Akademia Ekonomiczno-Humanistyczna w Warszawie

Wydział Technologii Informatycznych

Praca magisterska

Krzysztof Małczak

studia niestacjonarne

nr albumu 53378

Metody oceny efektywności wielkich modeli językowych na przykładzie benchmarku HELPS

Promotor:
dr inż. Krzysztof Rychlicki-Kicior

Warszawa
marzec 2025

Słowa kluczowe:

Generative AI, Model Assesment, Large Language Models, LLM Evaluation, LLM Benchmark

Abstrakt

Dynamiczny rozwój generatywnej sztucznej inteligencji w formie Wielkich Modeli Językowych (LLM) oraz ich stale rosnąca ilość zastosowań sprawiają, że staje się ona nieodłącznym elementem ludzkiego życia. Niestety w obliczu lawiny LLM-ów i chatbotów opartych o techniki NLP pojawiających się niemal co tydzień, często z górnolotnymi deklaracjami na temat ich wydajności, trudno jest oddzielić technologie, które mają szansę wprost wpłynąć na poprawę jakości codziennego życia człowieka, od tych, które dla zwykłych użytkowników nie mają większego znaczenia. Niniejsza praca stanowi przegląd istniejących metod badania efektywności (tzw. ewaluacji) modeli językowych oraz ich zastosowania na przykładzie zbudowanego w ramach tej pracy benchmarku **HELPS** (**H**uman **E**veryday **L**ife **P**ractical **S**upport), który w przeciwieństwie do już istniejących tego typu rozwiązań nie skupia się na ocenie poprawności generowanych treści w pewnej ścisłej dziedzinie, a raczej ocenia jak bardzo pomocne są odpowiedzi modelu względem problemu przedstawionego przez użytkownika. W tym celu **HELPS** wykorzystuje zbiór problemów napotykanych na co dzień przez potencjalnych użytkowników tych technologii, który stworzony został na bazie ankiety, jednocześnie opierając swoją ideę ewaluacyjną na wzorcu 'LLM jako sędzia', co nie tylko pozwala na ocenę odpowiedzi na pytania otwarte ale również na pełną automatyzację oraz możliwość odtworzenia wyników na bazie udostępnionego kodu źródłowego benchmarku. Dodatkowo praca przedstawia wyniki benchmarku **HELPS** dla trzech spośród najpopularniejszych modeli (GPT-4, Gemini 1.5 Pro, Claude 3.5 Sonnet).

Spis treści

Spis rysunków

Spis tabel

Rozdział 1

Wstęp

Dynamiczny rozwój generatywnej sztucznej inteligencji oraz jej stale rosnące wdrożenie w wielu narzędziach codziennego użytku[1] czy też w różnych gałęziach przemysłu[2] sprawiają, że staje się ona nieodłącznym elementem ludzkiego życia.

Jako szczególny przypadek Generative AI można wyróżnić Wielkie Modele Językowe takie jak GPT-4, Gemini 1.5 Pro, Claude 3.5 Sonnet czy LLama3.3, które znajdują swoje zastosowanie nie tylko w biznesie ale coraz częściej są wykorzystywane przez przeciętnego użytkownika[3], dzięki łatwo dostępnym oraz przystępnym interfejsom konwersacyjnym jak ChatGPT, Gemini czy Claude. Pojawia się więc potrzeba zbadania ich pod kątem wartości użytkowej dla tej grupy docelowej.

1.1 Cel pracy

Celem pracy jest zwiększenie świadomości społecznej w kwestii tego czy Wielkie Modele Językowe w praktyce są w stanie wspomagać codzienne funkcjonowanie człowieka poprzez pomoc w procesach decyzyjnych oraz rozwiązywaniu takich problemów jak priorytetyzacja zadań, organizacja czasu czy wyciąganie wniosków. W tym celu wybrane spośród najpopularniejszych i najbardziej powszechnie dostępnych modeli poddane zostaną badaniu efektywności za pośrednictwem przygotowanego na bazie ankiety zbioru pytań, zadań i problemów oraz specjalnie zaimplementowanego systemu oceniającego - benchmarku **HELPS**.

Praca wprowadza teorie niezbędną do zrozumienia zagadnień związanych z ewaluacją treści generowanych przez Wielkie Modele Językowe, opisuje proces budowy zbioru problemów użytych do ewaluacji, uzasadnia sens implementacji oraz opisuje zasadę działania stworzonego w ramach niniejszej pracy benchmarku **HELPS**, jednocześnie omawiając jego wyniki dla wybranych modeli.

1.2 Podstawowe pojęcia i koncepty

W tym podrozdziale przedstawione zostaną pojęcia niezbędne do pełnego zrozumienia treści niniejszej pracy.

Generative AI

Generatywna Sztuczna inteligencja (ang. Generative Artificial Intelligence - GenAI) - to poddziedzina sztucznej inteligencji, której celem jest tworzenie nowych, oryginalnych treści. Modele generatywne uczą się wzorców i struktur z danych wejściowych w fazie treningu, takich jak tekst, obrazy czy dźwięki [27].

[1] Szacuje się, że do 2025 roku 750 milionów aplikacji będzie wykorzystywało modele językowe [72]

[2] Rozwiązania oparte o LLMy już teraz wykorzystywane są w takich gałęziach przemysłu jak e-commerce, marketing, edukacja, finanse czy nawet opieka medyczna [72]

[3] W okresie 07.2023 - 02.2024 zanotowano 18% wzrost liczby dorosłych w USA używających ChatGPT jako pomoc w pracy [70] [50]

Dataset

Zbiór danych (ang. Dataset) - ogólne określenie na zbiór danych, w domenie AI często odnosi się do zbioru danych treningowych lub testowych.

Model uczenia maszynowego (Model ML)

Model uczenia maszynowego (ang. Machine Learning Model, w skrócie model ML) - statystyczny model pozwalający na wykrycie wzorców oraz zależności w danych [8].

LLM

Wielki Model Językowy (ang. Large Language Model, w skrócie LLM) - rodzaj statystycznego modelu sztucznej inteligencji, trenowanego na ogromnych datasetach w celu przetwarzania języka naturalnego [22].

Kontekst

Kontekst (ang. context) - dodatkowe informacje zawarte w zapytaniu (patrz podrozdział 2.1) do LLM-a [7].

Halucynacje

Halucynacje (ang. hallucination) - zjawisko, w którym LLM generuje wiarygodne, ale niepoparte faktami treści [39]. Halucynacje mogą wystąpić zwłaszcza w sytuacjach, gdy model musi operować na informacjach nie zawartych w kontekście, które nie były uwzględnione w treningowym zbiorze danych.

AGI

AGI (ang. Artificial General Intelligence) - odnosi się do hipotetycznego systemu posiadającego inteligencję podobną do ludzkiej [31].

NLP

NLP (ang. Natural Language Processing - przetwarzanie języka naturalnego) - gałąź dziedziny sztucznej inteligencji, która skupia się na interakcji człowieka z komputerem poprzez język naturalny. NLP pozwala maszynom na ”rozumienie”, interpretację oraz generowanie języka naturalnego [25].

NLG

NLG (ang. Natural Language Generation) - dziedzina skoncentrowana na generowaniu języka naturalnego na bazie zorganizowanego wejścia [26].

Dane syntetyczne

Dane wygenerowane za pomocą technik NLG np. przy użyciu LLM-a.

Fine-Tuning

Fine-Tuning (ang. fine-tuning - dostrajanie) - proces dostrajania, adaptacji wcześniej wytrenowanego modelu do wykonywania zadań pewnej klasy. Inaczej, jest to dopasowanie modelu do specyficznej, na ogół wąskiej dziedziny problemu, która stanowi wyzwanie dla modelu wstępnie wytrenowanego na szerokiej bazie danych. Zwykle polega na zmianie wartości parametrów modelu, poprzez częściowe dotrenowanie modelu za pomocą par w postaci: `<INPUT, OUTPUT>`, które stanowią przykłady oczekiwanego zachowania modelu [65] [41].

Prompt Engineering

Dziedzina zajmująca się opracowywaniem oraz optymalizacją zapytań wysyłanych do LLM-ów [23].

Rozdział 2

Wprowadzenie do Prompt Engineeringu

Ten rozdział skupia się na wprowadzeniu pojęć z dziedziny prompt engineeringu niezbędnych do dogłębnego zrozumienia części praktycznej niniejszej pracy. Każdy przykład promptu i opcjonalnego wyniku przedstawiony jest jak na rysunku 2.1, przy czym kategorie i rodzaje promptów omawiane są w podrozdziale 2.4.

PROMPT:
[*Kategoria, Rodzaj*]

TREŚĆ PROMPTU

OUTPUT (opcjonalny):

TREŚĆ ODPOWIEDZI

Rysunek 2.1: Opis

2.1 Prompt

Prompt to pojęcie ogólnie odnoszące się do treści podawanej jako wejście do Modelu Generatywnej Sztucznej Inteligencji [51, 61], która ma na celu sterować jego działaniem. W zależności od zastosowania może przyjmować różne formy zaczynając od prostego pytania a kończąc na szczegółowym opisie konkretnego zadania (np. opis obrazu, który ma zostać wygenerowany) [6]. Może składać się z tekstu, obrazu, dźwięku lub innych mediów [61] w zależności od modelu.

W kontekście Wielkich Modeli Językowych pojęcie promptu zazwyczaj odnosi się do tekstu w formie stwierdzenia, pytania lub zestawu instrukcji mających na celu wywołanie określonej odpowiedzi. Prompt może być przekazany do modelu w celu dostosowania zachowania modelu do określonych wymagań (tzw. system prompt) lub w formie zadania zlecanego przez użytkownika (user prompt). Na rysunku 2.2 przedstawiono jeden z najbardziej podstawowych przykładów promptu, zaś rysunek 2.3 przedstawia tzw. prompt template [66] (ang. template - szablon), w którym zmienne umieszcza się w nawiasach klamrowych - jest to zapis często używany w celu uproszczenia rzeczywistej zawartości promptu.

PROMPT:
[*ICL, instruction*]

> Napisz e-mail składający się z trzech akapitów na potrzeby kampanii marketingowej dla firmy księgowej.

Rysunek 2.2: Podstawowy prompt

PROMPT:
[*ICL, instruction*]

> Sklasyfikuj jako pozytywny lub negatywny: {KOMENTARZ}

Rysunek 2.3: Prompt template [66]

2.2 Komponenty promptu

Mimo tego, że treść promptu przekazywanego do Wielkiego Modelu Językowego jest dowolna, to istnieją pewne komponenty, które można spotkać w większości przykładów.

2.2.1 Dyrektywa

Dyrektywa (ang. directive) to element promptu mający na celu 'zachęcenie' modelu do działania [62]. Występuje zazwyczaj w formie instrukcji lub pytania. Jej zadaniem jest określenie celu. Często w literaturze nazywana jest *intent* (ang. intent - zamiar) [61]. Wśród dyrektyw można wyróżnić dwa rodzaje: jawne (rysunek 2.4) i niejawne (rysunek 2.5).

PROMPT:
[*ICL, instruction*]

> Podaj pięć wartych przeczytania książek.

Rysunek 2.4: Prompt zawierający dyrektywę jawną

PROMPT:
[*ICL, one-shot*]

> Noc: Night
> Poranek:

Rysunek 2.5: Prompt zawierający dyrektywę niejawną (intencja: tłumaczenie słowa 'Poranek' na język angielski)

Tak jak wskazuje na to nazwa w przypadku dyrektywy jawnej cel określony jest wprost, zaś w przypadku niejawnej oczekiwanym jest aby to model wywnioskował intencję promptu.

2.2.2 Przykłady

Przykłady (ang. examples) służą jako demonstracja operacji lub zadania (i jego rozwiązania) jakie ma wykonać model. W literaturze przykłady określane są mianem shotów (ang. shot - strzał) lub examplars [61, 15]. Wtedy prompt może zostać określony jako n-shot, gdzie n jest liczbą przykładów (patrz rysunki 2.6 i 2.7).

PROMPT:
[*ICL, one-shot*]

> drapieżny kot o pomarańczowym futrze w czarne pasy: tygrys
>
> król wszystkich zwierząt:

Rysunek 2.6: One-shot prompt na przykładzie odgadywania zwierząt po opisie

PROMPT:
[*ICL, few-shot*]

> Jasno: Ciemno
> Ciepło: Zimno
> Wesoło: Smutno
> Szybko:

Rysunek 2.7: Few-shot prompt na przykładzie antonimów

Warto zaznaczyć, że prompty skonstruowane tak jak na rysunkach 2.6 i 2.7 w literaturze często określane są mianem promptów spełniających paradygmat wypełniania formularza (ang. form-filling paradigm) [48], ponieważ z perspektywy LLM-a jego zadaniem jest dokończenie kolejnego przykładu tak jakby wypełniał swego rodzaju formularz.

2.2.3 Formatowanie wyjścia

Formatowanie wyjścia (ang. output formatting) odnosi się do elementu promptu, który określa jaki ma być format wygenerowanej treści [61] np. CSV, JSON, XML, Markdown itp. przykład przedstawiono na rysunku 2.8. W praktyce określenie formatowania wyjścia często odbywa się poprzez podanie przykładów, szczególnie gdy pożądany format nie jest istniejącym standardem (patrz rysunek 2.9).

PROMPT:
[*ICL, instruction*]

Imię,Nazwisko,Kraj
Krzysztof,Małczak,Polska
Przedstaw powyższy tekst w formacie JSON

Rysunek 2.8: Prompt z formatowaniem wyjścia jako JSON

PROMPT:
[*ICL, few-shot*]

{AKAPIT}
przedstaw wszystkie rzeczowniki z powyższego akapitu w następującym formacie:
- rzeczownik1
- rzeczownik2
- rzeczownik3

Rysunek 2.9: Prompt z formatowaniem wyjścia na bazie przykładów

2.3 Prompt Engineering

To relatywnie nowa dyscyplina zajmująca się opracowywaniem i optymalizacją promptów w celu efektywnego wykorzystania Wielkich Modeli Językowych [23]. W praktyce można ją sprowadzić do iteracyjnego procesu rozwoju oraz dostosowywania promptu [61] w celu uzyskania pożądanego wyniku generowania treści. Proces ten może być zautomatyzowany lub manualny [61].

Techniki prompt engineeringu wykorzystywane są przez badaczy w celu podnoszenia kompetencji LLM-ów w odpowiadaniu na pytania, rozumowaniu dedukcyjnym jak i arytmetycznym [23] oraz innych złożonych zadaniach jak np. generowanie streszczeń.

Ta dyscyplina wykorzystywana jest również bezpośrednio przez inżynierów oprogramowania nie tylko w ramach wsparcia w generowaniu kodu ale przede wszystkim w budowaniu rozwiązań opartych o LLM-y takie jak np. systemy RAG [1].

Poza określaniem strategii iterowania na promptach w celu uzyskania najlepszych wyników [61], dyscyplina ta zajmuje się przede wszystkim definiowaniem podstawowych technik promptingu, czyli schematów określających w jaki sposób definiować, strukturyzować i sekwencjonować prompty [61].

2.4 Wybrane techniki promptingu

Techniką promptingu nazywamy pewien wzorzec określający w jaki sposób tworzyć i definiować prompty, w jaki sposób strukturyzować ich komponenty (patrz podrozdział 2.2) oraz jak dynamicznie sekwencjonować wiele promptów w celu uzyskania jak najlepszego wyniku. Techniki promptingu mogą polegać na logice warunkowej, w celu tworzenia rozgałęzień [61].

[1] RAG (ang. Retrieval Augmented Generation - generowanie wzbogacone wyszukiwaniem) - wzorzec, w którym prompt wysyłany do LLM-a wzbogacany jest informacjami z pewnej bazy wiedzy, która nie zawiera się w jego treningowym zbiorze danych [44]

Ten podrozdział skupia się na technikach promptingu tekstowego dzieląc je na trzy główne kategorie. Ze względu na stale rosnącą ilość różnych metod promptingu podrozdział ten ogranicza się do omówienia jedynie tych, które są istotne dla niniejszej pracy.

2.4.1 In-Context Learning (ICL)

Jest to kategoria, w której prompty konstruuje się w taki sposób, aby model był w stanie 'nauczyć się'[2] rozwiązywać zadany problem na bazie przykładów (rysunek 2.7) i instrukcji (rysunek 2.3) zawartych w propmpcie bez konieczności fine-tuningu modelu [56, 61]. Kontekst (ang. context) w tym przypadku oznacza informacje ogólnie zawarte w prompcie (niekoniecznie w formie przykładów).

