istnieją pewne ogólne zasady, które warto przestrzegać.</p> <p>Oto kilka najlepszych praktyk podczas pisania kodu:</p> <ul style="list-style-type: none">• Używaj jasnych i opisowych nazw zmiennych, funkcji i innych elementów kodu. Nazwy powinny być łatwe do zrozumienia i zapamiętania.• Korzystaj ze standardowej składni i konwencji. Pomoże to innym programistom zrozumieć Twój kod.• Koduj w sposób uporządkowany i czytelny. Używaj komentarzy, aby wyjaśnić złożone fragmenty kodu.• Testuj swój kod regularnie. Pomognie to w wykryciu i naprawieniu błędów.• Koduj w sposób bezpieczny. Stosuj odpowiednie zabezpieczenia, aby chronić swój kod przed atakami. <p>Oto kilka dodatkowych wskazówek, które mogą pomóc Ci w tworzeniu czystego i przejrzystego kodu:</p> <ul style="list-style-type: none">• Unikaj nadmiernego zagnieżdżania kodu. Zagnieżdżony kod może być trudny do zrozumienia i utrzymania. | <p>Pisanie czytelnego, efektywnego i łatwo zarządzalnego kodu jest kluczowym aspektem tworzenia wysokiej jakości oprogramowania. Oto niektóre najlepsze praktyki, które warto przestrzegać podczas pisania kodu:</p> <ol style="list-style-type: none">1. Zgodność z konwencją nazewniczą:<ul style="list-style-type: none">- Przestrzegaj określonych konwencji nazewniczych w swoim kodzie. To obejmuje stosowanie czytelnych nazw zmiennych, funkcji i klas oraz konsekwentne używanie podkreśleń lub notacji camelCase lub PascalCase.2. Dokumentacja:<ul style="list-style-type: none">- Dodawaj komentarze i dokumentację do swojego kodu, aby opisać, co robi dany fragment kodu, jakie są jego założenia, jakie parametry przyjmuje i jakie zwraca wartości.3. Dekompozycja kodu:<ul style="list-style-type: none">- Dziel swój kod na mniejsze, bardziej zrozumiałe funkcje i metody. Unikaj tworzenia zbyt długich funkcji, które robią wiele rzeczy naraz. To ułatwia debugowanie i testowanie.4. Zasada DRY (Don't Repeat Yourself):<ul style="list-style-type: none">- Unikaj powtarzania tego samego kodu. Zamiast tego wydziel wspólną funkcjonalność do jednej funkcji lub klasy, którą można wielokrotnie używać.5. Testowanie: |

- Używaj funkcji i modułów do organizowania kodu. Funkcje i moduły mogą pomóc w podzieleniu kodu na mniejsze, bardziej łatwe do zrozumienia części.
- Stosuj wzorce projektowe. Wzorce projektowe to sprawdzone rozwiązania, które mogą pomóc Ci w tworzeniu bardziej elastycznego i niezawodnego kodu.
- Koduj w sposób modularny. Modularna konstrukcja kodu może ułatwić jego rozbudowę i modyfikację.

Stosowanie najlepszych praktyk podczas pisania kodu może pomóc Ci w tworzeniu oprogramowania o wyższej jakości, które jest łatwiejsze w utrzymaniu i bardziej niezawodne.

- Pisz testy jednostkowe, integracyjne i testy akceptacyjne, aby upewnić się, że twój kod działa poprawnie. Testowanie pomaga w wykrywaniu błędów wcześniej i utrzymaniu stabilności kodu.

6. Używanie czytelnego formatowania:

- Dbaj o czytelne formatowanie kodu, takie jak wcięcia, rozmieszczenie nawiasów i przejrzysty układ. To ułatwia innym osobom czytanie i zrozumienie twojego kodu.

7. Unikanie nadmiernego użycia globalnych zmiennych:

- Unikaj nadmiernego używania zmiennych globalnych, ponieważ mogą one sprawić, że kod staje się trudniejszy do zrozumienia i debugowania. Stosuj zasady hermetyzacji danych.

8. Optymalizacja i wydajność:

- Pisz wydajny kod, unikając zbędnych obliczeń, powtarzających się operacji i niepotrzebnych alokacji pamięci. Profilowanie kodu może pomóc zidentyfikować miejsca, które wymagają optymalizacji.

