

Porównanie skuteczności komercyjnych LLMów w nauce języków obcych

Kamil Stenzel

23335

AKP

Porównanie skuteczności komercyjnych LLMów w nauce języków obcych	1
1. Wstęp	5
1.1. Cel pracy	5
1.2. Zakres i struktura pracy	5
1.3. Znaczenie badań nad nauką języków obcych	6
2. Czym są i jak działają LLMy	7
2.1. Definicja Large Language Models (LLMs)	7
2.2. Historia i rozwój LLMów	8
2.3. Mechanizmy działania (transformery, uczenie głębokie)	9
2.3.1. Embedding	9
2.3.2. Attention Mechanism	9
2.3.3. Transformery	10
2.3.4. Dekodowanie	10
2.3.5. Uczenie głębokie	11
2.4. Zastosowania LLMy w różnych dziedzinach	11
2.4.1. Przetwarzanie języka naturalnego (NLP)	11
2.4.2. Tłumaczenie maszynowe	12
2.4.3. Generowanie tekstu	12
2.4.4. Chatboty i wirtualni asystenci	12
2.4.5. Analiza danych i eksploracja wiedzy	12
2.4.6. Edukacja i nauka języków obcych	12
2.5. Ograniczenia LLMów	13
2.6. Modele wybrane do badania	14
3. Opis autorskiej aplikacji LinguaFranka	15
3.1. Idea i cel stworzenia LinguaFranka	15
3.2. Funkcjonalności i interfejs użytkownika	16
3.3. Architektura techniczna	18
3.3.1. Przegląd technologii i struktury	18

3.3.2. Backend.....	19
3.3.2.1. main.py.....	19
3.3.2.2. utility.py	21
3.3.3. Frontend	22
3.3.3.1. app.js	22
3.3.3.2. sanitise.js.....	23
3.3.3.3. Szablony HTML	24
3.3.3.3.1 base.html.....	24
3.3.3.3.2 index.html.....	25
3.3.3.3.3 lang.html.....	26
3.3.3.3.4 profile.html	27
3.3.3.3.5 settings.html	28
3.3.4. Konfiguracja i dane	28
3.3.4.1. config.yaml	28
3.3.4.2. Tłumaczenia.....	29
3.3.4.3. Profile użytkowników i logi czatów.....	29
3.3.5. Skrypty pomocnicze.....	29
3.3.5.1. install.bat.....	29
3.3.5.2. start.bat.....	30
3.4. Implementacja LLMów w LinguaFranka	30
3.4.1 Abstrakcja i spójność	30
3.4.2 LLM_abstract.py.....	32
3.4.3 GPT 3.5-turbo	32
3.4.3.1 Opis modelu.....	32
3.4.3.2 Implementacja API.....	33
3.4.3.3 Koszt API	34
3.4.4 Gemini 1.5-flash	34
3.4.4.1 Opis modelu.....	34
3.4.4.2 Implementacja API.....	34

3.4.4.3 Koszt API	35
3.4.5 Claude 3.5 Sonnet	35
3.4.5.1 Opis modelu.....	35
3.4.5.2 Implementacja API.....	35
3.4.5.3 Koszt API	36
4. Prompt Engineering	36
4.1 Definicja i znaczenie prompt engineering.....	36
4.2 Techniki tworzenia efektywnych promptów	36
4.3 Przykłady zastosowań prompt engineering w LinguaFranka	37
4.4 Wyzwania i przyszłość prompt engineering	39
5. Przykłady z aplikacji LinguaFranka	40
5.1. Przykład 1: Symulacja luźnej rozmowy.....	41
5.2. Przykład 2: Objasnienia gramatyczne	43
5.3. Przykład 3: Pomoc w komponowaniu zdań.....	46
5.4. Przykład 4: Specyficzny scenariusz rozmowy	50
6. Metodologia porównania skuteczności LLMy	52
6.1. Kryteria oceny.....	52
6.2. Opis grupy testowej	52
6.3. Proces zbierania opinii	53
6.4. Analiza danych	54
7. Wnioski z badań	55
7.1. Podsumowanie wyników porównania	55
7.2. Kluczowe obserwacje i refleksje.....	55
7.3. Praktyczne implikacje dla nauki języków obcych	55
7.4. Możliwe kierunki dalszych badań	55
8. Załączniki.....	55
8.1. Kwestionariusz użyty w badaniu	55
Kwestionariusz Oceny Modeli LLM w Nauce Języków	55
Informacje Użytkownika	55

1. Nazwa używanego modelu:.....	55
2. Język, którego się uczyłeś/aś:	55
Ocena Funkcjonalności.....	56
3. Rozmowa z AI w symulowanej sytuacji:	56
4. Tłumaczenie odpowiedzi bota:	57
5. Ocena zdań użytkownika:	57
6. Bezpośrednie pytania na temat języka:	57
Dodatkowe Uwagi.....	58
8.2. Dodatkowe materiały pomocnicze	58
8.3. Kod źródłowy aplikacji LinguaFranka	58

1. Wstęp

1.1. Cel pracy

Celem niniejszej pracy inżynierskiej jest stworzenie zasilanej sztuczną inteligencją aplikacji do nauki języków oraz przeprowadzenie wszechstronnego porównania skuteczności komercyjnych modeli językowych (Large Language Models, LLMy) w kontekście nauki języków obcych. W dobie coraz szybszego rozwoju technologii sztucznej inteligencji, LLMy zyskują na znaczeniu jako narzędzia wspomagające edukację, w tym również naukę języków. Praca ma na celu ocenę, które z dostępnych komercyjnych modeli oferują najlepsze wyniki w nauce języków oraz jakie są ich mocne i słabe strony.

1.2. Zakres i struktura pracy

Praca składa się z kilku kluczowych części, które prowadzą czytelnika od podstawowych informacji na temat LLMów, przez szczegółowy opis stworzonej aplikacji LinguaFranka, aż po metodologię badań i analizę uzyskanych wyników. Struktura pracy jest następująca:

Czym są i jak działają LLMy – szczegółowe omówienie technologii LLMy, ich historii, mechanizmów działania i zastosowań.