Few-Shot Prompting

Tak jak wskazuje na to nazwa Few-Shot Prompting [15] jest techniką, w której model uczy się jak rozwiązać zadany problem na podstawie zaledwie kilku (ang. few - niewiele) przykładów [61] (patrz rysunek 2.10)

PROMPT:
[*ICL, few-shot*]

2+2: cztery
4+5: dziewięć
8+0:

Rysunek 2.10: ICL few-shot prompt

2.4.2 Zero-shot

Jest to kategoria, w której prompty nie zawierają żadnych przykładów (patrz podrozdział 2.2.2) ani konkretnych instrukcji [61] w kontekście.

Role Prompting

Role Prompting [77] znany również jako persona prompting [76] jest techniką polegającą na przypisaniu modelowi pewnej osobowości, co może pozwalać na generowanie treści wyższej jakości w przypadku zadań otwartych [58, 61] (patrz rysunki 2.11 oraz 2.13).

PROMPT:
[*zero-shot, role/persona prompting*]

Jesteś copywriterem z wieloletnim doświadczeniem.
Napisz tekst na stronę internetową kawiarni.

Rysunek 2.11: Prompt z określeniem persony

Style Prompting

Style prompting [49] to technika, w której treść promptu określa styl, ton, rodzaj języka itp. Role prompting (patrz podrozdział 2.4.2) może pozwalać na uzyskanie podobnych wyników w zależności od przypisanej persony [61]. Przykłady przedstawiono na rysunkach 2.12 oraz 2.13.

[2] w tym wypadku pojęcie 'uczenia się' mimo, że używane w literaturze jest mylące, ponieważ ICL może być jedynie specyfikacją zadania, którego model nauczył się już w trakcie treningu np. formatowanie jako JSON (rysunek 2.8)

PROMPT:
[*zero-shot, style prompting*]

> Napisz abstrakt pracy magisterskiej. Użyj profesjonalnego, akademickiego języka.

Rysunek 2.12: Prompt z określeniem stylu wypowiedzi

PROMPT:
[*zero-shot, role/persona prompting*]

> Jesteś pracownikiem akademickim. Napisz abstrakt pracy magisterskiej.

Rysunek 2.13: Prompt z określeniem persony w celu wymuszenia stylu wypowiedzi. Alternatywa dla style promptingu

2.4.3 Thought Generation

Jest to kategoria obejmująca szeroką gamę technik promptingu, w których oprócz zadanego problemu wymagane jest od LLM-a przedstawienie uzasadnienia swojej odpowiedzi [80, 61] - tzw. 'toku myśli' stąd też nazwa (ang. Thought Generation - generowanie myśli).

Chain-of-Thought (CoT) Prompting

Rozwiązując złożony problem (np. zadanie matematyczne wymagające wielu kroków) ludzie bardzo często rozkładają go na etapy pośrednie [79], technika CoT stara się zasymulować właśnie taki proces.

Chain-of-Thought [79] lub Chain-of-Thoughts [71] prompting jest metodą, polegająca na generowaniu sekwencji pośrednich kroków rozumowania, dzięki przykładom sekwencji myśli zawartym w prompcie [79]. Innymi słowy technika CoT bezpośrednio wykorzystuje Few-Shot prompting (patrz podrozdział 2.4.1), za pomocą którego wymusza na LLM-ie przedstawienie swojego 'procesu myślowego' zanim dostarczy ostateczną odpowiedź [79, 61, 81]. Każdy przykład zawarty w prompcie składa się z pytania oraz łańcucha rozumowania z oczekiwaną odpowiedzią [81].

CoT znacząco zwiększa efektywność LLM-ów w rozwiązywaniu problemów matematycznych oraz problemów wymagających umiejętności rozumowania dedukcyjnego [79], dzięki zasugerowanemu w prompcie mechanizmowi dekompozycji problemu [79]. Przykład tej techniki przedstawiono na rysunku 2.14.

PROMPT:
[*thought generation, few-shot CoT (one-shot CoT)*]

> **PYT**: Jacek ma 5 piłek tenisowych. Kupił 2 dodatkowe puszki piłek tenisowych. Każda puszka zawiera 3 piłki. Ile piłek tenisowych ma teraz Jacek?
>
> **ODP**: Na początku Jacek miał 5 piłek. W jednej puszcze znajdują się 3 piłki, więc w dwóch puszkach jest łącznie 6 piłek. 5 + 6 = 11. Odpowiedź to 11.
>
> **PYT**: W stołówce znajdowały się 23 jabłka. 20 jabłek zostało zużytych do przygotowania śniadania oraz 6 jabłek zostało dokupionych. Ile jabłek jest teraz w stołówce?

OUTPUT:

> **ODP**: Na początku w stołówce znajdowały się 23 jabłka. 20 zostało zużytych do przygotowania śniadania, czyli zostało im 23 - 20 = 3. Następnie 6 jabłek zostało dokupionych, więc zostało ich 3 + 6 = 9. Odpowiedź to 9.

Rysunek 2.14: One-shot CoT prompt. Wyróżnione fragmenty to łańcuchy rozumowania charakterystyczne dla CoT. Na podstawie [79]

Zero-shot Chain-of-Thought

Nie podając LLM-owi żadnych przykładowych łańcuchów rozumowania, jednocześnie w inny sposób 'zachęcając' go do dekompozycji problemu np. przy użyciu takiej frazy jak "Zastanówmy się krok po kroku"(ang. Let's think step by step) możemy wykonać tzw. Zero-Shot CoT [43] również osiągający przyzwoite wyniki [81]. W przeciwieństwie do Few-Shot CoT, w przypadku którego format odpowiedzi zdefiniowany jest w ramach przykładów, Zero-Shot CoT wymaga dodatkowego zapytania w celu wydobycia ostatecznej odpowiedzi. Oznacza to, że tę technikę można podzielić na dwa etapy. Pierwszym jest ekstrakcja rozumowania (rysunek 2.15), zaś drugim ekstrakcja ostatecznej odpowiedzi na bazie rozumowania (rysunek 2.16) [43].

PROMPT:
[*thought generation, zero-shot CoT (reasoning extraction)*]

PYT: Janek w ciągu minuty zadaje średnio 25 ciosów. Walka trwa 5 rund. Każda runda trwa 3 minuty. Ile ciosów zada Janek?

ODP: Zastanówmy się krok po kroku.

OUTPUT:

W jedną minutę Janek zadaje 25 ciosów.
W 3 minuty Janek zadaje 3 * 25 = 75 ciosów.
W 5 rund Janek rzuca 5 * 75 = 375 ciosów

Rysunek 2.15: Zero-Shot CoT - ekstrakcja rozumowania. Na podstawie [43]

PROMPT:
[*thought generation, zero-shot CoT (answer extraction)*]

PYT: Janek w ciągu minuty zadaje średnio 25 ciosów. Walka trwa 5 rund. Każda runda trwa 3 minuty. Ile ciosów zada Janek?

ODP: Zastanówmy się krok po kroku.
W jedną minutę Janek zadaje 25 ciosów.
W 3 minuty Janek zadaje 3 * 25 = 75 ciosów.
W 5 rund Janek rzuca 5 * 75 = 375 ciosów
Zatem odpowiedź (cyframi arabskimi) to:

OUTPUT:

375

Rysunek 2.16: Zero-Shot CoT - ekstrakcja odpowiedzi z wygenerowanego rozumowania. Na podstawie [43]

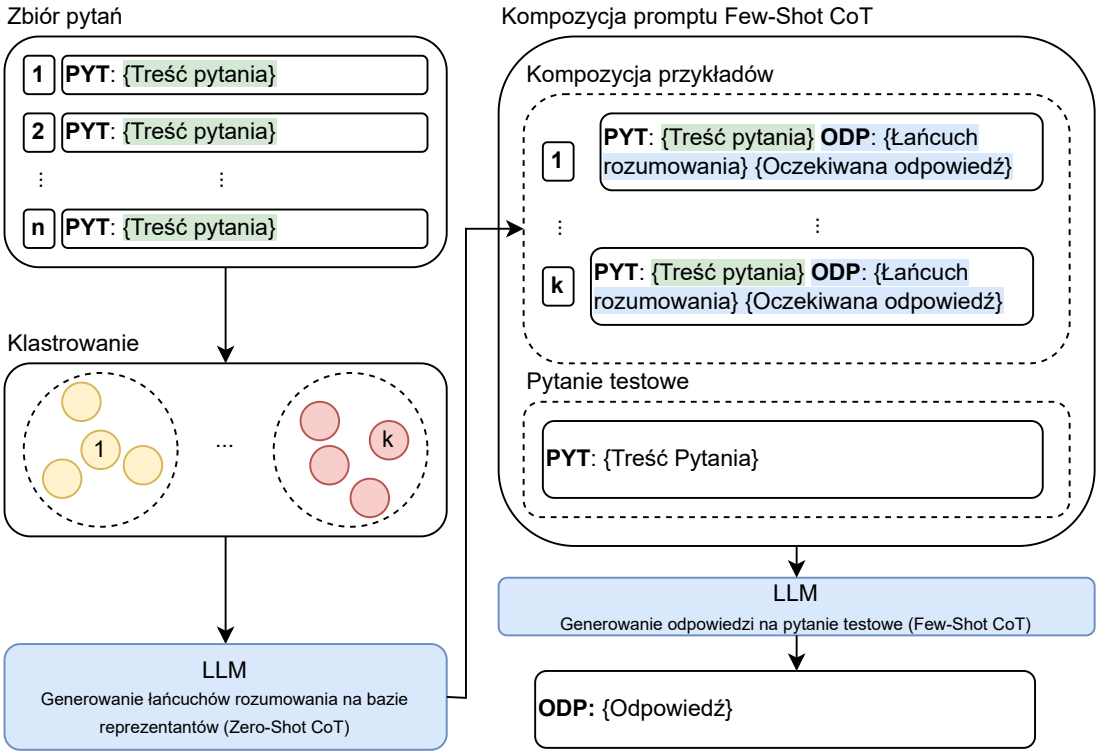
Automatic Chain-of-Thought (Auto CoT) prompting

Wykazano, że Few-Shot CoT (patrz podrozdział 2.4.3) osiąga lepsze wyniki w generowaniu treści niż Zero-Shot CoT [79, 43, 81]. Niestety w przypadku Few-Shot CoT konieczne jest ręczne zdefiniowanie przykładów[3], czyli pytań wraz z łańcuchem rozumowania i poprawną odpowiedzią, co w przypadku większości złożonych problemów jest nietrywialne [81] czyniąc tę technikę bardzo kosztowną. Z tego powodu większość mechanizmów mających na celu automatyzację przy jednoczesnym zachowaniu poprawności i wysokiego poziomu generowanych treści nie jest w stanie wykorzystać tej metody.

Auto Chain-of-Thought ma na celu wyeliminowanie potrzeby ręcznego przygotowywania łańcuchów rozumowania wraz z odpowiedziami poprzez wygenerowanie ich dla każdej klasy pytań[4] za pomocą techniki Zero-Shot CoT (patrz podrozdział 2.4.3). Następnie wykonywana jest kompozycja pytania, łańcucha rozumowania oraz odpowiedzi w celu uzyskania przykładów. Ostatnim krokiem jest zastosowanie Few-Shot CoT (patrz podrozdział 2.4.3) przy użyciu skomponowanych przykładów [81]. Na rysunku 2.17 wizualnie przedstawiono zasadę działania tej techniki. Warto zwrócić uwagę, że w przypadku wykorzystania Auto

[3] z tego powodu w literaturze można tę technikę spotkać pod nazwą Manual-CoT (ang. manual - ręczny, manualny) [81]
[4] generowanie dla danej klasy odbywa się na podstawie jednego jej reprezentanta [81]

CoT do generowania rozwiązań dla problemów jednej klasy etap ich klastrowania może zostać pominięty zaś jej reprezentant może zostać wybrany ręcznie.



Rysunek 2.17: Schemat działania Auto CoT. Na podstawie: [81]

Rozdział 3

Metody oceny efektywności Wielkich Modeli Językowych

Wielkie Modele Językowe stale zyskują na popularności zarówno w środowisku akademickim jak i w różnych domenach przemysłowych [12, 78, 82]. Ze względu na ich rosnącą ilość zastosowań w codziennym życiu ludzi oraz bardziej krytycznych dziedzinach jak medycyna czy prawo, analiza ich efektywności, poprawności i przydatności znacznie zwiększa swoje znaczenie.

Ocena jakości tekstów generowanych maszynowo od dawna stanowi wyzwanie w dziedzinie przetwarzania języka naturalnego, zaś dziś pełni rolę krytycznego filaru niezbędnego do zrozumienia właściwości i zachowania LLM-ów [42, 46, 18, 86, 21, 36]. Ten rozdział skupia się na metodykach związanych z oceną Wielkich Modeli Językowych zaczynając od ogólnego przedstawienia konceptu ewaluacji a kończąc na jego szczególnych zastosowaniach porównawczych. Omawiane są podstawowe metryki służące do oceniania treści generowanych przez modele, różne strategie implementowania tych metryk, zarówno z oraz bez wykorzystania w tych celach sztucznej inteligencji.

3.1 Ewaluacja LLM-ów

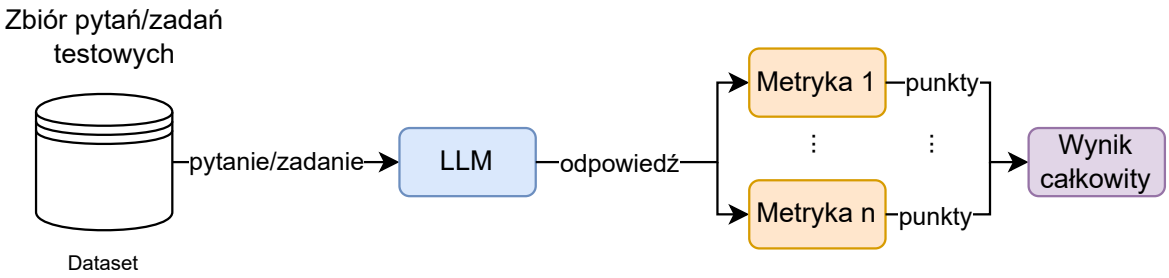
Patrząc na obecne badania [16] obietnica osiągnięcia AGI staje się coraz bardziej realna [18]. W przeciwieństwie do poprzednich modeli, tworzonych i trenowanych z myślą o konkretnej klasie zadań, obecnie LLM-y wykazują wysoką wydajność w rozwiązywaniu szerokiej gamy problemów takich jak ogólne i specyficzne dla danej domeny zadania związane z językiem naturalnym [18], przez co pojawia się szereg specjalistów różnych dziedzin, którzy chcą wykorzystywać LLM-y jako narzędzie do pracy. Jednakże bardzo często te jednostki (np. personel medyczny) wymagają ogromnej precyzji, przez co modele językowe muszą być badane pod kątem efektywności. Podstawą takich badań jest szeroko pojęty koncept ewaluacji.

Ewaluacja Wielkich Modeli Językowych odnosi się ogólnie do procesu oceny ich możliwości, zgodności z wartościami ludzkimi [32] ale przede wszystkim skuteczności w realizacji określonych zadań, takich jak generowanie tekstu, tłumaczenie, klasyfikacja danych, streszczanie, czy bardziej specyficznych [32] np. odpowiadanie na zapytania klienta w roli asystenta sklepu internetowego. Ogólny mechanizm ewaluacji LLM-ów przedstawiono na rysunku 3.1.