9. Zarządzanie błędami i wyjątkami:

- Obsługuj błędy i wyjątki w odpowiedni sposób, aby zapewnić, że twój program będzie bardziej niezawodny. Unikaj używania pustych bloków `try-catch`, jeśli nie jest to konieczne.

10. Używanie narzędzi kontroli wersji:

- Korzystaj z systemów kontroli wersji, takich jak Git, aby śledzić historię zmian w kodzie i ułatwić współpracę z innymi programistami.

11. Ciągłe doskonalenie:

- Nigdy nie przestawaj uczyć się i doskonalić swoje umiejętności programistyczne. Śledź nowe trendy, dobre praktyki i rozwijaj swoją wiedzę.

12. Ustalane standardy:

- Współpracuj z innymi członkami zespołu lub projektu nad ustaleniem i przestrzeganiem standardów kodowania. To pomaga utrzymać spójność w całym projekcie.

13. Przeglądy kodu:

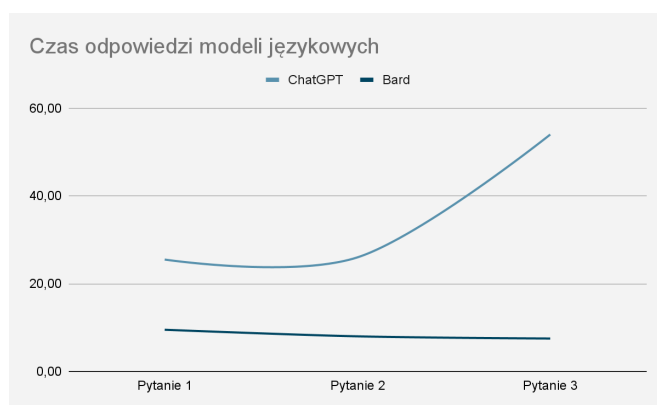
- Organizuj przeglądy kodu, w trakcie których inni członkowie zespołu oceniają twój kod. To pomaga w identyfikowaniu błędów i poprawianiu jakości kodu.

| | | |
|--|--|--|
| | | <p>Pisanie dobrego kodu to proces, który wymaga praktyki, cierpliwości i zaangażowania. Przestrzeganie tych najlepszych praktyk przyczyni się do utrzymania kodu w dobrej kondycji, ułatwi jego zrozumienie i poprawi jakość oprogramowania, które tworzysz.</p> |
|--|--|--|

Podsumowanie:

Obydwa modele piszą bardzo dobre teksty, a różnice są bardziej kosmetyczne które to można załatwić promptami np. "Odpowiedz w formie rozprawki." lub "Odpowiedz w formie notatki.", lecz bard czasem zapomina użyć jakiś synonimów lub popełnia błąd (Odpowiedź do pytania 3) "Pomognie" a prawdopodobnie powinno być *pomorze*. Są to błahostki ale jednak lepiej jest je wypunktować tym bardziej że modelowi od OpenAI nie zdarzyły się takie wpadki. Również należy przypomnieć że ChatGPT 3,5 ma zamkniętą bazę danych sięgającą do września 2021 r. w przeciwieństwie do barda który jak twierdzi "Jestem wciąż w fazie rozwoju i zawsze się uczę nowych rzeczy." co może oznaczać że chatbot googlea powinien być lepszy w aktualnych tematach. Jest jeszcze ChatGPT 4 który posiada nowsze bazy danych, dodatki i inne unowocześnieńia ale kosztuje 20\$/miesiąc co raczej nie jest do końca adekwatne w stosunku cena/nowe funkcje. Ostatnią rzeczą na którą warto zwrócić uwagę jest czas generacji tekstu który wygląda tak:

| | Google Bard | OpenAI ChatGPT (GPT-3.5) |
|--|-------------|--------------------------|
| Pytanie 1: Jakie są główne różnice między językami programowania kompilowanymi a interpretowanymi? | 9,5s | 25s |
| Pytanie 2: Czym jest zmienna w programowaniu i jakie są jej podstawowe cechy? | 8s | 26s |
| Pytanie 3: Jakie są najlepsze praktyki podczas pisania kodu? | 7,5s | 54s |



Widać że Bard jest wiele szybszy co pokazuje że posiada ogromny potencjał, i w przyszłości jeśli dogoni merytoryką model OpenAI może być najlepszym wśród tych darmowych chatbotów.