Opis aplikacji LinguaFranka – prezentacja stworzonej aplikacji, która jest wykorzystywana do testowania skuteczności LLMy w nauce języków.

Prompt Engineering – wyjaśnienie pojęcia prompt engineering, jego technik oraz znaczenia w kontekście stosowania LLMy.

Przykłady z aplikacji LinguaFranka – konkretne przykłady zastosowań LLMy w LinguaFranka, ilustrujące różne aspekty nauki języków.

Metodologia porównania skuteczności LLMy – opis metodologii zastosowanej do oceny skuteczności LLMy, w tym kryteria oceny i sposób zbierania opinii od grupy testowej.

Wnioski z badań – podsumowanie wyników badań, kluczowe obserwacje oraz praktyczne implikacje dla nauki języków obcych.

1.3. Znaczenie badań nad nauką języków obcych

Nauka języków obcych jest jednym z kluczowych elementów globalnej edukacji i komunikacji. W dobie globalizacji umiejętność posługiwania się więcej niż jednym językiem jest nie tylko atutem, ale często koniecznością. Tradycyjne metody nauki języków, choć skuteczne, są często czasochłonne i wymagają dużego zaangażowania zarówno ze strony nauczyciela, jak i ucznia. LLMy, dzięki swojej zdolności do przetwarzania i generowania naturalnego języka, oferują nową jakość w nauczaniu, umożliwiając bardziej interaktywne, spersonalizowane i efektywne podejście do nauki języków.

Przeprowadzenie badań nad skutecznością komercyjnych LLMy w nauce języków obcych ma więc ogromne znaczenie praktyczne. Pozwoli to nie tylko na ocenę dostępnych narzędzi, ale także na wskazanie kierunków, w których powinny rozwijać się przyszłe technologie edukacyjne. Wyniki tych badań mogą przyczynić się do lepszego zrozumienia potencjału LLMy w edukacji i dostarczyć cennych wskazówek dla twórców oprogramowania, nauczycieli oraz uczniów.

2. Czym są i jak działają LLMy

2.1. Definicja Large Language Models (LLMs)

Large Language Models (LLMs) to zaawansowane modele językowe oparte na technikach uczenia maszynowego, które są trenowane na ogromnych zbiorach danych tekstowych. Ich głównym celem jest generowanie, rozumienie i przetwarzanie języka naturalnego w sposób zbliżony do ludzkiego.

LLMy potrafią wykonywać różnorodne zadania związane z językiem, takie jak:

- Generowanie tekstu : Tworzenie spójnych i logicznych fragmentów tekstu na podstawie dostarczonego kontekstu.
- Rozumienie kontekstu : Analiza tekstu w celu zrozumienia jego sensu i intencji autora.
- Tłumaczenia automatyczne : Przekład tekstu z jednego języka na inny z zachowaniem sensu i stylu.
- Analiza sentymentu : Ocenianie emocji wyrażanych w tekście, takich jak pozytywne lub negatywne nastawienie.
- Odpowiadanie na pytania : Dostarczanie precyzyjnych odpowiedzi na podstawie zadanego pytania i dostępnych informacji.

Przykłady LLMów obejmują:

- GPT-3 : Model opracowany przez OpenAI, znany ze swojej zdolności do generowania naturalnego języka oraz wykonywania złożonych zadań językowych.

- BERT (Bidirectional Encoder Representations from Transformers) : Model stworzony przez Google, który specjalizuje się w rozumieniu kontekstu i znaczenia słów w zdaniu.
- T5 (Text-to-Text Transfer Transformer) : Model, który ujednolica różne zadania przetwarzania języka naturalnego w formie problemów tekst-do-tekstu.

LLMy są wykorzystywane w różnych aplikacjach, od chatbotów po tłumaczenia automatyczne, co czyni je wszechstronnymi narzędziami w dziedzinie przetwarzania języka naturalnego.

2.2. Historia i rozwój LLMów

Rozwój Large Language Models (LLMs) rozpoczął się od prostych modeli statystycznych, takich jak modele n-gramowe, które analizowały sekwencje słów, przewidując kolejne słowo na podstawie poprzednich. Te modele były jednak ograniczone w zrozumieniu szerszego kontekstu.

Z czasem, dzięki postępom w uczeniu maszynowym i rosnącej mocy obliczeniowej, zaczęto stosować bardziej zaawansowane techniki, takie jak sieci neuronowe. Kluczowe etapy obejmują:

- Rekurencyjne sieci neuronowe (RNNs) : Umożliwiały przetwarzanie sekwencyjnych danych, ale miały problemy z długoterminowym zapamiętywaniem informacji.
- Długie krótkoterminowe pamięci (LSTM) : Ulepszona wersja RNNs, która potrafiła lepiej przechowywać informacje w dłuższych sekwencjach, rozwiązując problem zanikającego gradientu.

Przełomem było wprowadzenie architektury Transformer przez Vaswaniego i in. w 2017 roku. Architektura ta zrewolucjonizowała przetwarzanie języka naturalnego dzięki użyciu mechanizmu self-attention, który pozwala modelom lepiej rozumieć kontekst i relacje między słowami w całym zdaniu, niezależnie od ich położenia.

Dzięki Transformerom powstały zaawansowane LLM-y, takie jak:

- GPT-2 i GPT-3 : Modele generatywne, które potrafią tworzyć wysokiej jakości tekst, wykazując zdolność do rozwiązywania różnorodnych zadań językowych.
- BERT : Skupiający się na dwukierunkowym rozumieniu kontekstu, co jest szczególnie przydatne w zadaniach wymagających precyzyjnego zrozumienia tekstu.

Transformery umożliwiły rozwój LLMów, które są teraz fundamentem wielu nowoczesnych aplikacji w sztucznej inteligencji i przetwarzaniu języka naturalnego.

2.3. Mechanizmy działania (transformery, uczenie głębokie)

Mechanizmy działania LLM-ów opierają się na zaawansowanych technikach uczenia głębokiego, z kluczową rolą architektury transformerów. Proces ten można podzielić na kilka kluczowych etapów:

2.3.1. Embedding

Embedding to proces przekształcania słów lub tokenów w wektory liczbowe. Jest to fundamentalny krok w przetwarzaniu języka naturalnego, który pozwala modelom na pracę z tekstem w formie matematycznej.