3.2 Potrzeba ewaluacji: Motywacje i Korzyści

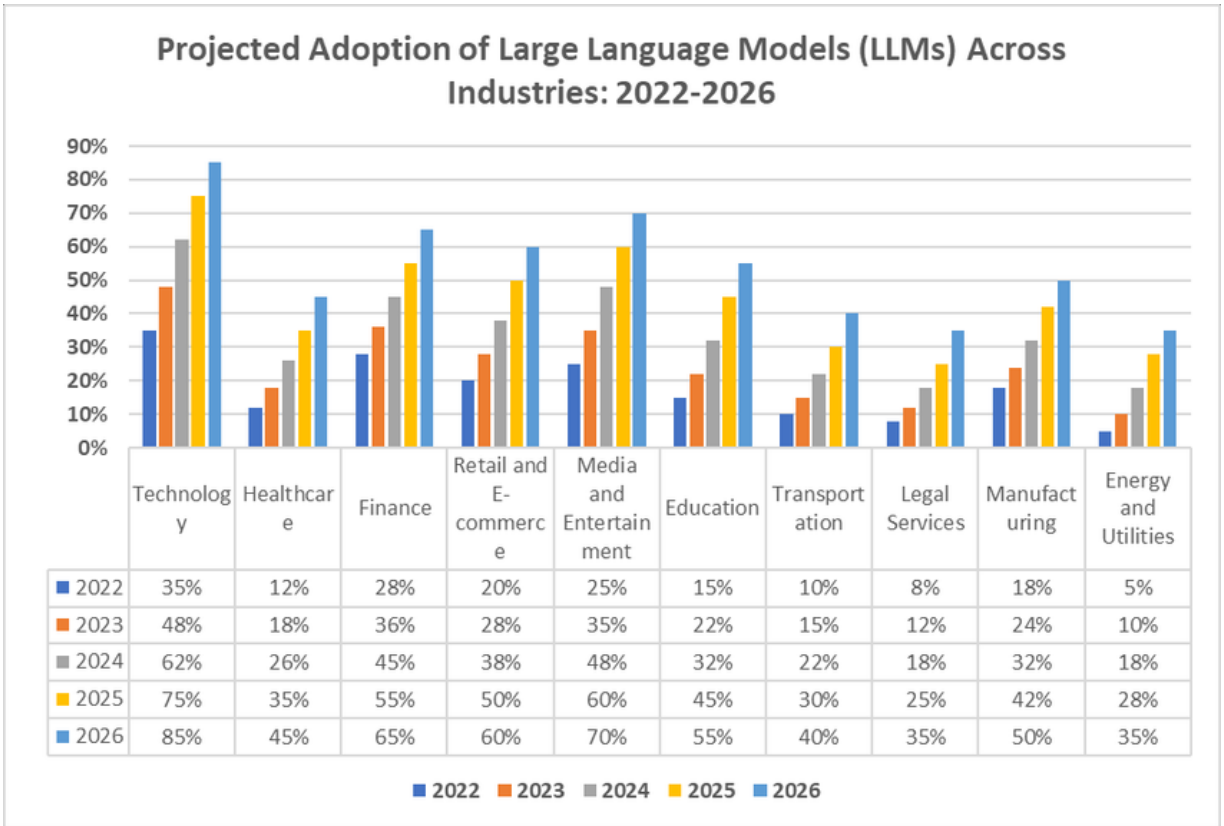
Wielkie Modele Językowe jako sztuczna inteligencja stworzona do celów nieodłącznie związanych z NLP, musi być stale monitorowana. Ewaluacja jest kluczowym etapem w rozwoju i tworzeniu LLM-ów oraz rozwiązań o nich opartych, ponieważ umożliwia zrozumienie potencjalnych zastosowań, jak również mocnych oraz słabych stron konkretnych modeli [32].

Jak przedstawiono na rysunku 3.2 rozwiązania oparte o LLM-y coraz częściej znajdują swoje zastosowanie w niezwykle ważnych sektorach jak medycyna czy finanse [84] [45], gdzie niezawodność oraz bezpieczeństwo są najistotniejszymi aspektami [18]. Dzięki odpowiednio dobranemu zbiorowi danych testowych



Rysunek 3.1: Ogólny mechanizm ewaluacji LLM-ów

oraz stosowanej strategii ewaluacji modele lub systemy o nie oparte mogą być poddawane odpowiedniemu fine-tuningowi, dzięki czemu możliwe jest zwiększenie rzetelności generowanych treści.



	Technology	Healthcare	Finance	Retail and E-commerce	Media and Entertainment	Education	Transportation	Legal Services	Manufacturing	Energy and Utilities
2022	35%	12%	28%	20%	25%	15%	10%	8%	18%	5%
2023	48%	18%	36%	28%	35%	22%	15%	12%	24%	10%
2024	62%	26%	45%	38%	48%	32%	22%	18%	32%	18%
2025	75%	35%	55%	50%	60%	45%	30%	25%	42%	28%
2026	85%	45%	65%	60%	70%	55%	40%	35%	50%	35%

Rysunek 3.2: Przewidywane wdrożenie Wielkich Modeli Językowych w wybranych branżach do 2026r. Źródło: [55]

Dodatkowo naturalnym efektem ubocznym ewaluacji jest szansa na optymalizację generowanego tekstu oraz identyfikację obszarów, w których modele mogą być podatne na błędy i halucynacje.

Potrzeba ewaluacji wynika również z różnorodności zastosowań LLM-ów – modele te są wykorzystywane zarówno w prostych chatbotach, jak i w bardziej zaawansowanych aplikacjach takich jak systemy wspierające procesy decyzyjne. W takich przypadkach niezbędne jest dokładne zrozumienie poziomu ich trafności, zaufania, wiarygodności [32] itp., które badane są przy użyciu specjalnych metryk.

3.3 Metryki ewaluacyjne

Badając treści generowane przez LLM-y niezbędna jest możliwość kwantyfikacji, w celu ocenienia czy dana treść jest wystarczająco dobra [40] dla danego przypadku użycia. Powszechną praktyką jest nadawanie takim treściom pewnej ilości punktów (patrz rysunek 3.1), do czego wykorzystywane są metryki ewaluacyjne wraz z odpowiednim mechanizmem punktowania (ang. scorer) - więcej w podrozdziale 3.5. W literaturze dla prostoty same mechanizmy punktowania często nazywane są metrykami, gdyż są z nimi nieodłącznie związane. Istnieje wiele różnych metryk wykorzystywanych do ewaluacji LLM-ów, poza popularnymi, które zostały przedstawione w tabeli 3.1 często pojawia się potrzeba definiowania metryk wymagających niestandardowych kryteriów oceny w zależności od przypadku użycia np. ocena generowanych streszczeń [40].

Metryka	Opis
Adekwatność (ang. Relevancy)	Określa, czy wygenerowana odpowiedź odnosi się do danych wejściowych w sposób informacyjny i zwięzły
Poprawność (ang. Correctness)	Określa, czy wygenerowana odpowiedź jest faktycznie poprawna w oparciu o pewną prawdę (tzw. ground truth)
Halucynacja (ang. Hallucination)	Określa, czy wygenerowana odpowiedź zawiera fałszywe informacje
Toksyczność (ang. Toxicity)	Określa, czy wygenerowana odpowiedź zawiera szkodliwe lub obraźliwe treści

Tabela 3.1: Zestawienie wybranych, najbardziej powszechnych metryk do ewaluacji treści generowanych przez LLM-y

3.4 Metody ewaluacji

Uwzględniając aktorów biorących udział w procesie ewaluacji, jej obecnie istniejące metody można podzielić na dwie podstawowe kategorie:

- Ewaluacja dokonywana przez człowieka
- Ewaluacja automatyczna

3.4.1 Ewaluacja ludzka (ang. Human Evaluation)

Ewaluacja wygenerowanej treści dokonywana przez człowieka (tzw. anotatora[1]) polega na ręcznym przypisywaniu ilości punktów dla każdej z uwzględnianych metryk. Naturalnie w przeciwieństwie do ewaluacji zautomatyzowanej cechuje się obniżoną możliwością skalowania, ponieważ wymaga aktywnego udziału i wysokiego nakładu pracy człowieka, przez co wydłuża się jej czas oraz koszt ale również wprowadza się czynnik subiektywności [32, 18, 74, 53, 9, 24] - ocena jest subiektywna w zależności od oceniającego. Dodatkowym problemem tej metody jest fakt, iż ludzie mają tendencję do postrzegania stanowczych treści jako bardziej zgodnych z faktami, niezależnie od informacji w nich zawartych [37]. Jednakże mimo wspomnianej subiektywności, podatności i potencjalnej omylności anotatorów ewaluacja dokonywana przez człowieka pozostaje konsekwentnie dominującą metodą ze względu na ludzką zdolność uwzględniania subtelnych niuansów w generowanych treściach oraz docenianie takich aspektów jak zwięzłość czy kultura wypowiedzi [42].

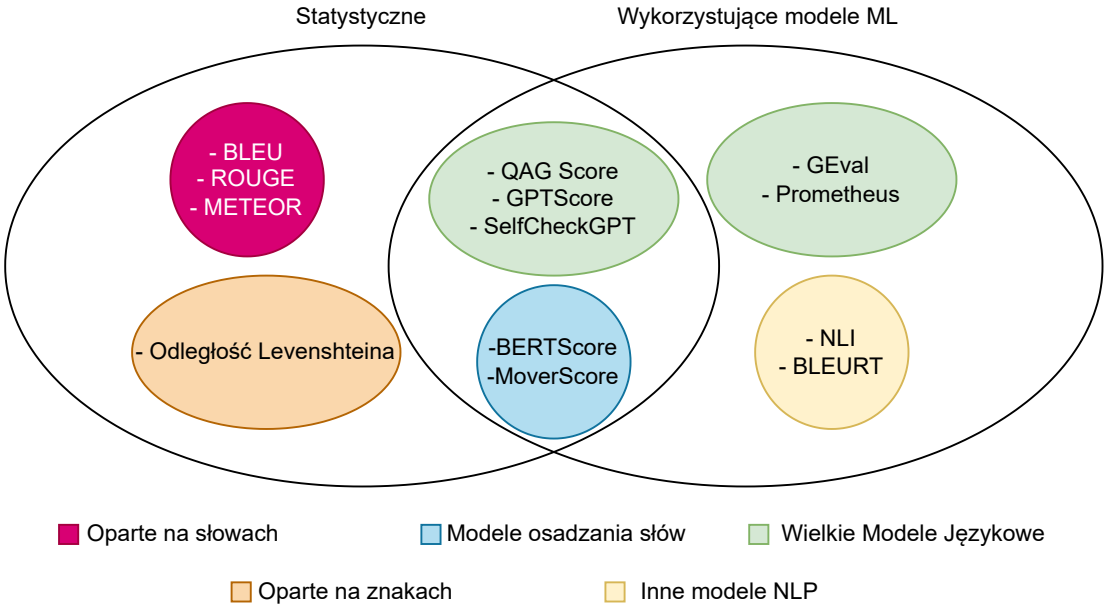
[1] ang. to annotate - dodawać adnotacje, etykietować

3.4.2 Ewaluacja zautomatyzowana (ang. Automatic Evaluation)

Ze względu na ilość manualnej pracy wymaganej w przypadku ewaluacji ludzkiej oraz idącymi za tym kosztami i czasem ewaluacja zautomatyzowana staje się obecnie bardziej powszechna, szczególnie w zastosowaniach biznesowych. Wszystkie strategie pozwalające na zautomatyzowaną ocenę generowanych treści dążą do osiągnięcia możliwie wysokiej zgodności z ocenami ludzkimi (ewaluacją ludzką), z tego powodu wyniki ewaluacji zautomatyzowanej powinny być poddawane ludzkiemu nadzorowi, przynajmniej w ograniczonym stopniu [11, 37].

3.5 Mechanizmy punktowania w ewaluacji zautomatyzowanej

Jedną z zalet ewaluacji zautomatyzowanej jest brak czynnika subiektywności, dzięki czemu możliwy jest nawet pewien stopień standaryzacji [32, 18]. Jednakże ocena pod kątem takich kryteriów jak trafność, czy przydatność (patrz tabela 3.1) wymaga możliwości analizy języka naturalnego lub chociaż jej symulacji przy użyciu metod statystycznych. Z tego powodu do automatycznego punktowania generowanych treści dla takich metryk często wykorzystuje się inny model uczenia maszynowego. Wprowadza to naturalny podział wśród mechanizmów punktowania wykorzystywanych w ewaluacji zautomatyzowanej, jak przedstawiono to na rysunku 3.3.



Rysunek 3.3: Mechanizmy punktowania wykorzystywane w zautomatyzowanej ewaluacji LLM-ów. Na podstawie: [40]

3.5.1 Statystyczne

Mimo obliczeniowej niezawodności czysto statystycznych mechanizmów punktowania, nie są one wystarczające do oceny zazwyczaj długich i złożonych treści generowanych przez LLM-y w przypadku większości istotnych metryk. Wynika to głownie z faktu, że mechanizmy te opierają się na porównywaniu treści wygenerowanej z treścią oczekiwaną, co oznacza, że w zbiorze danych wykorzystywanych do ewaluacji dla każdego wejścia testowego musi być zdefiniowane oczekiwane wyjście - tak zwana referencja [83]. Wynika z tego, że metody te nie nadają się do oceny odpowiedzi wygenerowanych np. na pytania otwarte, dla

których taka referencja nie może być jednoznacznie zdefiniowana [83]. Dodatkowo metody te nie biorą pod uwagę semantyki [2] tekstów, mają ogromne ograniczenia w porównywaniu tekstów wymagających umiejętności rozumowania dedukcyjnego [40]. Ponadto poziom zaawansowania generowanych tekstów jest na tyle wysoki, że porównywanie ich na podstawie cech powierzchniowych może być niemiarodajne [85, 29].

BLEU (BiLingual Evaluation Understudy)

Pozwala ocenić wygenerowaną treść na podstawie pewnej prawdy (tzw. referencji) czyli oczekiwanego wyniku. Obliczana jest precyzja dla każdego n-gramu (ciągu n kolejnych słów), porównując wygenerowaną treść z jedną lub kilkoma referencjami przygotowanymi przez ludzi. Precyzja oznacza, jaki procent n-gramów w wygenerowanej treści występuje również w referencji. Końcowy wynik oblicza się jako średnią geometryczną precyzji dla różnej długości n-gramów. Dodatkowo aby uniknąć zawyżania precyzji w przypadku krótkich treści stosuje się tzw. "brevity penalty"(ang. karę za zwięzłość). Wynik znajduje się w zakresie od 0 do 1 (lub w formie procentowej), gdzie wyższy wynik oznacza lepsze dopasowanie [54].

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

Głównie służy do oceny generowanych streszczeń i tłumaczeń maszynowych. Oblicza tzw. 'recall' (ang. odwołanie), który określa jak dobrze wygenerowana treść pokrywa się z oczekiwanym wynikiem, poprzez porównanie n-gramów (ciągów słów) wygenerowanego tekstu z n-gramami występującymi w tekście referencyjnym. Wynik ROUGE mieści się w przedziale od 0 do 1, gdzie wyższa wartość oznacza większe podobieństwo tekstów. Wysoki wynik wartości recall wskazuje na to, że model skutecznie uchwycił istotne informacje zawarte w treści referencyjnej [47].

METEOR (Metric for Evaluation of Translation with Explicit Ordering)

Ocenia wygenerowaną treść względem oczekiwanego wyniku (referencji) na podstawie wartości zarówno precyzji jak i recall, jednocześnie biorąc pod uwagę możliwą różnicę w kolejności słów. Dodatkowo wykorzystuje zewnętrzne lingwistyczne zbiory danych jak WordNet [3] w celu znajdowania synonimów. Ostatecznym wynikiem jest średnia harmoniczna wartości precyzji oraz recall obarczona karą za rozbieżności w kolejności lub użyciu innej formy słów [10].

Odległość Levenshteina (edycyjna)

Ocenia jak bardzo różni się treść wygenerowana z oczekiwanym wynikiem (referencją), poprzez określenie ile operacji (wstawienie, usunięcie lub zamiana znaku) trzeba wykonać, aby przekształcić jeden ciąg znaków w drugi [33].

3.5.2 Wykorzystujące modele uczenia maszynowego

Implementacja statystycznego mechanizmu punktowania, czy też proceduralnego algorytmu efektywnie oceniającego odpowiedzi na pytania otwarte w postaci języka naturalnego jest niezwykle wymagająca [83]. Z tego powodu w takich przypadkach do oceny generowanych tekstów często wykorzystuje się modele uczenia maszynowego. Patrząc na diagram z rysunku 3.5 wśród nich można wyróżnić te, które wykorzystują Wielkie Modele Językowe i te, które tego nie robią. Podobnie jak w przypadku czysto statystycznych mechanizmów, te które nie wykorzystują LLM-ów uzyskują niższy poziom korelacji z ocenami ludzkimi, szczególnie w kontekście zadań kreatywnych i pytań o charakterze otwartym [48, 40, 42].

NLI (Natural Language Inference)

Odnosi się do wykorzystania modeli lub technik NLP, które pozwalają na określenie jak dobrze LLM radzi sobie z określaniem relacji między zdaniami lub tekstami. Relacje klasyfikowane są zazwyczaj do trzech głównych kategorii:

[2] Semantyka - dział językoznawstwa, którego przedmiotem jest analiza znaczeń wyrazów [3]
[3] WordNet - baza angielskich słów oraz relacji między nimi zapoczątkowana przez George'a Millera w 1978r.

- ciągłość logiczna (ang. entilement) - zdanie kontynuuje ciąg logiczny swojego poprzednika (poprzedników)
- sprzeczność (ang. contradiction) - zdanie jest sprzeczne z pozostałymi w tekście
- neutralność (ang. neutral) - zdanie jest neutralne względem pozostałych w tekście

Ostateczny wynik pochodzi z przedziału od 0 (sprzeczność) do 1 (ciągłość logiczna) [13].

BLEURT (BiLingual Evaluation Understudy with Representations from Transformers)

Mechanizm wykorzystujący modele takie jak np. BERT [4] , które poddawane są kolejny fazom treningu, na początku na bazie danych syntetycznych a następnie danych etykietowanych przez ludzi. Ostatnia faza jest opcjonalna, lecz pozwala na uzyskanie lepszych wyników dla danego przypadku użycia. Oceny BLEURT są bardziej trafne niż w przypadku użycia BLEU lub ROUGE, nawet w przypadku niskiej jakości danych treningowych [64, 63, 2].

G-Eval

G-Eval [48] znacznie różni się od pozostałych metod, ponieważ nie funkcjonuje jedynie jako mechanizm punktowania ale raczej pełnoprawny framework ewaluacyjny [48], dokładnie określający poszczególne etapy prowadzące do uzyskania efektywnej oceny generowanych treści. G-Eval bezpośrednio wykorzystuje Wielki Model Językowy[5], którego zadaniem jest ocenienie wcześniej wygenerowanej treści przy użyciu zaawansowanych technik promptingu. Na rysunku 3.4 wizualnie przedstawiono mechanizm działania frameworka G-Eval.

W celu oceny wygenerowanej treści wykorzystywana jest strategia Auto CoT (patrz podrozdział 2.4.3), która na bazie wprowadzenia do zadania (ang. task introduction) oraz określonych kryteriów ewaluacyjnych (ang. evaluation criteria), pozwala na wygenerowanie szczegółowych kroków niezbędnych do wykonania w ramach procesu oceny. Następnie wygenerowane kroki wraz z wejściem z testowego zbioru danych w formie promptu spełniającego paradygmat wypełniania formularza (patrz podrozdział 2.2.2) ponownie trafiają do LLM-a, który jako odpowiedź zwraca ocenę treści. Opcjonalnie ostateczna ocena zwrócona przez LLM może zostać znormalizowana na bazie prawdopodobieństwa każdego ze zwracanych tokenów poprzez obliczenie sumy ważonej:

$$score = \sum_{i=1}^{n} p(s_i) \times s_i$$

gdzie s_i należy do zbioru możliwych ocen (np. od 1 do 5) $S = \{s_1, s_2, \ldots, s_n\}$ zdefiniowanych bezpośrednio w prompcie, zaś $p(s_i)$ to prawdopodobieństwo wystąpienia obliczane przez LLM dla każdego tokenu będącego wynikiem. Wykorzystanie G-Eval pozwala na uzyskanie wyników o znacznie wyższej korelacji z ocenami ludzkimi [48], co pozwala na bardziej niezawodną ewaluację zautomatyzowaną.

Prometheus

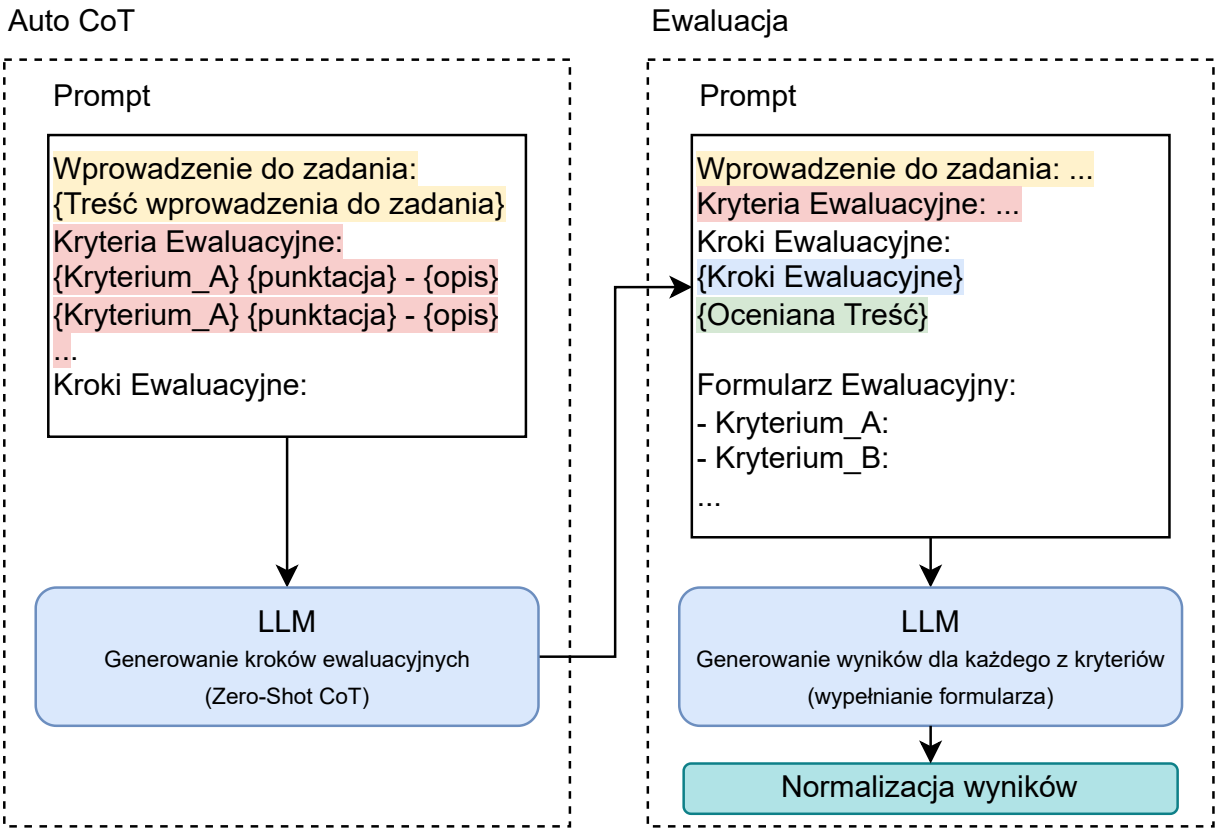
Prometheus [42] jest modelem językowym posiadającym $13 * 10^9$ parametrów, który pozwala na ewaluację treści na bazie tzw. rubryki punktów (ang. score rubric) oraz odpowiedzi referencyjnej, przekazywanej do niego w treści promptu ewaluacyjnego (patrz rysunek 3.5).

W rzeczywistości modelem bazowym jest LLama-2-chat, który poddany został fine-tuningowi z wykorzystaniem specjalnie przygotowanego datasetu nazwanego 'Feedback Collection', w którym każda para składa się z wejścia i wyjścia. Wejście stanowi:

- **Instrukcja**: treść zadania podawana przez użytkownika jako wejście do LLM-a
- **Treść do ewaluacji**: odpowiedź na zadanie zdefiniowane w instrukcji wygenerowana przez LLM, która ma zostać oceniona

[4] BERT (Bidirectional Encoder Representations from Transformers) - model językowy opracowany przez Google. Pozwala m.in. na: analizę sentymentu, rozpoznawanie emocji, generowanie tekstu, generowanie streszczeń [52].

[5] w oryginalnej implementacji wykorzystano modele GPT-4 oraz GPT-3.5 [48]



Rysunek 3.4: G-Eval - machanizm działania. Na podstawie: [48]

PROMPT:
[Prometheus evaluation prompt]

Instrukcja:
{TREŚĆ ZADANIA}
Rubryka Punktów: Czy odpowiedź wykazuje wrażliwość kulturową?

- 1 punkt: {KRYTERIA PRZYZNANIA}
- ...
- 5 punktów: {KRYTERIA PRZYZNANIA}

Odpowiedź referencyjna:
{TREŚĆ OCZEKIWANEJ ODPOWIEDZI}

Rysunek 3.5: Schemat promptu ewaluacyjnego dla modelu Prometheus. Na podstawie [42]

- **Dostosowana rubryka punktów**: składająca się z opisu kryteriów oraz wysokości punktów (od 1 do 5)
- **Odpowiedź referencyjna**: oczekiwana odpowiedź - lub też odpowiedź której model powinien przyznać 5 punktów

zaś wyjście stanowi:

- **Feedback** (ang. informacja zwrotna): uzasadnienie dlaczego treść poddawana ewaluacji otrzymałaby dany wynik - analogicznie do Chain-of-Thought (patrz podrozdział 2.4.3) - pozwala to na interpretacje uzyskanych wyników
- **Wynik**: ilość punktów (liczba naturalna między 1 a 5)

Warto zaznaczyć, że większość elementów zbioru 'Feedback Collection' zostało wygenerowanych przy użyciu modelu GPT-4, co pozwoliło na efektywne dostrojenie a w efekcie uzyskanie modelu, służącego jako ewaluator o zdolności oceny generowanych treści zbliżonej do GPT-4 [42], z tą różnicą, że Prometheus jest w pełni open-source'owym rozwiązaniem [42]. Podobnie do G-Eval efektywność ewaluacji przy użyciu modelu Prometheus nie jest zależna od zadania [40], jednakże między tymi rozwiązaniami istnieją znaczące różnice, które przedstawiono w tabeli 3.2.

G-Eval	Prometheus
G-Eval jest frameworkiem, który w swojej oryginalnej implementacji bezpośrednio wykorzystuje model GPT-3.5/GPT-4 jako podmiot oceniający generowane treści	Prometheus jest LLM-em poddanym treningowi, który czyni go wydajnym ewaluatorem
W przypadku G-Eval kryteria ewaluacyjne definiowane są już na etapie generowania kroków ewaluacyjnych w ramach procesu Auto CoT (patrz podrozdziały 3.4, 2.4.3)	W przypadku Prometheus kryteria ewaluacyjne w postaci rubryki punktów przekazywane są bezpośrednio do modelu jako część promptu
G-Eval charakteryzuje się znacznie większą elastycznością, gdyż nie wymaga definiowana oczekiwanej odpowiedzi (jednocześnie tego nie zabraniając), co czyni go lepszym kandydatem dla problemów otwartych	Prometheus wymaga zdefiniowania odpowiedzi referencyjnej, która pozwala mu na dokonanie oceny
G-Eval skupia się na dostarczeniu ewaluacji możliwie najbliższej do ewaluacji ludzkiej, bezpośrednio wykorzystując jeden z najnowocześniejszych komercyjnych modeli	Prometheus został stworzony w celu zapewnienia rozwiązania ewaluacyjnego uniezależnionego od komercyjnego modelu o zbliżonych możliwościach ewaluacyjnych

Tabela 3.2: Różnice między G-Eval a Prometheus. Na podstawie [48, 42]

3.6 LLM jako sędzia w ocenie generowanych treści

Wykazano, że tradycyjne mechanizmy punktowania takie jak BLEU [54], ROUGE [47] czy METEOR [10] szeroko stosowane do ewaluacji systemów NLG osiągają stosunkowo niską korelację z ocenami ludzkimi [42], w szczególności w problemach o charakterze otwartym oraz zadaniach nie posiadających referencji (tzw. reference-free), które w przypadku tych metryk muszą być zdefiniowane dla każdego wejścia testowego [48, 83]. Wraz z rozwojem LLM-ów, rośnie ich potencjał w całkowitym zastąpieniu ludzkich anotatorów w procesach ewaluacyjnych [30, 38, 83]. Ostatnie badania wprost sugerują bezpośrednie wykorzystanie Wielkich Modeli Językowych jako 'sędziów' dokonujących oceny treści generowanych przez LLM-y [28] na podstawie zdefiniowanych kryteriów. Ten wzorzec w literaturze często nazywany jest 'LLM as a judge' (ang. LLM w roli sędziego)[83]. Nie trudno zauważyć, że zarówno G-Eval (patrz 3.4) jak i Prometheus (patrz 3.5.2) są implementacjami tego wzorca.

3.6.1 Korzyści

Wykorzystanie LLM-a do oceny tekstu generowanego przez inny model, podobnie jak w przypadku większości zautomatyzowanych metod ewaluacji (patrz 3.5), pozwala na kompletne usunięcie ludzkiego aktora z procesu ewaluacji, co skutkuje:

- lepszą skalowalnością i szybszą iteracją - LLM-y mogą relatywnie szybko przetwarzać ogromne ilości danych [83]
- redukcją kosztów - zmniejszając potrzebę intensywnej pracy ludzkiej
- większą elastycznością - fine-tuning może pozwolić na dostosowanie do danego rodzaju ewaluacji poprzez zwiększenie trafności
- usunięciem czynnika ludzkiej subiektywności - ocena nie jest dokonywana przez anotatora - ograniczana jest stronniczość
- możliwością automatycznego generowania uzasadnień - LLM może automatycznie wygenerować wyjaśnienia dla swoich ocen, pozwalając na interpretacje [83]

3.6.2 Ograniczenia

Idea wykorzystania Wielkiego Modelu Językowego jako podmiotu przypisującego punkty generowanym treściom, ma rację bytu jedynie przy założeniu, że obecny stan wytrenowania LLM-ów rzeczywiście pozwala im na przypisywanie wyższego prawdopodobieństwa wysokiej jakości tekstom [48]. Jednakże poprawność oraz niezawodność LLM-ów jako sędziów nie jest jeszcze dokładnie przebadana, co potwierdzają meta-ewaluacje [6] , które pokazują, że niektóre metryki ewaluacyjne oparte o LLM-y mogą wykazywać niższą korelację z ocenami ludzkimi niż metryki oparte o średniej wielkości modele uczenia maszynowego [39]. Poza tym podobnie jak w przypadku ewaluacji dokonywanej przez ludzi, pojawia się problem uprzedzeń - tzw. Self-enhancement bias (ang. autowaloryzacja) [14]. Jest to zjawisko, w którym LLM działający jako sędzia faworyzuje odpowiedzi generowane przez samego siebie (przez ten sam model) [83], co może negatywnie wpływać np. na testy mające na celu porównanie różnych modeli, wśród których jeden z ewaluowanych jest również wykorzystany jako sędzia.

3.7 LLM Benchmarking

Wielkie Modele Językowe można podzielić na dwie podstawowe kategorie:

- komercyjne
- Open-Source

Twórcy modeli komercyjnych w przeciwieństwie do Open-Source, nie udostępniają kodu źródłowego ani dokładnej architektury swojego rozwiązania, zaś dostęp do tych zasobów jest ściśle kontrolowany i ograniczony prawnie. Mimo tych różnic ogólny trend wskazuje na znaczącą rolę obydwu klas modeli w rozwoju dziedziny Gen AI.

Model	Dostawca	Data udostępnienia	Rodzaj	Liczba parametrów
GPT-4o	OpenAI	13.05.2024	Komercyjny	Nieznana
Claude 3.5	Anthropic	20.06.2024	Komercyjny	Nieznana
Grok-1	xAI	4.11.2023	Open-Source	$314 * 10^9$
Gemini 1.5	Google DeepMind	02.02.2024	Komercyjny	Nieznana
Llama 3.1	Meta AI	23.06.2024	Open-Source	$405 * 10^9$

Tabela 3.3: Zestawienie wybranych najpopularniejszych LLM-ów. Stan na wrzesień 2024. Na podstawie: [17]

[6] Meta-ewaluacja - proces oceniania użyteczności metryk ewaluacyjnych wraz z mechanizmami punktowania [20].

Aktualnie istnieje szeroka gama różnych modeli językowych. Kilka najważniejszych przykładów przedstawiono w tabeli 3.3. Dokładna liczba wszystkich tego typu rozwiązań na rynku jest trudna do oszacowania i stale rośnie, jednakże patrząc na takie zestawienia jak:

- Comparison of Models: Quality, Performance & Price Analysis
- Open-Source LLMs

można oszacować, że jest to liczba rzędu tysięcy, w której brane są pod uwagę zarówno małe modele Open-Source, jak i potężne modele komercyjne. Oznacza to, że wybór Wielkiego Modelu Językowego jako narzędzia do rozwiązania konkretnego problemu może stanowić poważne wyzwanie. Właśnie w tym celu wykorzystuje się tzw. benchmarki.