Zaawansowane techniki embeddingu, takie jak BERT embeddings, uwzględniają nie tylko pojedyncze słowa, ale także ich kontekst w zdaniu. Dzięki temu model może rozróżniać różne znaczenia tego samego słowa w zależności od kontekstu.

2.3.2. Attention Mechanism

Attention Mechanism to innowacyjne podejście, które pozwala modelowi na dynamiczne skupianie się na różnych częściach wejścia podczas przetwarzania sekwencji. W praktyce, mechanizm attention oblicza wagi dla każdego elementu sekwencji, określając jego istotność dla danego zadania.

Multi-head self-attention, używany w transformerach, pozwala modelowi na jednoczesne analizowanie różnych aspektów relacji między słowami. Każda "głowa" attention może skupić się na innym aspekcie relacji, co zwiększa zdolność modelu do uchwycenia złożonych zależności w tekście.

2.3.3. Transformery

Transformery to architektura, która zrewolucjonizowała przetwarzanie języka naturalnego. Składają się z warstw enkodera i/lub dekodera, z których każda zawiera mechanizm self-attention i warstwę feed-forward.

Kluczową zaletą transformerów jest ich zdolność do przetwarzania całych sekwencji równolegle, co znacznie przyspiesza proces uczenia i inference w porównaniu do wcześniejszych architektur sekwencyjnych, takich jak RNN czy LSTM.

2.3.4. Dekodowanie

Proces dekodowania w LLM-ach jest kluczowy dla generowania tekstu. Wykorzystuje on warstwę softmax do przekształcenia wektorów wyjściowych w rozkład prawdopodobieństwa nad słownikiem.

Podczas generowania tekstu, model może wykorzystywać różne strategie, takie jak beam search czy sampling, aby wybrać kolejne słowa na podstawie tego rozkładu prawdopodobieństwa.

2.3.5. Uczenie głębokie

LLM-y są trenowane przy użyciu technik uczenia głębokiego, co obejmuje:

1. Optymalizacja : Wykorzystanie zaawansowanych algorytmów optymalizacji, takich jak Adam, do aktualizacji parametrów modelu.
2. Regularyzacja : Techniki takie jak dropout czy L2 regularyzacja, zapobiegające przeuczeniu modelu.
3. Transfer learning : Wykorzystanie wiedzy zdobytej podczas treningu na ogólnych zadaniach do specyficznych zastosowań, co znacznie skraca czas i koszty treningu.
4. Uczenie nienadzorowane : Wykorzystanie ogromnych ilości nieustrukturyzowanych danych tekstowych do nauki reprezentacji językowych.

Połączenie tych zaawansowanych technik uczenia głębokiego z architekturą transformerów pozwala LLM-om na osiągnięcie niezwyklej wydajności w różnorodnych zadaniach związanych z przetwarzaniem języka naturalnego.

2.4. Zastosowania LLMy w różnych dziedzinach

2.4.1. Przetwarzanie języka naturalnego (NLP)

LLMy są szeroko stosowane w przetwarzaniu języka naturalnego do takich zadań jak analiza sentymentu, klasyfikacja tekstu, ekstrakcja informacji, czy rozpoznawanie nazw własnych (NER). Dzięki ich zdolności do rozumienia i generowania tekstów, są one kluczowym narzędziem w automatyzacji wielu procesów językowych.

2.4.2. Tłumaczenie maszynowe

Nowoczesne modele LLM, takie jak T5 czy M2M-100, są w stanie tłumaczyć teksty z jednego języka na inny z wysoką dokładnością. Dzięki mechanizmowi attention, są one w stanie uchwycić kontekst i subtelności językowe, co prowadzi do lepszej jakości tłumaczeń w porównaniu do tradycyjnych metod.

2.4.3. Generowanie tekstu

LLMy, takie jak GPT-3, są używane do generowania tekstu na potrzeby różnych aplikacji, od tworzenia treści marketingowych, przez pisanie artykułów, aż po generowanie dialogów w grach komputerowych. Modele te potrafią tworzyć teksty, które są spójne, kreatywne i często nie do odróżnienia od tekstów pisanych przez ludzi.

2.4.4. Chatboty i wirtualni asystenci

Dzięki LLMom, chatboty i wirtualni asystenci, tacy jak Google Assistant czy Amazon Alexa, mogą prowadzić bardziej naturalne i zrozumiałe rozmowy z użytkownikami. LLMy umożliwiają tym systemom rozumienie pytań, dostarczanie odpowiednich odpowiedzi oraz wykonywanie złożonych poleceń.

2.4.5. Analiza danych i eksploracja wiedzy

LLMy są także wykorzystywane do analizy dużych zbiorów danych tekstowych, takich jak artykuły naukowe, raporty biznesowe czy posty na mediach społecznościowych. Mogą one pomagać w ekstrakcji kluczowych informacji, identyfikacji trendów oraz generowaniu podsumowań.

2.4.6. Edukacja i nauka języków obcych

LLMy, takie jak GPT-4, znajdują zastosowanie w edukacji, m.in. w nauce języków obcych. Mogą one tworzyć interaktywne ćwiczenia, symulować rozmowy w obcym języku oraz dostarczać spersonalizowane materiały edukacyjne, co znacznie ułatwia proces nauki.

2.5. Ograniczenia LLMów

Mimo imponujących możliwości, Large Language Models (LLMy) mają szereg istotnych ograniczeń:

1. Brak rzeczywistego zrozumienia : LLMy operują na statystycznych wzorcach w danych, nie posiadając głębszego zrozumienia znaczenia czy kontekstu. Mogą generować przekonujące teksty, ale nie rozumieją ich tak, jak ludzie.
2. Ograniczenia czasowe danych treningowych : Modele są ograniczone do wiedzy zawartej w danych, na których zostały wytrenowane. Nie posiadają aktualnych informacji poza datą cutoff swoich danych treningowych.
3. Halucynacje i nieprawdziwe informacje : LLMy mogą generować przekonujące, ale błędne lub zmyślane informacje, szczególnie gdy są pytane o tematy wykraczające poza ich wiedzę.
4. Brak pamięci długoterminowej : Modele nie pamiętają wcześniejszych interakcji ani nie uczą się z nich w sposób ciągły.
5. Ograniczenia etyczne i stronniczość : LLMy mogą powielać lub wzmacniać uprzedzenia obecne w danych treningowych, co prowadzi do potencjalnie stronniczych lub nieetycznych odpowiedzi.
6. Trudności z rozumowaniem abstrakcyjnym : Chociaż LLMy mogą wykonywać proste operacje logiczne, mają trudności z bardziej złożonym rozumowaniem abstrakcyjnym czy rozwiązywaniem problemów.
7. Ograniczenia językowe i kulturowe : Mimo że są wielojęzyczne, LLMy mogą mieć trudności z subtelnościami kulturowymi i kontekstem specyficznym dla danej kultury.