3.7.1 Czym jest benchmark?

W świecie LLM-ów benchmark (ang. benchmark - punkt odniesienia) to specyficzny rodzaj ewaluacji zautomatyzowanej (patrz 3.4.2), pozwalający na analizę porównawczą efektywności różnych modeli, poprzez zdefiniowanie ustandaryzowanej procedury ewaluacji modelu względem pewnych metryk, przy jednoczesnym użyciu ściśle określonego datasetu [5, 73]. Innymi słowy jest to pewien standard określający sposób mierzenia efektywności modeli, pozwalający na ich uczciwe i spójne porównanie [59], a tym samym wyłonienie najlepszego modelu do danego przypadku użycia.

Benchmarki zazwyczaj obejmują testy sprawdzające szeroką gamę zadań oraz umiejętności jak na przykład:

- **Generowanie Języka Naturalnego** - generowanie spójnych oraz trafnych treści na bazie określonych promptów
- **Pytania i odpowiedzi (Q&A)** - interpretowanie znaczenia tekstu i udzielanie dokładnych, trafnych odpowiedzi na pytania
- **Tłumaczenia** - tłumaczenie tekstu na inny język
- **Rozumowanie logiczne i zdrowy rozsądek** - stosowanie logiki (np. umiejętność rozumowania indukcyjnego i dedukcyjnego) w celu rozwiązywania problemów
- **Testy standardowe** - testy takie jak SAT, ACT[7] i inne standardowe testy stosowane w edukacji ludzi
- **Generowanie kodu** - rozumienie i umiejętność generowania kodu źródłowego

[5]

Warto wspomnieć, że benchmarki wykorzystywane są również w fine-tuningu modeli [5, 73, 59], co oznacza, że nie tylko pomagają w wyborze odpowiedniego modelu, ale również stanowią kluczowe narzędzie dla badaczy rozwijających dziedzinę Wielkich Modeli Językowych.

3.7.2 Komponenty i ogólna zasada działania

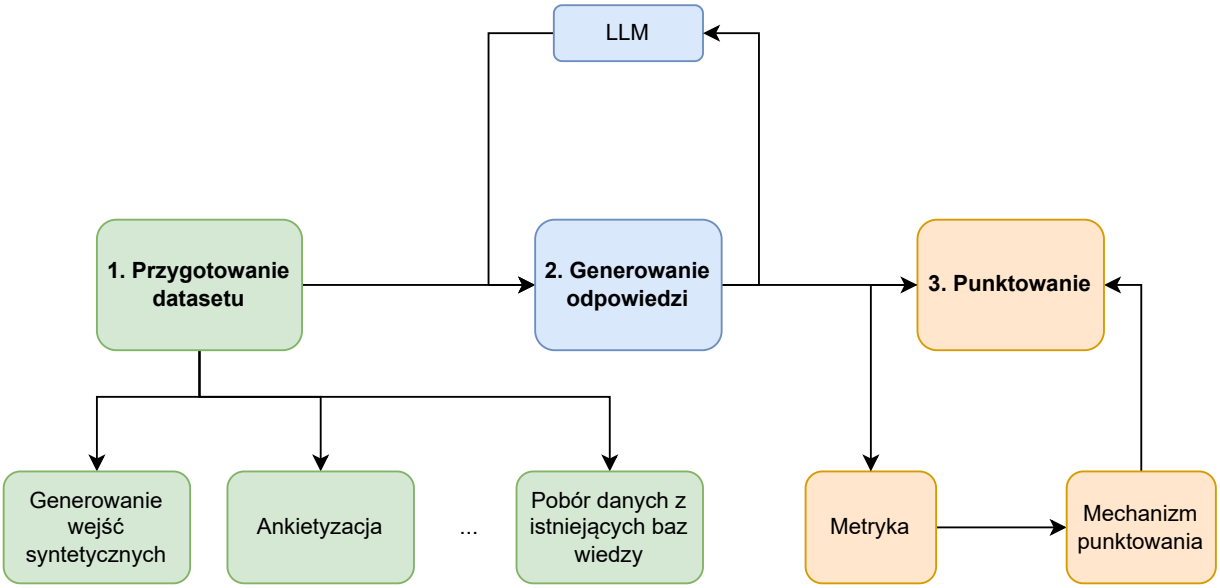
Benchmarki LLM-ów z uwagi na bycie specyficznym przypadkiem ewaluacji wymagają zdefiniowania podobnych elementów. Główne komponenty wchodzące w skład benchmarku to:

- **Dataset** - zbiór pytań, zadań lub problemów pewnego rodzaju lub z określonej dziedziny
- **Metryki** - określające kryteria oceny generowanych treści
- **Mechanizmy punktowania** - pozwalające na wyznaczenie ilości punktów na bazie metryk

[7] SAT (Scholastic Aptitude Test), ACT (American College Test) - egzaminy wstępne na studia wykorzystywane w USA

[59]

Benchmarki mogą znacząco różnić się od siebie na poziomie implementacyjnym, jednakże mimo to wszystkie składają się z faz przedstawionych na rysunku 3.6. Przygotowanie datasetu może być tak proste jak wykorzystanie już istniejącej bazy lub jej podzbioru, lub też tak trudne jak ręczne jego skonstruowanie. Istnieje również możliwość wykorzystania syntetycznych zbiorów danych wygenerowanych np. przy użyciu modelu językowego. Podczas generowania odpowiedzi w zależności od celu oraz charakterystyki danego benchmarku wykorzystywane mogą być różne metody promptingu (patrz 2.4). Na przykład benchmark badający bazową wiedzę modelu może ograniczyć się do wykorzystania promptów typu zero-shot (patrz 2.4.2). To samo tyczy się ostatniej fazy - wykorzystane metryki dobiera się mając na uwadze cel benchmarku.



Rysunek 3.6: Ogólne fazy działania benchmarków LLM. Na podstawie: [59]

3.7.3 Zestawienie wybranych najpopularniejszych benchmarków

Istnieje wiele benchmarków oceniających modele językowe, które różnią się wielkością datasetu, zakresem, metodyką oraz metrykami. W celu usystematyzowania wiedzy w tabeli 3.4 przedstawiono przegląd najpopularniejszych benchmarków, które wybrane zostały ze względu na różnorodność testowanych zdolności w zakresie rozumowania i wiedzy ogólnej przy jednoczesnym wykorzystaniu promptów typu zero-shot (patrz 2.4.2) oraz few-shot (patrz 2.2.2).

Benchmark	Opis
IFEval (Instruction-Following Evaluation) [87]	Skupia się na testowaniu zdolności modelu do przestrzegania jawnych instrukcji, takich jak „uwzględnij słowo kluczowe x” lub „użyj formatu y”. Skupia się on na zgodności modelu z instrukcjami dotyczącymi formatowania, a nie na generowanej treści, co umożliwia wykorzystanie rygorystycznych i precyzyjnych metryk.
BBH (Big Bench Hard) [69]	Oparty jest o podzbiór 23 wymagających zadań z datasetu Big-Bench [68]. Zadania opierają się na obiektywnych metrykach, charakteryzują się wysokim poziomem trudności oraz odpowiednią wielkością próbek, co pozwala na uzyskanie statystycznie istotnych wyników. Obejmują one m.in. arytmetykę wieloetapową, rozumowanie algorytmiczne (np. wyrażenia logiczne, kształty SVG), rozumienie języka (np. wykrywanie sarkazmu, rozróżnianie nazw) oraz wiedzę o świecie. Wyniki BBH dobrze korelują z preferencjami ludzkimi, dostarczając cennych informacji o możliwościach modeli.
MATH [35]	Wykorzystuje zbiór matematycznych problemów konkursowych na poziomie szkoły średniej, zebranych z różnych źródeł i sformatowanych w sposób spójny za pomocą LaTeX (do równań) i Asymptote (do figur). Generowane odpowiedzi muszą spełniać bardzo specyficzne wymagania dotyczące formatu.
GPQA (Graduate-Level Google-Proof Q&A Benchmark) [57]	GPQA opiera się o wyjątkowo wymagający dataset, zawierający pytania opracowane przez ekspertów z tytułami doktorskimi w takich dziedzinach jak biologia, fizyka czy chemia. Pytania te są trudne dla osób spoza danej dziedziny, lecz stosunkowo łatwe dla specjalistów. Zbiór danych przeszedł wiele rund walidacji, aby zapewnić zarówno wysoki poziom trudności, jak i dokładność merytoryczną. Dostęp do GPQA jest ograniczony aby dane nie zostały wykorzystane do trenowania modeli.
MuSR (Multistep Soft Reasoning) [67]	MuSR wykorzystuje dataset zawierający algorytmicznie generowane, złożone problemy, z których każdy liczy około 1000 słów. Problemy te obejmują zagadki kryminalne, pytania dotyczące rozmieszczenia obiektów oraz optymalizacje przydziału zespołów. Rozwiązanie tych problemów wymaga od modeli łączenia umiejętności rozumowania z analizą kontekstu na dużą skalę. Większość modeli osiąga w tym benchmarku wyniki zbliżone do losowych.
MMLU-PRO (Massive Multitask Language Understanding - Professional) [75]	MMLU-Pro to ulepszona wersja datasetu MMLU [34], który był standardem w ocenie wiedzy w formie pytań wielokrotnego wyboru. MMLU-PRO rozwiązuje liczne problemy swojego poprzednika jak występowanie pytań bez poprawnych odpowiedzi, dodatkowo zwiększając liczbę odpowiedzi z 4 do 10, kładąc większy nacisk na rozumowania oraz ekspercką weryfikację pytań.

Tabela 3.4: Wybrane najpopularniejsze benchmarki LLM-ów. Na podstawie [18, 60]

Rozdział 4

HELPS Benchmark

Niniejszy rozdział skupia się na opisaniu celu, genezy oraz procesu powstania benchmarku **HELPS**. Wyniki eksperymentów dla trzech spośród najpopularniejszych modeli komercyjnych znajdują się w kolejnym rozdziale.

4.1 Motywacja i cel

W obliczu lawiny Wielkich Modeli Językowych i chatbotów opartych o techniki NLP pojawiających się niemal co tydzień, często z górnolotnymi deklaracjami na temat ich wydajności, trudno jest oddzielić technologie, które mają szansę wprost wpłynąć na poprawę jakości codziennego życia człowieka, od tych, które dla zwykłych użytkowników nie mają większego znaczenia.

Bardzo często benchmarki wykorzystywane są do manipulacji w celu reklamy i sprzedaży konkretnego modelu, przy czym odbiorcy wyników nie są w stanie ich zweryfikować ze względu na brak kodu źródłowego, który by na to pozwolił. Jednocześnie istniejące benchmarki takie jak MATH [35], GPQA [57] czy IFEval [87] skupiają się ściśle na sprawdzeniu umiejętności modelu w konkretnej dziedzinie czy też sprawdzają jego wiedzę na określonym poziomie. Jednakże żaden z nich nie został stworzony w celu badania skuteczności Wielkich Modeli Językowych w zupełnie losowych pytaniach oraz zadaniach, które rzeczywiście zadawane są modelom językowym przez użytkowników za pośrednictwem takich narzędzi jak ChatGPT (openai), Gemini (Google) czy Claude (Anthropic).

W codziennym życiu potencjalny użytkownik, napotykając niekrytyczny problem, na który nie jest w stanie poświęcić tyle czasu by rozwiązać go samodzielnie może zdecydować się na przekazanie go komuś innemu. Obecnie tym kimś coraz częściej jest sztuczna inteligencja, szczególnie w postaci LLM-ów, lub narzędzi o nich opartych. Można więc dojść do wniosku, że benchmarki oceniające zdolności modeli w rozwiązywaniu problemów akademickich, doktoranckich czy pochodzących z uczelnianych egzaminów kwalifikacyjnych nie wnoszą informacji w kwestii ich użyteczności z perspektywy zwykłego konsumenta.

Z tych powodów w ramach niniejszczej pracy zbudowany został zupełnie nowy benchmark, który został nazwany **HELPS** - jest to akronim od angielskich słów '**H**uman **E**veryday **L**ife **P**ractical **S**upport', czyli 'Praktyczne Wsparcie Człowieka w Życiu Codziennym', co bezpośrednio stanowi cel badawczy tego benchmarku. Dodatkowo jego kod źródłowy pozwalający na odtworzenie wyników wraz z datasetem, wynikami pośrednimi oraz ostatecznymi udostępniony zostaje za pośrednictwem repozytorium (patrz Dodatek A).

4.2 Dataset

Jak przedstawiono na rysunku 3.6 pierwszą i najważniejszą fazą budowy benchmarku jest utworzenie datasetu.

Aby wiarygodnie zbadać jak dobrze Wielkie Modele Językowe mogą wspierać człowieka w codziennych problemach niezbędnym było skompletowanie przykładowych problemów, które ludzie rzeczywiście zlecają modelom. W tym celu wykorzystana została ankieta, w której udział wzięły osoby reprezentujące różne grupy zawodowe. Dokładne wyniki ankiety przedstawiono w dodatku B.

Każdy z respondentów miał za zadanie podać do pięciu problemów, z którymi rzeczywiście spotyka się na co dzień, których rozwiązanie zleca, zlecał lub zleciłby Wielkiemu Modelowi Językowemu. Ankieta wymagała od respondentów odpowiedzi w języku angielskim w cel uniknięcia niuansów związanych z translacją dokonywaną przez modele. Następnie zaproponowane przez respondentów problemy zostały poddane manualnej weryfikacji, w wyniku której część z nich została odrzucona[1]. Proces ten pozwolił na skompletowanie 100 pytań oraz zadań pożądanej klasy - przykład przedstawiono na rysunku 4.1.

```
{
    "task_id": "helps/39",
    "input": "My stove is dead. make a recipe with instant noodles, cheese,
              ham, bacon, spices without using a stove."
}
```

Rysunek 4.1: Przykład problemu pochodzącego z datasetu **HELPS**

Uzyskany zbiór można określić jako 'reference-free' (ang. bez referencji) co oznacza, że dataset nie zawiera w sobie oczekiwanych odpowiedzi. Jest to działanie celowe, gdyż zdefiniowanie oczekiwanej odpowiedzi wprowadza czynnik subiektywności, co może negatywnie wpłynąć na ocenę generowanych treści w kontekście wykorzystanych metryk (patrz 4.3). Brak referencji jest kolejną cechą znacznie wyróżniającą benchmark **HELPS** na tle innych tego typu rozwiązań.

Utworzenie datasetu od podstaw miało również na celu uniknięcie zjawiska zanieczyszczenia danymi (ang. Data Contamination), czyli sytuacji, w której dane użyte do ewaluacji modelu stanowiły pewien podzbiór danych treningowych tego modelu. Prowadzi to do nieuczciwie wysokich wyników w benchmarku opartym o ten dataset [34], gdyż model poznaje odpowiedzi na pytania już w fazie treningu.

4.3 Metryki i mechanizm punktowania

Tak jak wspomniano w poprzednim podrozdziale analizując elementy datasetu **HELPS** okazało się, że znaczna część problemów zgłoszonych przez respondentów ma charakter otwarty, innymi słowy oczekiwana odpowiedź nie może być jednoznacznie określona. Z tego powodu wszystkie metryki o nie oparte (patrz 3.4.2) musiały zostać wykluczone. Dodatkowo ze względu na potrzebę automatyzacji oraz umożliwienia odtworzenia wyników zdecydowano się na wykorzystanie wzorca 'LLM as a judge' (patrz 3.6).

Benchmarki bazujące na referencjach skupiają się na głównej metryce jaką jest poprawność (ang. correctness), czyli sprawdzają czy wygenerowana odpowiedź jest poprawna w zestawieniu z odpowiedzią oczekiwaną. Tutaj benchmark **HELPS** również się różni, gdyż jego celem nie jest ocena poprawności generowanych treści, lecz zmierzenie jak bardzo ta treść jest pomocna dla człowieka w kontekście przedstawionego przez niego problemu. W tym celu wykorzystywane są trzy metryki przedstawione w tabeli 4.1.