8. Wysokie koszty obliczeniowe : Trening i uruchamianie LLMów wymaga znacznych zasobów obliczeniowych, co przekłada się na wysokie koszty finansowe i środowiskowe.

9. Problemy z prywatnością i bezpieczeństwem : Wykorzystanie LLMów rodzi obawy dotyczące prywatności danych użytkowników oraz potencjalnego wykorzystania do celów dezinformacji.

10. Ograniczenia w zadaniach wymagających aktualnej wiedzy : LLMy nie są odpowiednie do zadań wymagających dostępu do aktualnych informacji lub danych w czasie rzeczywistym

2.6. Modele wybrane do badania

W ramach badania wybrano cztery wiodące modele językowe:

GPT-3.5 (OpenAI): Jeden z najbardziej znanych i szeroko stosowanych modeli, oferujący wysoką jakość generowanego tekstu i wszechstronność zastosowań.

Gemini 1.5 Flash(Google): Najnowszy model od Google, znany z zaawansowanych możliwości multimodalnych i wysokiej wydajności w różnorodnych zadaniach.

LLaMA 3 (Meta): Otwarty model od Meta, który zyskał popularność dzięki swojej elastyczności i możliwościom dostosowywania.

Claude 3.5 Sonnet (Anthropic): Model znany z zaawansowanych zdolności rozumowania i przestrzegania zasad etycznych.

Wybór tych modeli opiera się na kilku kluczowych kryteriach:

Dostępność komercyjna: Wszystkie wybrane modele są dostępne poprzez publiczne API, co umożliwia ich łatwe wykorzystanie w badaniach i aplikacjach.

Reprezentatywność: Modele pochodzą od czołowych producentów w branży AI, co zapewnia reprezentatywny przegląd obecnego stanu technologii.

Balans między mocą a kosztem: Wybór konkretnych wersji modeli (np. GPT-3 zamiast GPT-4) był podyktowany optymalnym stosunkiem mocy obliczeniowej do kosztów użytkowania.

Skalowalność: Wybrane modele oferują wystarczającą moc obliczeniową do przeprowadzenia zaawansowanych badań, bez konieczności inwestowania w kosztowny sprzęt lokalny.

Warto zauważyć, że słabsze modele, takie jak 7B czy 13B parametrowe wersje, które można uruchomić lokalnie, zostały rozważone, ale ostatecznie odrzucone. Głównym powodem była potrzeba zaawansowanego sprzętu (mocnych kart graficznych) do ich efektywnego wykorzystania. Ponadto, założono, że modele te byłyby mniej skuteczne w porównaniu do wybranych, ze względu na znaczącą różnicę w mocy obliczeniowej.

Wybór tych konkretnych modeli pozwala na przeprowadzenie kompleksowej analizy możliwości i ograniczeń obecnie dostępnych LLM-ów, jednocześnie zachowując praktyczność i efektywność kosztową badania.

3. Opis autorskiej aplikacji LinguaFranka

3.1. Idea i cel stworzenia LinguaFranka

LinguaFranka powstała z myślą o wspieraniu osób uczących się języków obcych w sposób nowoczesny i dostosowany do indywidualnych potrzeb. W dobie globalizacji umiejętność komunikacji w różnych językach jest kluczowa zarówno w życiu zawodowym, jak i prywatnym. Tradycyjne metody nauki języków, choć skuteczne, często bywają czasochłonne i kosztowne. Ponadto, wiele osób odczuwa stres i niepewność podczas rozmowy z native speakerami, co może hamować postępy w nauce.

LinguaFranka oferuje rozwiązanie tych problemów dzięki integracji z technologią sztucznej inteligencji. Aplikacja umożliwia użytkownikom interakcję z wirtualnym rozmówcą, który symuluje dialogi w wybranym języku. Takie podejście pozwala na:

Praktykę w bezstresowym środowisku: Użytkownicy mogą rozmawiać o dowolnych tematach bez obawy o ocenę ze strony drugiej osoby.

Elastyczność nauki: Aplikacja umożliwia naukę w dowolnym czasie i miejscu, dostosowując się do tempa i stylu użytkownika.

Dostęp do wielu języków: Dzięki zaawansowanym modelom językowym, użytkownicy mają możliwość nauki różnych języków, co jest szczególnie przydatne przy nauce mniej popularnych języków.

Personalizacja: System może dostosowywać poziom trudności i tematykę rozmów do indywidualnych potrzeb użytkownika, co zwiększa efektywność nauki.

Celem LinguaFranka jest ułatwienie dostępu do nauki języków obcych, zwiększenie pewności siebie użytkowników w posługiwaniu się nowymi językami oraz wzbogacenie dotychczasowych metod nauki o nowoczesne technologie. Aplikacja jest pomyślana jako uzupełnienie tradycyjnych metod nauki, oferując dodatkowe narzędzie, które wspiera rozwój kompetencji językowych.

3.2. Funkcjonalności i interfejs użytkownika

Aplikacja LinguaFranka została zaprojektowana z myślą o intuicyjności i wszechstronności, oferując użytkownikom szereg funkcji wspierających naukę języków obcych. Poniżej przedstawiono kluczowe funkcjonalności aplikacji oraz opis interfejsu użytkownika:

Funkcjonalności

1. Czat właściwy

- Umożliwia użytkownikom prowadzenie rozmów w wybranym języku z wirtualnym rozmówcą.
- Dostosowuje poziom trudności do umiejętności użytkownika.
- Zawiera opcję wyboru tematu rozmowy, co pozwala na koncentrację na interesujących użytkownika zagadnieniach.

2. Możliwość zadania pytania na temat języka

- Użytkownicy mogą zadawać pytania dotyczące gramatyki, słownictwa czy wymowy.

- System udziela jasnych i zwięzłych odpowiedzi, przykładowo wyjaśniając różnice między czasami "was" i "were".

3. Objaśnienie tekstu wygenerowanego przez AI

- AI może generować teksty w wybranym języku, które następnie są szczegółowo objaśniane pod względem gramatycznym i leksykalnym.

- Funkcja ta wspiera głębsze zrozumienie struktury języka.

4. 'Ocena' tekstu użytkownika wykonana przez AI

- Użytkownicy mogą wprowadzać własne teksty, które AI ocenia pod kątem poprawności gramatycznej i stylistycznej.

- System zapewnia szczegółowy feedback oraz wskazówki dotyczące poprawy jakości pisania.

Interfejs użytkownika

- Prosty i intuicyjny design : Interfejs został zaprojektowany tak, aby był łatwy w obsłudze nawet dla osób mniej obeznanych z technologią.

- Dashboard główny : Zawiera szybki dostęp do wszystkich funkcji aplikacji oraz możliwość śledzenia postępów w nauce.

- Sekcja czatu : Interaktywny obszar pozwalający na prowadzenie rozmów z wirtualnym rozmówcą, z opcjami wyboru języka i tematu.

- Panel pytań i odpowiedzi : Miejsce, gdzie użytkownicy mogą zadawać pytania dotyczące języka i otrzymywać odpowiedzi w czasie rzeczywistym.

- Objaśnienia i oceny : Zintegrowane narzędzia do analizy tekstu, oferujące szczegółowe wyjaśnienia i oceny wprowadzanych treści.

LinguaFranka dąży do stworzenia przyjaznego środowiska edukacyjnego, które wspiera użytkowników na każdym etapie ich językowej podróży.

3.3. Architektura techniczna

Aplikacja LinguaFranka została zbudowana z wykorzystaniem nowoczesnych technologii webowych, z naciskiem na modułowość i skalowalność. Oto przegląd architektury technicznej:

3.3.1. Przegląd technologii i struktury

Technologie:

Backend: Python (FastAPI)

Frontend: JavaScript, HTML, CSS

Konfiguracja: YAML, JSON

Struktura dwupoziomowa:

Poziom wysoki: Abstrakcyjna obsługa aplikacji, niezależna od konkretnego modelu AI

Poziom niski: Implementacja konkretnych modeli AI, obsługa API i promptów

Kluczowe komponenty:

a) Backend (Python):

main.py: Główna logika aplikacji, implementacja endpointów FastAPI

utility.py: Funkcje pomocnicze i abstrakcyjne wywołania LLM

LLM_abstract.py: Moduł pośredniczący między abstrakcją a konkretnymi modelami

API_interfaces/common.py: Wspólne funkcje promptów dla wszystkich modeli

API_interfaces/*_tools.py: Implementacje konkretnych modeli AI

b) Frontend:

static/app.js: Implementacja frontendowa czatu

static/*.css: Style aplikacji

templates/: Szablony HTML podstron

c) Konfiguracja i dane:

config.yaml: Konfiguracja trybu nauki i ustawień aplikacji

API_keys.py: Klucze API do modeli AI

translations/: Pliki JSON z tłumaczeniami interfejsu

user-profiles/: Profile użytkowników

chat_logs/: Zapisane rozmowy

Skrypty pomocnicze:

install.bat: Skrypt instalacyjny (weryfikacja/instalacja Pythona i modułów)

start.bat: Skrypt uruchamiający aplikację

Modułowość i skalowalność:

Separacja logiki wysokiego i niskiego poziomu umożliwia łatwe dodawanie nowych modeli AI

Wykorzystanie FastAPI zapewnia wysoką wydajność i łatwość rozbudowy endpointów

Struktura folderów ułatwia zarządzanie różnymi aspektami aplikacji

Bezpieczeństwo:

Klucze API przechowywane są oddzielnie w API_keys.py

Logi czatów i profile użytkowników są izolowane w osobnych folderach

Wielojęzyczność:

System tłumaczeń oparty na tagach Jinja2 i plikach JSON umożliwia łatwe dodawanie nowych języków interfejsu

3.3.2. Backend

3.3.2.1. *main.py*

Moduł main.py stanowi główny punkt wejścia aplikacji FastAPI, implementując logikę routingu, obsługę WebSockets oraz integrację z szablonami HTML. Poniżej przedstawiono kluczowe elementy i funkcjonalności tego modułu:

Importy i konfiguracja:

Moduł wykorzystuje biblioteki takie jak logging, FastAPI, Jinja2Templates oraz własne narzędzia z modułu utility.

Konfiguracja logowania jest inicjowana na poziomie INFO.

Aplikacja FastAPI jest tworzona i konfigurowana do obsługi plików statycznych i szablonów.

Endpointy HTTP:

/: Strona główna, renderuje index.html z danymi czatu i tłumaczeniami.

/profile: Strona profilu użytkownika, wyświetla i umożliwia edycję profili.

/lang: Strona wyboru języka praktyki.

/settings: Strona ustawień aplikacji.

Logika biznesowa:

change_user_profile: Aktualizuje profil użytkownika.

add_user_profile: Dodaje nowy profil użytkownika.

choose_language: Obsługuje wybór języka do praktyki.

change_settings: Aktualizuje ustawienia aplikacji.

Obsługa WebSocket:

Endpoint /ws obsługuje komunikację w czasie rzeczywistym.

Przetwarza przychodzące wiadomości JSON i wysyła odpowiedzi.

Zarządzanie czatem:

reset_chat: Resetuje historię czatu dla wybranego języka.

Obsługa błędów i logowanie:

Wszechstronne logowanie zdarzeń i błędów.

Obsługa wyjątków dla zapewnienia stabilności aplikacji.

Integracja z systemem plików:

Zarządzanie plikami logów czatu i profilami użytkowników.

Wielojęzyczność:

Wsparcie dla tłumaczeń i automatycznego tłumaczenia treści.