Metryka	Opis	Punktacja	Waga
Pomocność (ang. helpfulness)	Określa jak bardzo wygenerowana treść jest pomocna względem przedstawionego problemu.	[0, 1]	0.5
Adekwatność (ang. relevancy)	Określa jak bardzo wygenerowana treść jest adekwatna względem przedstawionego problemu.	[0, 1]	0.3
Zwięzłość (ang. brevity)	Określa jak bardzo zwięzła i konkretna jest wygenerowana odpowiedź.	[0, 1]	0.2

Tabela 4.1: Metryki wykorzystane w benchmarku **HELPS**

Wszystkie metryki **HELPS** zaimplementowane są przy wykorzystaniu frameworka G-Eval (patrz 3.4)

[1]Problem był odrzucany gdy nie był codziennym problemem ludzkim, był niecenzuralny lub nie miał sensu

(wariant z normalizacją wyniku) w celu uzyskania ocen możliwie zbliżonych do ocen ludzkich. Ostateczny wynik obliczany jest według następujących kroków:

Krok 1

Obliczenie wyniku ważonego dla każdego elementu datasetu **HELPS**:

$$E = w_P \cdot P + w_A \cdot A + w_Z \cdot Z$$

Gdzie:

- E jest wynikiem ważonym dla elementu
- P, A, Z są wynikami odpowiednio dla pomocności, adekwatności i zwięzłości (patrz tabela 4.1)
- w_P, w_A, w_Z są wagami (patrz tabela 4.1)

Krok 2

Obliczenie wyniku całkowitego jako średniej wyników ważonych:

$$C = \frac{1}{N} \sum_{i=1}^{N} \mathrm{E}_i$$

Gdzie:

- C jest wynikiem całkowitym
- E_i jest wynikiem ważonym dla elementu i (patrz **Krok 1** - 4.3)
- N jest liczbą elementów w datasecie **HELPS** (patrz podrozdział 4.2)

4.4 Schemat działania

Uproszczony schemat działania benchmarku **HELPS** przedstawiono na rysunku 4.2.

4.5 Implementacja

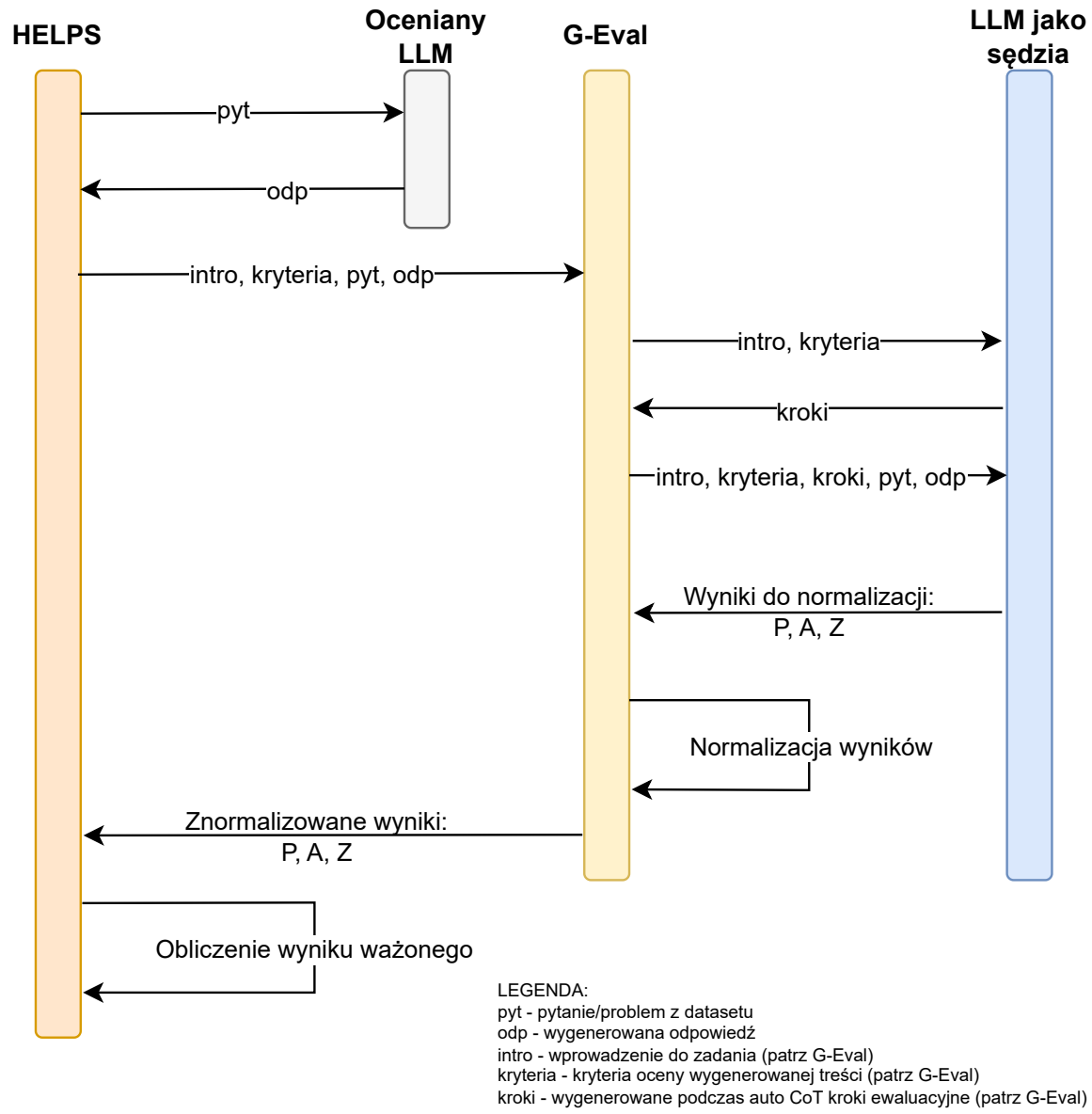
Aby uniknąć sytuacji, w której benchmark **HELPS** wykorzystywany jest do fałszywej reklamy nowych modeli, czy też do przedstawiania zmanipulowanych wyników w celu wyróżnienia pewnego modelu względem innych, jego całkowity kod źródłowy zostaje udostępniony za pośrednictwem repozytorium (patrz dodatek A), co w prosty sposób pozwala na weryfikację i odtworzenie wyników. W tym podrozdziale opisane zostały wykorzystane technologie, organizacja oraz działanie kodu źródłowego benchmarku.

4.5.1 Technologie

Z racji szybkiego rozwoju GenAI oraz jej rosnącego zastosowania rośnie również ilość narzędzi ułatwiających integrację, wykorzystanie czy nawet ewaluację Wielkich Modeli Językowych. W tym podrozdziale opisano dwie najbardziej kluczowe biblioteki wykorzystane w kodzie źródłowym benchmarku **HELPS**.

LangChain

LangChain to Open-Sourceowy framework służący do budowania aplikacji opartych o LLM-y [1, 4]. Dostarcza on zbiór narzędzi, które ułatwiają budowę złożonych aplikacji takich jak chatboty, rozwiązania oparte o agentów oraz narzędzia wymagające integracji z zewnętrznymi bazami wiedzy (np. RAG). Jednakże przede wszystkim LangChain zapewnia łatwą w użyciu warstwę abstrakcji, dzięki której wysyłanie zapytań do dowolnych spośród najpopularniejszych Wielkich Modeli Językowych możliwe jest za pośrednictwem jednolitego a zarazem elastycznego interfejsu programistycznego.



Rysunek 4.2: Diagram przebiegu benchmarku **HELPS**

DeepEval

DeepEval to również framework Open-Sourceowy, jednakże w przeciwieństwie do LangChain (patrz 4.5.1) nie skupia się na samej integracji z Wielkimi Modelami Językowymi, a raczej udostępnia narzędzia do ich zautomatyzowanej ewaluacji (patrz 3.4.2). Rozwiązanie to nie tylko pozwala na ewaluowanie treści w ramach testów jednostkowych, czy generowanie syntetycznych datasetów, ale przede wszystkim udostępnia elastyczną oraz łatwą w użyciu implementację metryk wykorzystujących LLM-y, w tym G-Eval (patrz 3.4).

4.5.2 Kod źródłowy

Aby pozwolić na proste odtworzenie wyników benchmarku **HELPS** jego implementacja (patrz dodatek A) została podzielona na kilka skryptów napisanych w jednym z najbardziej popularnych wysokopoziomowych języków programowania jakim jest python 3. Każdy ze skryptów odpowiedzialny jest za dany

etap działania benchmarku.

Generowanie treści

Tak jak przedstawiono na rysunku 3.6 każdy z benchmarków posiada fazę generowania treści. W wypadku benchmarku **HELPS** realizowana jest ona przy użyciu skryptu `generator.py` (patrz folder `generation`). Jego działanie sprowadza się do załadowania pliku w formacie JSON będącego datasetem benchmarku (patrz folder `dataset` oraz podrozdział 4.2). Następnie dla każdego wejścia z datasetu generowana jest odpowiedź za pomocą predefiniowanego system promptu (patrz plik `system_prompt.txt` lub rysunek 5.1) przez wybrany model (określony za pomocą parametru wejściowego skryptu). Otrzymane wyniki oraz ewentualne logi zapisywane są w plikach w formacie JSON (patrz foldery `results` oraz `logs`). W celu uproszczenia komunikacji z modelami wykorzystano bibliotekę LangChain (patrz 4.5.1).

Ewaluacja odpowiedzi

Ewaluacja wygenerowanych treści odbywa się za pośrednictwem skryptu `evaluator.py` (patrz folder `evaluation`). Skrypt ten pobiera odpowiedzi wygenerowane przez dany model z pliku JSON otrzymanego w fazie generowania a następnie wykorzystuje implementację frameworka ewaluacyjnego G-Eval (patrz podrozdział 3.4) pochodzącą z biblioteki DeepEval (patrz 4.5.1) w celu wyznaczenia wartości metryk opisanych w podrozdziale 4.3. Otrzymane wyniki wraz z uzasadnieniami wygenerowanymi przez model pełniący rolę sędziego (patrz tabela 5.2) ponownie zapisywane są w formacie JSON (patrz folder `results`) - przykład wyniku ewaluacji przedstawiono w dodatku C.

Na rysunku 4.3 znajduje się fragment kodu stanowiący główną pętlę ewaluacyjną benchmarku **HELPS**, zaś na rysunku 4.4 przedstawiono jej wywołanie wraz z definicją odpowiednich metryk (patrz 4.3). Dzięki abstrakcji z biblioteki DeepEval (patrz 4.5.1) kod benchmarku odpowiedzialny jest jedynie za zdefiniowanie kryteriów ewaluacyjnych (patrz 3.4) oraz określenie modelu, który będzie pełnił rolę sędziego (patrz 3.6).

Generowanie raportu

W celu przeprowadzenia analizy wyników benchmarku **HELPS** zaimplementowane zostały dodatkowe skrypty (patrz folder `report`) pozwalające na pobranie wyników ewaluacji z plików JSON oraz wygenerowanie na ich podstawie wykresów przedstawiających ogólny wynik benchmarku (patrz podrozdział 4.3 oraz rysunek 5.2) oraz średnią ilość punktów dla każdej z metryk, dla każdego z modeli (patrz rysunek 5.3).

```python
# python 3.12
import tqdm
from deepeval.metrics import GEval
from deepeval.test_case import LLMTestCase

def run_evaluation(generation_results_per_model, metrics: list[GEval], limit=None):
    for entry in tqdm.tqdm(generation_results_per_model):
        model_name = entry["model_name"]
        file_name = entry["file_name"]

        generation_results = load_generation_results(file_name)[:limit]

        evaluation_results = []
        log_file_name = get_log_file_name(model_name)
        results_file_name = get_results_file_name(model_name)

        for generation_result in tqdm.tqdm(generation_results):
            evaluation_result = {**generation_result, "evaluation": {"results": []}}
            test_input = generation_result["input"]
            test_actual_output = generation_result["output"]["output"]["content"][
                "content"
            ]
            test_case = LLMTestCase(input=test_input, actual_output=test_actual_output)
            for metric in tqdm.tqdm(metrics):
                metric.measure(test_case)
                evaluation_result["evaluation"]["results"].append(
                    {
                        "metric_name": metric.name,
                        "score": metric.score,
                        "reason": metric.reason,
                        "criteria": metric.criteria,
                        "evaluation_steps": metric.evaluation_steps,
                        "evaluation_model": metric.evaluation_model,
                        "verbose_logs": metric.verbose_logs,
                    }
                )
            evaluation_results.append(evaluation_result)
            dump_to_file(evaluation_result, log_file_name)
        dump_to_file(evaluation_results, results_file_name)
```

Rysunek 4.3: Główna pętla ewaluacyjna benchmarku **HELPS**

```python
# python 3.12
from deepeval.metrics import GEval
from deepeval.test_case import LLMTestCaseParams
run_evaluation(
    [
        {
            "model_name": "gpt-4",
            "file_name": "2024-12-13_22-46-23_results_gpt-4.json",
        },
        {
            "model_name": "gemini-1-5-pro",
            "file_name": "2024-12-13_23-34-06_results_gemini-1-5-pro.json",
        },
        {
            "model_name": "claude-3-5-sonnet",
            "file_name": "2024-12-14_00-00-23_results_claude-3-5-sonnet.json",
        },
    ],
    [
        GEval(
            name="brevity",
            criteria="Determine the brevity of the actual output in the context of input",
            evaluation_params=[
                LLMTestCaseParams.INPUT,
                LLMTestCaseParams.ACTUAL_OUTPUT,
            ],
            model=azure_openai_gpt4o_model,
            verbose_mode=True,
        ),
        GEval(
            name="relevancy",
            criteria="Determine the relevancy of the actual output in the context of input",
            evaluation_params=[
                LLMTestCaseParams.INPUT,
                LLMTestCaseParams.ACTUAL_OUTPUT,
            ],
            model=azure_openai_gpt4o_model,
            verbose_mode=True,
        ),
        GEval(
            name="helpfulness",
            criteria="Determine the helpfulness of the actual output in the context of input",
            evaluation_params=[
                LLMTestCaseParams.INPUT,
                LLMTestCaseParams.ACTUAL_OUTPUT,
            ],
            model=azure_openai_gpt4o_model,
            verbose_mode=True,
        ),
    ],
)
```

Rysunek 4.4: Definicja metryk benchmarku **HELPS** na bazie implementacji G-Eval pochodzącej z biblioteki DeepEval (patrz 4.5.1)

Rozdział 5

Podsumowanie

W tym rozdziale opisane zostały parametry oraz wyniki benchmarku **HELPS** dla wybranych modeli językowych, na podstawie których wyciągnięte zostały wnioski. Ponadto przeprowadzona zostaje dyskusja na temat wykorzystania LLM-ów jako sędziów oraz omówione zostają potencjalne plany rozwoju.

5.1 Parametry

W tabeli 5.1 przedstawiono zestawienie modeli, które poddane zostały badaniu za pośrednictwem benchmarku **HELPS**[1]. Zaś tabela 5.2 przedstawia szczegółowe dane modelu, który wykorzystany został jako sędzia[2].

Nazwa	Dostawca	Model	Wersja
GPT-4	openai	gpt-4	0613
Gemini 1.5 Pro	google	gemini-1.5-pro	001
Claude 3.5 Sonnet	Anthropic	anthropic.claude-3-5-sonnet	20240620-v1:0

Tabela 5.1: Modele zbadane za pośrednictwem benchmarku **HELPS**

Nazwa	Dostawca	Model	Wersja
GPT-4o	openai	gpt-4o	2024-08-06

Tabela 5.2: Model pełniący rolę sędziego w benchmarku **HELPS**

W fazie generowania odpowiedzi (patrz rysunek 3.6) niezależnie od ewaluowanego modelu wykorzystane zostały te same wartości ustawień oraz ten sam system prompt (patrz 2.1). Wartości ustawień modeli wraz z uzasadnieniem przedstawiono w tabeli 5.3 zaś treść system promptu przedstawiono na rysunku 5.1.

5.2 Wyniki

W celu wygenerowania odpowiedzi każdy problem z datasetu **HELPS** (patrz podrozdział 4.2) przekazywany był do modelu bez dodatkowego formatowania czy wzbogacania treści. Taką strategię promptingu

[1] modele zostały wybrane w oparciu o ich dostępność za pośrednictwem interfejsów łatwych w obsłudze przez osoby nietechniczne

[2] warto tutaj zaznaczyć, że w celu uniknięcia problemu autowaloryzacji (patrz 3.6.2) LLM pełniący rolę sędziego nie jest testowany przez benchmark **HELPS**

Ustawienie	Wartość	Uzasadnienie
Temperature	0.7	Zapewnia równowagę między kreatywnością a deterministycznością. Umiarkowana wartość sprzyja różnorodnym odpowiedziom przy zachowaniu spójności.
Top-p (nucleus sampling)	0.95	Model uwzględnia szeroki zakres najbardziej prawdopodobnych tokenów, ograniczając jednocześnie nadmierne cięcia różnorodności.
Frequency Penalty	0	Nakładanie kary za powtarzanie słów nie ma sensu ze względu na metryki wykorzystywane w **HELPS**.
Presence Penalty	0	Nakładanie kary za występowanie słów nie ma sensu ze względu na metryki wykorzystywane w **HELPS**.