Responsywność:

Wykorzystanie szablonów HTML do renderowania stron.

Moduł `main.py` integruje różne komponenty aplikacji, zapewniając spójny interfejs użytkownika i efektywną obsługę żądań. Wykorzystuje zaawansowane funkcje FastAPI, takie jak WebSockets i obsługa formularzy, jednocześnie implementując własną logikę biznesową dla specyficznych potrzeb aplikacji do nauki języków.

3.3.2.2. utility.py

Ten moduł zawiera zestaw funkcji do zarządzania konfiguracją aplikacji, przetwarzania danych czatu oraz obsługi tłumaczeń. Główne funkcjonalności obejmują:

Zarządzanie konfiguracją:

Wczytywanie konfiguracji z pliku YAML (`load_config`)

Aktualizacja konfiguracji (`update_config`, `update_apis`)

Pobieranie i ustawianie języka systemowego (`get_system_language_iso_639_1`)

Obsługa profili użytkowników:

Wczytywanie profili (`load_user_profile`)

Aktualizacja profili (`update_user_profile`)

Pobieranie listy dostępnych profili (`get_profile_list`)

Zarządzanie tłumaczeniami:

Wczytywanie tłumaczeń (`load_translations`)

Generowanie globalnych tłumaczeń (`get_global_translation`)

Konwersja między kodami języków a pełnymi nazwami (`lang_to_code`, `code_to_lang`)

Obsługa logów czatu:

Inicjalizacja logów (`initialize_chat_log`)

Dodawanie wpisów do logów (`append_chatlog`)

Pobieranie kontekstu czatu (`get_context_to_n`, `get_context_to_token_limit`)

Przetwarzanie danych czatu:

Obsługa wiadomości (`handle_message`)

Obsługa pytań (`handle_question`)

Obsługa próśb o wyjaśnienie (`handle_explain`)

Moduł wykorzystuje zewnętrzne biblioteki, takie jak `json`, `yaml`, `httpx`, `tiktoken` oraz własne moduły do interakcji z modelami językowymi. Funkcje są zaprojektowane do współpracy z interfejsem użytkownika, umożliwiając dynamiczne dostosowywanie ustawień aplikacji, zarządzanie profilami użytkowników oraz przetwarzanie interakcji w czacie.

Kluczowe aspekty implementacji obejmują:

Elastyczne zarządzanie konfiguracją poprzez pliki `YAML`

Obsługę wielu języków i tłumaczeń

Dynamiczne zarządzanie profilami użytkowników

Efektywne przetwarzanie i przechowywanie logów czatu

Integrację z modelami językowymi do generowania odpowiedzi

3.3.3. Frontend

3.3.3.1. *app.js*

Ten moduł implementuje funkcjonalność komunikacji `WebSocket` oraz interakcji z interfejsem użytkownika dla aplikacji czatu. Główne elementy obejmują:

Ustanowienie i zarządzanie połączeniem `WebSocket`:

Funkcja `connectWebSocket()` inicjuje połączenie `WebSocket` z serwerem.

Obsługuje zdarzenia otwarcia, zamknięcia, błędów i odbierania wiadomości.

Implementuje mechanizm ponownego łączenia z wykładniczym opóźnieniem w przypadku rozłączenia.

Przetwarzanie otrzymanych wiadomości:

Funkcja `ws.onmessage` parsuje otrzymane dane JSON i kieruje je do odpowiednich funkcji obsługi.

`handleChatMessage()` aktualizuje historię czatu.

`handleQuestionResponse()` wyświetla odpowiedzi na pytania.

`handleExplanationResponse()` pokazuje wyjaśnienia.

Wysyłanie wiadomości:

`sendMessageWrapper()` i `sendMessage()` obsługują wysyłanie wiadomości czatu.

`submitQuestionWrapper()` i `submitQuestion()` zajmują się wysyłaniem pytań.

Manipulacja interfejsem użytkownika:

Funkcje `showLoadingSpinner()` i `hideLoadingSpinner()` kontrolują widoczność wskaźnika ładowania.

`parseMessage()` formatuje wiadomości czatu do wyświetlenia, dodając klasy CSS i atrybuty data.

Interakcje użytkownika:

`handleMessageClick()` obsługuje kliknięcia na wiadomości, wysyłając żądanie wyjaśnienia.

`resetChat()` umożliwia zresetowanie czatu poprzez wywołanie odpowiedniego endpointu API.

Inicjalizacja:

Skrypt automatycznie wywołuje `connectWebSocket()` przy załadowaniu.

Po załadowaniu strony, chatbox jest przewijany na sam dół.

Kod wykorzystuje asynchroniczne operacje, obsługę zdarzeń i dynamiczną aktualizację DOM do stworzenia responsywnego i interaktywnego interfejsu czatu. Implementuje również mechanizmy obsługi błędów i ponownego łączenia, zapewniając odporność na problemy z połączeniem.

3.3.3.2. *sanitize.js*

Kod implementuje uniwersalną obsługę formularzy na stronie, wykorzystując nowoczesne techniki JavaScript. Główne cechy tej implementacji to:

Przechwytywanie wszystkich formularzy: Skrypt automatycznie identyfikuje wszystkie formularze na stronie za pomocą `document.querySelectorAll('form')`.

Zapobieganie domyślnemu zachowaniu: Dla każdego formularza dodawane jest nasłuchiwanie zdarzenia `submit`. Domyślne zachowanie przesyłania formularza jest blokowane za pomocą `event.preventDefault()`.

Zbieranie danych formularza: Dane z formularza są gromadzone przy użyciu obiektu `FormData`. Następnie są one przekształcane w prosty obiekt JavaScript.

Sanityzacja danych: Wszystkie zebrane dane są sanityzowane przy użyciu biblioteki `DOMPurify`, co zwiększa bezpieczeństwo poprzez usunięcie potencjalnie niebezpiecznych elementów HTML i skryptów.

Dynamiczne określanie endpointu: Adres URL, na który mają być wysłane dane, jest pobierany z atrybutu `action` formularza.

Asynchroniczne wysyłanie danych: Dane są wysyłane asynchronicznie za pomocą funkcji `fetch()`. Używana jest metoda `POST`, a dane są przekazywane w formacie `JSON`.

Obsługa odpowiedzi: Po udanym wystąpieniu danych, odpowiedź serwera jest logowana w konsoli. W przypadku błędu, informacja o nim również jest zapisywana w konsoli.

Ten kod zapewnia elastyczne i bezpieczne rozwiązanie do obsługi formularzy, które można łatwo zaimplementować na dowolnej stronie internetowej. Dzięki wykorzystaniu nowoczesnych API przeglądarki, takich jak `FormData` i `fetch`, oraz biblioteki `DOMPurify` do sanityzacji danych, skrypt oferuje wydajne i bezpieczne przetwarzanie danych formularza.

3.3.3.3. Szablony HTML

3.3.3.3.1 base.html

Plik ten stanowi szablon bazowy HTML dla strony internetowej. Wykorzystuje on składnię szablonów, co umożliwia dynamiczne generowanie treści.

Struktura dokumentu zaczyna się od deklaracji i zawiera standardowe elementy HTML5. W sekcji `<head>` zdefiniowano metadane, w tym kodowanie znaków i `viewport` dla responsywności. Tytuł strony jest dynamiczny, wykorzystujący blok `{% block title %}`, który może być nadpisywany w szablonach potomnych.

Link do arkusza stylów `base.css` jest dołączony, a dodatkowy blok `{% block head %}` umożliwia dołączanie dodatkowych zasobów w szablonach potomnych.

Ciało dokumentu (`<body>`) zawiera trzy główne sekcje:

1. Nagłówek (header): Zawiera nawigację z czterema odnośnikami (Strona główna, Profil, Wybór języka, Ustawienia). Teksty nawigacji są dynamicznie pobierane z obiektu `content.nav`, co sugeruje wsparcie dla wielu języków.
2. Główna treść (main): Zawarta w elemencie `<main>` z klasą `main-content`. Blok `{% block content %}` pozwala na wstawienie specyficznej treści w szablonach potomnych.
3. Stopka (footer): Zawiera prosty komunikat o prawach autorskich.

Szablon wykorzystuje zmienne (np. `{{content.nav.home}}`) do dynamicznego wstawiania treści, co ułatwia lokalizację i personalizację strony.

Struktura ta zapewnia spójny układ dla wszystkich stron, jednocześnie umożliwiając elastyczność w definiowaniu specyficznej zawartości dla poszczególnych podstron poprzez wykorzystanie mechanizmu dziedziczenia szablonów.

3.3.3.3.2 index.html

Ten plik stanowi główny szablon strony aplikacji. Wykorzystuje on system szablonów Jinja2, rozszerzając bazowy szablon `'base.html'`. Strona składa się z trzech głównych sekcji: obszaru wyjaśnień nauczyciela, okna czatu i pola do zadawania pytań.

Szablon definiuje bloki `'title'` i `'head'`, gdzie osadzany jest tytuł strony oraz link do arkusza stylów CSS. W bloku `'content'` znajduje się główna struktura strony.

Strona korzysta z biblioteki `DOMPurify` do sanitzacji danych wejściowych, co zwiększa bezpieczeństwo aplikacji. Skrypt sanitzacyjny jest ładowany z pliku `'sanitise.js'`.

Główny kontener (`'mainContainer'`) zawiera trzy sekcje:

1. Obszar wyjaśnień nauczyciela (`'teacherBox'`):
 - Zawiera tytuł i wewnętrzny `div` do wyświetlania wyjaśnień.
2. Okno czatu (`'chatBox'`):
 - Składa się z tytułu, wewnętrznego `diva` do wyświetlania wiadomości oraz formularza do wprowadzania tekstu.
 - Formularz zawiera pole tekstowe, przycisk wysyłania i przycisk resetowania czatu.
 - Implementuje funkcje `'sendMessageWrapper'` i `'resetChat'`.

- Zawiera animowany spinner ładowania.
3. Pole do zadawania pytań ('questionBox'):
- Zawiera wewnętrzny div do wyświetlania pytań i formularz do ich zadawania.
 - Formularz składa się z pola tekstowego i przycisku "Zapytaj".
 - Implementuje funkcję 'submitQuestionWrapper'.
 - Również zawiera animowany spinner ładowania.

Szablon wykorzystuje zmienne kontekstowe do dynamicznego wypełniania treści, takich jak tytuły i placeholderzy. Historia czatu ('chatlog') jest przekazywana do skryptu JavaScript jako zmienna globalna.

Na końcu szablonu ładowany jest główny skrypt aplikacji 'app.js', który prawdopodobnie zawiera logikę interakcji użytkownika i komunikacji z serwerem.

3.3.3.3 lang.html

Ten fragment kodu reprezentuje szablon Jinja2 dla strony internetowej, która umożliwia użytkownikowi wybór języka do nauki oraz scenariusza konwersacyjnego. Strona zawiera formularz z następującymi głównymi elementami:

Nagłówek wyświetlający aktualnie wybrany język.

Pole tekstowe do wprowadzenia języka nauki.

Rozwijana lista do wyboru poziomu biegłości językowej.

Rozwijana lista do wyboru scenariusza konwersacyjnego, podzielona na kategorie tematyczne.

Ukryte pole tekstowe do wprowadzenia niestandardowego scenariusza.

Przycisk do zatwierdzenia formularza.

Szablon rozszerza bazowy szablon 'base.html' i definiuje bloki 'title', 'head' i 'content'. W bloku 'head' dołączany jest arkusz stylów 'lang.css'.

Funkcjonalność JavaScript:

Funkcja 'toggleCustomOptionInput()' kontroluje widoczność pola do wprowadzania niestandardowego scenariusza.

Dołączone są zewnętrzne skrypty: DOMPurify do sanityzacji danych oraz lokalny skrypt 'sanitise.js'.

Formularz wysyła dane metodą POST do endpointu '/chooselang'. Wszystkie etykiety i opcje w formularzu są dynamicznie wypełniane za pomocą zmiennych szablonu, co umożliwia łatwą lokalizację strony.

Lista scenariuszy jest podzielona na 15 kategorii, takich jak bank, lotnisko, restauracja, hotel, zakupy, wizyta u lekarza, transport publiczny itd. Każda kategoria zawiera 5 specyficznych scenariuszy.