Tabela 5.3: Wartości najważniejszych ustawień modeli testowanych przez **HELPS**

SYSTEM PROMPT:

> You are a friendly and knowledgeable assistant, skilled in providing practical advice and support for everyday life. Your role is to help users solve real-life problems, make decisions, and answer questions related to daily activities in a clear, concise, and actionable manner. Ensure your responses are helpful, fact-based, and empathetic. Always clarify uncertainties and avoid guessing if you're unsure about something. Where appropriate, provide step-by-step instructions or examples to support your advice.

Rysunek 5.1: System prompt wykorzystany w fazie generowania benchmarku **HELPS**

można sklasyfikować jako zero-shot (patrz podrozdział 2.4.2). Stanowi ona dla modeli większe wyzwanie [75], jednakże odwzorowuje sposób w jaki ludzie rzeczywiście wchodzą w interakcję z modelami[3] poprzez interfejsy konwersacyjne, co prowadzi do bardziej wiarygodnych wyników.

Ostateczne wyniki benchmarku **HELPS** dla wybranych Wielkich Modeli Językowych przedstawiono na rysunku 5.2.

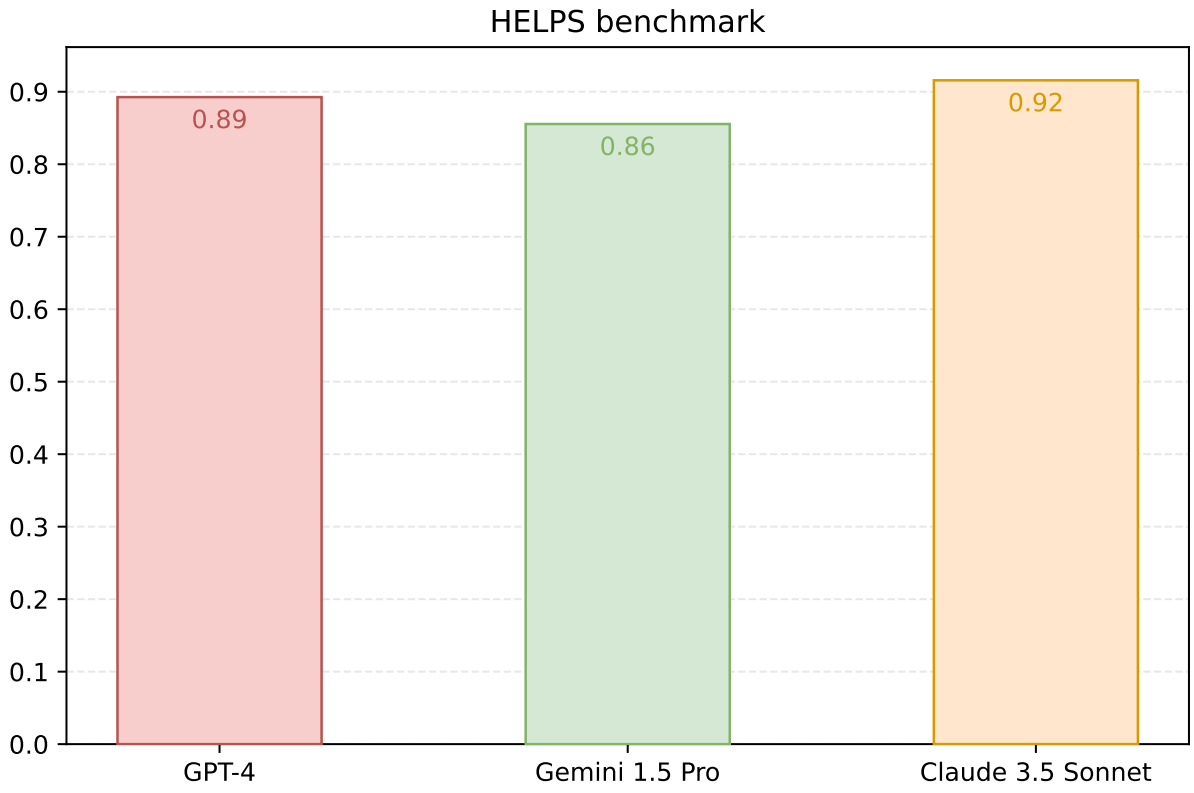
5.3 Dyskusja

Mimo jawnych korzyści jakie w procesie ewaluacji daje całkowite zastąpienie ludzkiego anotatora sędzią będącym Wielkim Modelem Językowym (patrz 3.6.1) nadal pozostaje problem jakości generowanych ocen oraz ich korelacji z ocenami ludzkimi (patrz podrozdział 3.6.2). Benchmark **HELPS** w celu zwiększenia tej korelacji wykorzystuje framework G-Eval (patrz 3.4), który z kolei wykorzystuje szereg rozwiązań mających bezpośredni wpływ na wspomnianą korelację: autoCot (patrz 2.4.3), paradygmat wypełniania formularza (patrz 2.2.2) czy normalizacja liczby punktów (patrz 3.4). Jednakże praca opisująca framework G-Eval wykazuje wyższość tej metody tylko dla problemów pewnych klas (generowanie streszczeń, konwersacja etc.) [48]. Oznacza to, że użycie tego rozwiązania w implementacji benchmarku **HELPS** może mieć za równo pozytywny jak i negatywny wpływ na generowane oceny.

Patrząc na losową próbkę z datasetu **HELPS** oraz przydzielone jej oceny (patrz Dodatek C) można stwierdzić, że są one uzasadnione, jednakże takie stwierdzenie jest subiektywne i powinno być poddane ocenie szerszej grupie ludzi aby zminimalizować czynnik stronniczości.

Badając wyniki benchmarku osobno dla każdej z metryk (patrz rysunek 5.3) można stwierdzić, że modele mają problem ze zwięzłym formułowaniem odpowiedzi, co potencjalnie może wpływać na zawyżanie ocen pozostałych metryk, ze względu na zwiększoną ilość kontekstu jaką musi przetworzyć sędzia.

[3] wynika to bezpośrednio z przykładów problemów podanych przez respondentów ankiety (patrz Dodatek B)



Rysunek 5.2: Wyniki benchmarku **HELPS** dla wybranych modeli

Podobnie bezpośredni wpływ na oceny mogą mieć preferencje modelu pełniącego rolę sędziego - np. model może faworyzować lub penalizować modele z tej samej rodziny (patrz 3.6.2) [19].

Potencjalne rozwiązania tych problemów opisano w podrozdziale 5.5.

5.4 Wnioski

Zakładając, że wykorzystanie wzorca 'LLM as a judge' (patrz 3.6) w połączeniu z frameworkiem G-Eval (patrz 3.4) skutkuje ocenami treści zbliżonymi do ocen jakie przydzielili by ludzie, na podstawie wyników benchmarku **HELPS** (patrz rysunek 5.2) można dojść do wniosku, że najnowocześniejsze Wielkie Modele Językowe są zdecydowanie zdolne do pełnienia funkcji narzędzia wspierającego człowieka na co dzień, ze względu na wysoką pomocność oraz adekwatność odpowiedzi[4]. Jednakże przyglądając się wykorzystanym metrykom z osobna (patrz rysunek 5.3) widać, że modele domyślnie mają tendencję do generowania rozwlekłych odpowiedzi.

Model Claude 3.5. Sonnet okazał się najbardziej efektywny, jednak warto zauważyć, że różnice w wydajności między ocenianymi modelami są niewielkie.

Na podstawie ankiety oraz utworzonego na jej podstawie datasetu **HELPS** nasuwa się wniosek, iż zapytania formułowane przez ludzi w stronę Wielkich Modeli Językowych mają w większości przypadków charakter zero-shot (patrz 2.4.2).

5.5 Możliwości dalszego rozwoju

Poniżej przedstawione zostają potencjalne plany rozwoju oraz ewaluacji benchmarku:

[4] wysoka pomocność i adekwatność nie gwarantują poprawności odpowiedzi

Rysunek 5.3: Średnia ilość punktów przydzielonych w benchmarku **HELPS** dla każdej z metryk, dla wszystkich modeli

- Przeprowadzenie meta-ewaluacji lub budowa meta-benchmarku pozwalającego na badanie korelacji ocen generowanych przez benchmark **HELPS** z faktycznymi ocenami ludzkimi - można w tym celu wykorzystać ScaleEval [20]
- Zwiększenie ilości wykorzystanych metryk np. zaufanie, wiarygodność, toksyczność etc.
- Poszerzenie ilości oraz podwyższenie jakości problemów w datasecie **HELPS** - ponowne przeprowadzenie ankiety na większej oraz bardziej zróżnicowanej grupie ludzi
- Dokonanie etykietowania problemów w datasecie dzieląc je na pewne podgrupy np. pod kątem tematu lub charakteru problemu - pozwoli to na bardziej dogłębną analizę wyników oraz pozwoli podzielić problemy na klastry co korzystnie wpłynie na kroki ewaluacyjne generowane w ramach G-Eval (patrz 3.4 oraz 2.4.3)
- Wykorzystanie benchmarku **HELPS** w celu zbadania szerszej grupy modeli językowych zarówno komercyjnych jak i nie - w szczególności uwzględniając fine-tuningowane SLM-y (ang. Small Language Model)
- Dokonanie eksperymentów na zmodyfikowanej implementacji benchmarku np. generowanie pojedynczych odpowiedzi zastąpić generowaniem kilku alternatywnych odpowiedzi dla każdego elementu datasetu w celu uśrednienia wyników ewaluacyjnych
- Dokonanie eksperymentów używając innych wartości ustawień modeli, innego system promptu oraz wykorzystując inny model w roli sędziego - dzięki temu będzie można stwierdzić, czy GPT-4o jako sędzia faworyzował model ze swojej rodziny

Bibliografia

[1] LangChain Documentation (Python). https://python.langchain.com/docs/.

[2] Metric: bleurt. https://huggingface.co/spaces/evaluate-metric/bleurt.

[3] Słownik Języka Polskiego - PWN. https://sjp.pwn.pl/slowniki/semantyka.html.

[4] What is LangChain? https://aws.amazon.com/what-is/langchain/.

[5] How does LLM benchmarking work? An introduction to evaluating models. https://symflower.com/en/company/blog/2024/llm-evaluation-101, 2024.

[6] X. Amatriain. Prompt design and engineering: Introduction and advanced methods, 2024.

[7] S. An, Z. Ma, Z. Lin, N. Zheng, J.-G. Lou. Make your llm fully utilize the context, 2024.

[8] T. Ayodele. *Introduction to Machine Learning*. 02 2010.

[9] Y. Bai, A. Jones, K. Ndousse, *et al.* Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022.

[10] S. Banerjee, A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. J. Goldstein, A. Lavie, C.-Y. Lin, C. Voss, redaktorzy, *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, strony 65–72, Ann Arbor, Michigan, Czerw. 2005. Association for Computational Linguistics.

[11] P. Bhavsar, B. Gheorghe. LLM-as-a-Judge vs Human Evaluation. https://www.galileo.ai/blog/llm-as-a-judge-vs-human-evaluation, 2024.

[12] R. Bommasani, D. A. Hudson, E. Adeli, *et al.* On the opportunities and risks of foundation models, 2022.

[13] S. R. Bowman, G. Angeli, C. Potts, C. D. Manning. A large annotated corpus for learning natural language inference, 2015.

[14] J. D. Brown. Evaluations of self and others: Self-enhancement biases in social judgments. *Social cognition*, 4(4):353–376, 1986.

[15] T. B. Brown, B. Mann, N. Ryder, *et al.* Language models are few-shot learners, 2020.

[16] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, Y. Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.

[17] A. Cardillo. List of the Best 21 Large Language Models (LLMs) (September 2024). https://explodingtopics.com/blog/list-of-llms, 2024.

[18] Y. Chang, X. Wang, J. Wang, *et al.* A survey on evaluation of large language models, 2023.

[19] G. H. Chen, S. Chen, Z. Liu, F. Jiang, B. Wang. Humans or llms as the judge? a study on judgement biases, 2024.

[20] S. Chern, E. Chern, G. Neubig, P. Liu. Can large language models be trusted for evaluation? scalable meta-evaluation of llms as evaluators via agent debate, 2024.

[21] Y. K. Chia, P. Hong, L. Bing, S. Poria. Instructeval: Towards holistic evaluation of instruction-tuned large language models, 2023.

[22] A. Chockalingam. A Beginner's Guide to Large Language Models Part 1. https://www.amax.com/content/files/2024/03/llm-ebook-part1-1.pdf, 2023.

[23] DAIR.AI. Prompt Engineering Guide. https://www.promptingguide.ai/, 2024.

[24] S. Diao, R. Pan, H. Dong, K. S. Shum, J. Zhang, W. Xiong, T. Zhang. Lmflow: An extensible toolkit for finetuning and inference of large foundation models, 2024.

[25] K. K. S. S. Diksha Khurana, Aditya Koli. Natural language processing: state of the art,current trends and challenges. https://rdcu.be/dZdv3, 2022.

[26] C. Dong, Y. Li, H. Gong, M. Chen, J. Li, Y. Shen, M. Yang. A survey of natural language generation. *ACM Computing Surveys*, 55(8):1–38, Gru. 2022.

[27] S. Feuerriegel, J. Hartmann, C. Janiesch, P. Zschech. Generative ai. *Business amp; Information Systems Engineering*, 66(1):111–126, Wrze. 2023.

[28] J. Fu, S.-K. Ng, Z. Jiang, P. Liu. Gptscore: Evaluate as you desire, 2023.

[29] S. Gehrmann, E. Clark, T. Sellam. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text, 2022.

[30] F. Gilardi, M. Alizadeh, M. Kubli. Chatgpt outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30), Lip. 2023.

[31] B. Goertzel. Artificial general intelligence: Concept, state of the art, and future prospects. *Journal of Artificial General Intelligence*, 0, 01 2014.

[32] Z. Guo, R. Jin, C. Liu, Y. Huang, D. Shi, Supryadi, L. Yu, Y. Liu, J. Li, B. Xiong, D. Xiong. Evaluating large language models: A comprehensive survey, 2023.

[33] R. Haldar, D. Mukhopadhyay. Levenshtein distance technique in dictionary lookup methods: An improved approach, 2011.

[34] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, J. Steinhardt. Measuring massive multitask language understanding, 2021.

[35] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, J. Steinhardt. Measuring mathematical problem solving with the math dataset, 2021.

[36] A. Holtzman, P. West, L. Zettlemoyer. Generative models as a complex systems science: How can we make sense of large language model behavior?, 2023.

[37] T. Hosking, P. Blunsom, M. Bartolo. Human feedback is not gold standard, 2024.

[38] F. Huang, H. Kwak, J. An. Is chatgpt better than human annotators? potential and limitations of chatgpt in explaining implicit hate speech. *Companion Proceedings of the ACM Web Conference 2023*, WWW '23, strona 294–297. ACM, Kwi. 2023.

[39] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, T. Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, 2023.

[40] J. Ip. LLM Evaluation Metrics: The Ultimate LLM Evaluation Guide. https://www.confident-ai.com/blog/llm-evaluation-metrics-everything-you-need-for-llm-evaluation, 2024.

[41] M. R. J, K. VM, H. Warrier, Y. Gupta. Fine tuning llm for enterprise: Practical guidelines and recommendations, 2024.

[42] S. Kim, J. Shin, Y. Cho, J. Jang, S. Longpre, H. Lee, S. Yun, S. Shin, S. Kim, J. Thorne, M. Seo. Prometheus: Inducing fine-grained evaluation capability in language models, 2024.

[43] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa. Large language models are zero-shot reasoners, 2023.

[44] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, D. Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[45] Y. Li, S. Wang, H. Ding, H. Chen. Large language models in finance: A survey, 2024.

[46] P. Liang, R. Bommasani, T. Lee, *et al.* Holistic evaluation of language models, 2023.

[47] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, strony 74–81, Barcelona, Spain, Lip. 2004. Association for Computational Linguistics.

[48] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, C. Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment, 2023.

[49] A. Lu, H. Zhang, Y. Zhang, X. Wang, D. Yang. Bounding the capabilities of large language models in open text generation with prompt constraints, 2023.

[50] C. McClain. Americans' use of ChatGPT is ticking up, but few trust its election information. https://www.pewresearch.org/short-reads/2024/03/26/americans-use-of-chatgpt-is-ticking-up-but-few-trust-its-election-information/, 2024.

[51] B. Meskó. Prompt engineering as an important emerging skill for medical professionals: Tutorial. *J Med Internet Res*, 25:e50638, Oct 2023.

[52] B. Muller. BERT 101: State Of The Art NLP Model Explained. https://huggingface.co/blog/bert-101, 2022.

[53] L. Ouyang, J. Wu, X. Jiang, *et al.* Training language models to follow instructions with human feedback, 2022.

[54] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. P. Isabelle, E. Charniak, D. Lin, redaktorzy, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, strony 311–318, Philadelphia, Pennsylvania, USA, Lip. 2002. Association for Computational Linguistics.

[55] D. Peringanji. The impact of large language models (llms) on everyday applications: Opportunities, challenges, and considerations, 2024.

[56] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever. Language models are unsupervised multitask learners. 2019.

[57] D. Rein, B. L. Hou, A. C. Stickland, J. Petty, R. Y. Pang, J. Dirani, J. Michael, S. R. Bowman. Gpqa: A graduate-level google-proof qa benchmark, 2023.

[58] L. Reynolds, K. McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA, 2021. Association for Computing Machinery.

[59] C. S. Rina Caballar. What are LLM benchmarks? https://www.ibm.com/think/topics/llm-benchmarks, 2024.

[60] C. S. Rina Caballar. What are LLM benchmarks? https://www.ibm.com/think/topics/llm-benchmarks, 2024.

[61] S. Schulhoff, M. Ilie, N. Balepur, *et al.* The prompt report: A systematic survey of prompting techniques, 2024.

[62] J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, UK, 1969.

[63] T. Sellam. BLEURT: a Transfer Learning-Based Metric for Natural Language Generation. https://github.com/google-research/bleurt/blob/master/README.md, 2022.

[64] T. Sellam, D. Das, A. P. Parikh. Bleurt: Learning robust metrics for text generation, 2020.

[65] D. H. Shane Peckham, Jeff Day. Getting started with LLM fine-tuning. https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/generative-ai/working-with-llms/fine-tuning, 2024.

[66] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, S. Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. B. Webber, T. Cohn, Y. He, Y. Liu, redaktorzy, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, strony 4222–4235, Online, List. 2020. Association for Computational Linguistics.

[67] Z. Sprague, X. Ye, K. Bostrom, S. Chaudhuri, G. Durrett. Musr: Testing the limits of chain-of-thought with multistep soft reasoning, 2024.

[68] A. Srivastava, A. Rastogi, A. Rao, A. A. et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2023.

[69] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le, E. H. Chi, D. Zhou, J. Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022.

[70] B. E. Team. ChatGPT / OpenAI Statistics: How Many People Use ChatGPT? https://backlinko.com/chatgpt-stats, 2024.

[71] R. Tutunov, A. Grosnit, J. Ziomek, J. Wang, H. Bou-Ammar. Why can large language models generate correct chain-of-thoughts?, 2024.

[72] S. Uspenskyi. Large Language Model Statistics And Numbers. https://springsapps.com/knowledge/large-language-model-statistics-and-numbers-2024, 2024.

[73] K. Vongthongsri. LLM Benchmarks Explained: Everything on MMLU, HellaSwag, BBH, and Beyond. https://www.confident-ai.com/blog/llm-benchmarks-mmlu-hellaswag-and-beyond, 2024.

[74] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, H. Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2023.

[75] Y. Wang, X. Ma, G. Zhang, Y. Ni, A. Chandra, S. Guo, W. Ren, A. Arulraj, X. He, Z. Jiang, T. Li, M. Ku, K. Wang, A. Zhuang, R. Fan, X. Yue, W. Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024.

[76] Z. Wang, S. Mao, W. Wu, T. Ge, F. Wei, H. Ji. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration, 2024.

[77] Z. M. Wang, Z. Peng, H. Que, J. Liu, W. Zhou, Y. Wu, H. Guo, R. Gan, Z. Ni, J. Yang, M. Zhang, Z. Zhang, W. Ouyang, K. Xu, S. W. Huang, J. Fu, J. Peng. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models, 2024.

[78] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, W. Fedus. Emergent abilities of large language models, 2022.

[79] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[80] Z. Zhang, Y. Yao, A. Zhang, X. Tang, X. Ma, Z. He, Y. Wang, M. Gerstein, R. Wang, G. Liu, H. Zhao. Igniting language intelligence: The hitchhiker's guide from chain-of-thought reasoning to language agents, 2023.

[81] Z. Zhang, A. Zhang, M. Li, A. Smola. Automatic chain of thought prompting in large language models, 2022.

[82] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, J.-R. Wen. A survey of large language models, 2024.

[83] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, I. Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[84] Y. Zheng, W. Gan, Z. Chen, Z. Qi, Q. Liang, P. S. Yu. Large language models for medicine: A survey, 2024.

[85] M. Zhong, Y. Liu, D. Yin, Y. Mao, Y. Jiao, P. Liu, C. Zhu, H. Ji, J. Han. Towards a unified multi-dimensional evaluator for text generation, 2022.

[86] W. Zhong, R. Cui, Y. Guo, Y. Liang, S. Lu, Y. Wang, A. Saied, W. Chen, N. Duan. Agieval: A human-centric benchmark for evaluating foundation models, 2023.

[87] J. Zhou, T. Lu, S. Mishra, S. Brahma, S. Basu, Y. Luan, D. Zhou, L. Hou. Instruction-following evaluation for large language models, 2023.

Załącznik A

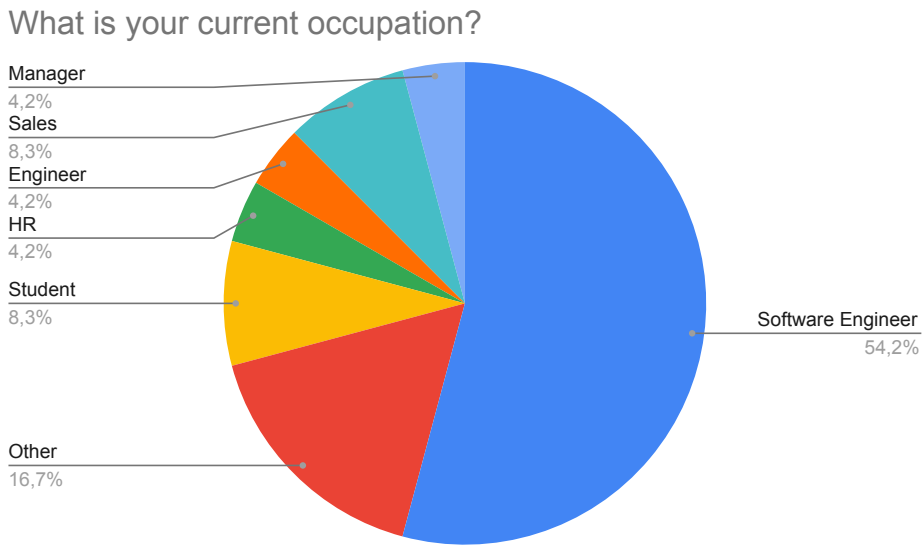
Kod źródłowy

Repozytorium zawierające dataset, wyniki oraz kod benchmarku **HELPS** znajduje się pod adresem: https://github.com/krzysztofMlczk/HELPS.

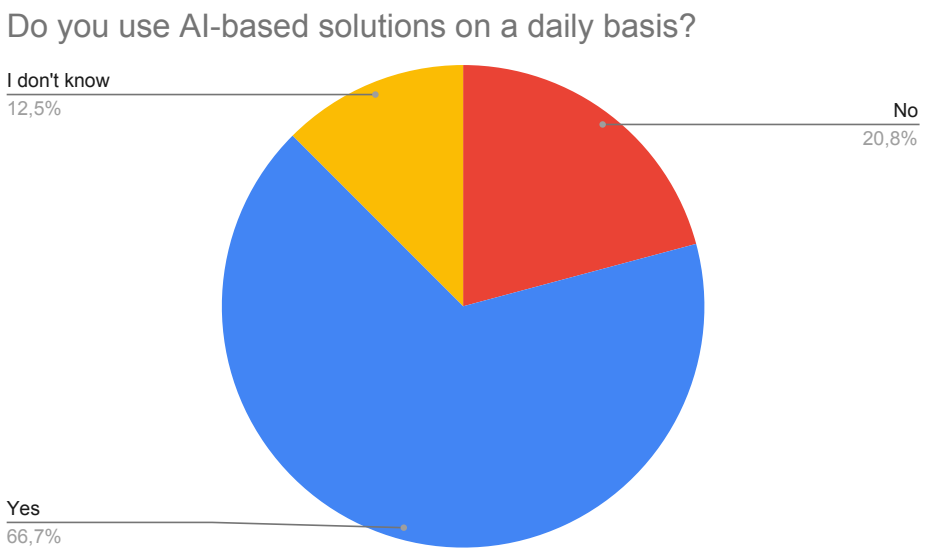
Załącznik B

Ankieta

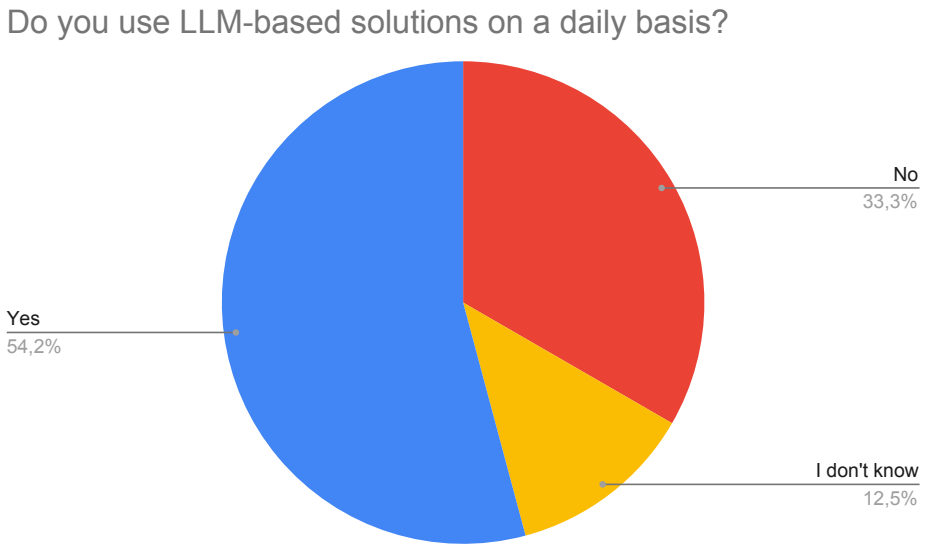
Poniżej znajdują się wyniki ankiety wykorzystanej do budowy datasetu benchmarku **HELPS**. Ankieta zrealizowana została w języku angielskim za pośrednictwem platformy Google Forms, dostępna jest pod adresem: https://forms.gle/wZ1wMrnynrbD93n69. W ankiecie udział wzięło 24 respondentów.



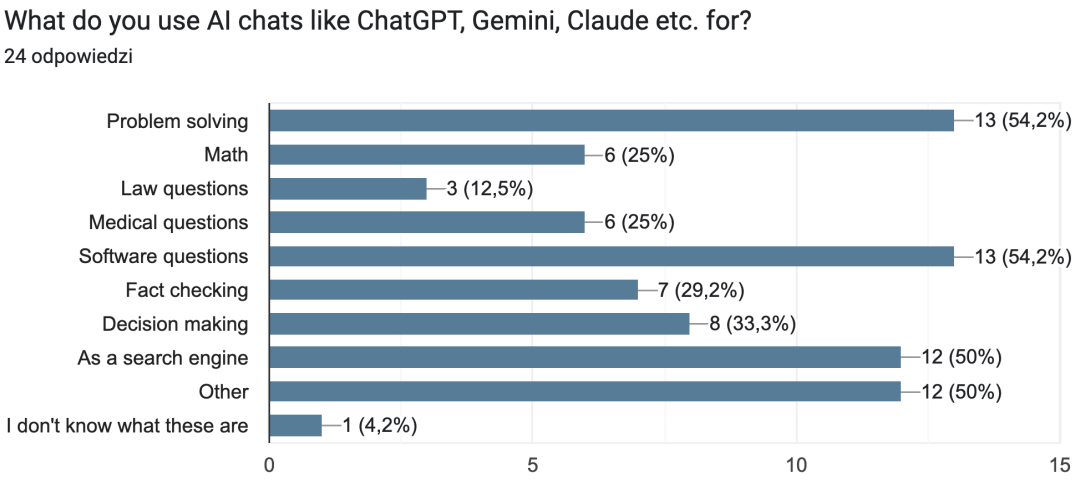
Rysunek B.1: Procentowy udział respondentów w poszczególnych grupach zawodach



Rysunek B.2: Świadome wykorzystanie rozwiązań opartych o sztuczną inteligencję w życiu codziennym



Rysunek B.3: Świadome wykorzystanie rozwiązań opartych o Wielkie Modele Językowe w życiu codziennym



Rysunek B.4: Cel wykorzystywania czatów opartych o Wielkie Modele Językowe

Załącznik C

Wynik pojedynczego testu ewaluacyjnego

Rysunek C.1 przedstawia przykład losowego testu ewaluacyjnego wykonywanego w ramach benchmarku **HELPS**. Wyniki wszystkich testów znajdują się w repozytorium wraz z kodem źródłowym (patrz Dodatek A).

```
{
  "task_id": "helps/2",
  "input": "I am the owner of an innovative carpentry shop making furniture combining wood and epoxy.
      ↪ Generate a short text (max. 5 sentences) to present my company on the website, at a conference or in
      ↪  conversations with potential business partners.",
  "output": "Welcome to our carpentry shop, where innovation meets tradition. We specialize in creating unique
      ↪ , high−quality furniture, seamlessly integrating wood and epoxy for a truly distinct aesthetic. With
      ↪ an unwavering commitment to craftsmanship and sustainability, our pieces not only add beauty to
      ↪ your spaces but also tell a story of durability and love for nature. Each piece we create is a testament
      ↪  to our passion, skill, and the transformative power of design. Partner with us and let's redefine the
      ↪ art of furniture making, one exquisite piece at a time.",
  "evaluation_results": [
    {
      "metric_name": "helpfulness",
      "score": 1.0,
      "reason": "The output directly addresses the main topic of presenting the carpentry shop, provides clear
          ↪ and relevant information about the shop's unique selling points, emphasizes actionable attributes
          ↪ like craftsmanship and sustainability, and enhances understanding by highlighting the shop's
          ↪ innovative approach and potential partnership.",
      "criteria": "Determine the helpfulness of the actual output in the context of input",
      "evaluation_steps": [
        "Check if the actual output directly addresses the main topic of the input.",
        "Evaluate if the actual output provides clear and relevant information in response to the input.",
        "Assess whether the actual output offers actionable or insightful suggestions related to the input.",
        "Determine if the actual output enhances understanding or resolves the query posed by the input."
      ],
    },
    {
      "metric_name": "relevancy",
      "score": 1.0,
      "reason": "The output effectively addresses key topics such as innovation, carpentry, and the combination
          ↪ of wood and epoxy. It provides relevant information suitable for a website, conference, or business
          ↪  discussions, maintaining consistency and completeness with the input request.",
      "criteria": "Determine the relevancy of the actual output in the context of input",
      "evaluation_steps": [
        "Compare the key topics or themes in the input and ensure they are addressed in the actual output.",
        "Check if the actual output provides information or answers that directly relate to the questions or
            ↪ requests in the input.",
        "Evaluate whether the actual output maintains contextual consistency with the input, ensuring logical
            ↪ flow and coherence.",
        "Assess the completeness of the actual output in covering all relevant aspects mentioned in the input."
      ],
    },
    {
      "metric_name": "brevity",
      "score": 0.9,
      "reason": "The actual output is succinct with 5 sentences as requested, effectively summarizing the
          ↪ company's focus on innovative furniture combining wood and epoxy. It maintains clarity and
          ↪ conveys the main points about craftsmanship, quality, and sustainability without unnecessary
          ↪ details.",
      "criteria": "Determine the brevity of the actual output in the context of input",
      "evaluation_steps": [
        "Compare the length of the actual output to the input to assess brevity.",
        "Evaluate whether the actual output sufficiently summarizes the input without unnecessary details.",
        "Check if the actual output conveys the main points of the input concisely.",
        "Determine if the actual output maintains clarity while being succinct in relation to the input."
      ],
    }
  ]
}
```


Rysunek C.1: Losowo wybrany przykład testu ewaluacyjnego benchmarku **HELPS** oraz jego punktacji dla każdej z metryk. Testowany model: GPT-4; Sędzia: GPT-4o (G-Eval) - kryteria oraz kroki widoczne powyżej