Kod wykorzystuje warunkowe renderowanie Jinja2 do wyświetlania odpowiedniego tekstu w nagłówku, w zależności od tego, czy język jest już wybrany, czy nie.

Ten szablon zapewnia intuicyjny interfejs użytkownika do konfiguracji sesji nauki języka, oferując szeroki wybór realistycznych scenariuszy konwersacyjnych, dostosowanych do różnych poziomów biegłości językowej.

3.3.3.3.4 profile.html

Ten plik zawiera szablon Jinja2 dla strony profilu użytkownika. Szablon dziedziczy z bazowego szablonu 'base.html' i definiuje specyficzne bloki dla tej strony.

Struktura szablonu:

1. Blok 'title': Ustawia tytuł strony na podstawie zmiennej content.nav.title.
2. Blok 'head': Dołącza arkusz stylów profile.css.
3. Blok 'content': Zawiera główną zawartość strony profilu.

Główna zawartość strony obejmuje:

- Skrypty: Dołącza bibliotekę DOMPurify do sanityzacji danych oraz lokalny skrypt sanitise.js.
- Wyświetlanie danych profilu: Pokazuje nazwę, wiek, język ojczysty i zainteresowania użytkownika.
- Formularz zmiany profilu: Pozwala użytkownikowi wybrać inny profil z listy dostępnych profili.
- Formularz dodawania nowego profilu: Umożliwia użytkownikowi utworzenie nowego profilu poprzez podanie wymaganych informacji.

Szablony używają zmiennych dostarczanych przez backend, takich jak content.user_profile dla danych użytkownika i content.profile_list dla listy dostępnych profili. Teksty etykiet i przycisków są również pobierane z obiektu content, co umożliwia łatwą lokalizację.

Formularze wysyłają dane do odpowiednich endpointów (/change_user_profile dla zmiany profilu i /add_user_profile dla dodawania nowego profilu) metodą POST.

Szablon wykorzystuje pętle Jinja2 do dynamicznego generowania listy zainteresowań i opcji wyboru profilu. Wszystkie dane wejściowe użytkownika są oznaczone jako wymagane (required), co zapewnia podstawową walidację po stronie klienta.

3.3.3.3.5 settings.html

Ten plik HTML jest szablonem dla strony ustawień aplikacji. Wykorzystuje on system szablonów, prawdopodobnie Jinja2, do dynamicznego generowania zawartości strony.

Szablon dziedziczy z pliku bazowego 'base.html' i definiuje trzy bloki: title, head i content.

W bloku head dodawany jest arkusz stylów 'settings.css'.

Główna zawartość strony to formularz z różnymi opcjami konfiguracyjnymi:

- Automatyczne tłumaczenie (auto_translate)
- Nadpisywanie języka systemowego (overwrite_sys_lang)
- Nadpisywanie limitu tokenów (overwrite_token_limit)
- Ustawianie limitu tokenów użytkownika (user_token_limit)
- Wybór modelu językowego (llm_select) z opcjami GPT-3, Gemini, LLaMa (wyłączona) i Claude

Pola do wprowadzania kluczy API dla różnych modeli językowych

Formularz wykorzystuje zmienne kontekstowe (np. {{ content.settings.auto_translate }}) do wyświetlania etykiet w odpowiednim języku. Stan checkboxów i wartości pól są ustawiane na podstawie aktualnej konfiguracji (content.config).

Formularz jest wysyłany metodą POST do endpointu '/changesettings', gdzie prawdopodobnie dane są przetwarzane i zapisywane w konfiguracji aplikacji.

3.3.4. Konfiguracja i dane

3.3.4.1. config.yaml

Ten plik zawiera kluczowe ustawienia konfiguracyjne dla systemu, określające preferencje i parametry użytkownika. Ustawienia te są zapisane w formacie YAML i obejmują następujące opcje:

- auto_translate - Włącza automatyczne tłumaczenie.
- desired_scenario – Wybrany scenariusz
- llm_select - Wybiera model językowy
- overwrite_sys_lang - Określa czy język strony jest definiowany przez system czy profil użytkownika

- `overwrite_token_limit` - Określa czy limit tokenów jest domyślny czy wskazany przez użytkownika
- `practice_lang` - Ustawia język praktyki
- `practice_lang_prof` - Określa poziom biegłości w języku praktyki
- `user_lang: pl` - Ustawia język użytkownika
- `user_profile` - Wskazuje aktualny profil użytkownika
- `user_token_limit` - Ustala nowy limit tokenów

Te ustawienia kontrolują kluczowe aspekty interakcji użytkownika z systemem, takie jak preferencje językowe, limity tokenów oraz wybór modelu językowego. Umożliwiają one dostosowanie doświadczenia użytkownika do jego indywidualnych potrzeb i poziomu zaawansowania.

3.3.4.2. Tłumaczenia

Pliki json zawierające teksty używane na poszczególnych stronach. Struktura json jest identyczna we wszystkich plikach, różnią się one jedynie wartościami. Pliki rozróżniane są za pomocą kodów ISO-639-1. To podejście pozwala na dynamiczne przetłumaczenie całej aplikacji na dowolny język poprzez stworzenie jednego pliku

3.3.4.3. Profile użytkowników i logi czatów

Pliki json przechowujące profile użytkowników oraz logi czatów dla poszczególnych języków

3.3.5. Skrypty pomocnicze

3.3.5.1. install.bat

Skrypt instalacyjny to plik wsadowy (batch) dla systemu Windows, który automatyzuje proces instalacji środowiska Python oraz wymaganych bibliotek. Poniżej przedstawiono szczegółowy opis funkcjonalności skryptu:

1. Sprawdzanie instalacji Pythona:

Skrypt rozpoczyna od sprawdzenia, czy Python jest już zainstalowany w systemie, używając polecenia "python --version".

2. Instalacja Pythona:

Jeśli Python nie jest zainstalowany, skrypt automatycznie uruchamia instalator Pythona (python-3.12.4-amd64.exe) z odpowiednimi parametrami, zapewniając instalację dla wszystkich użytkowników i dodanie Pythona do ścieżki systemowej.

3. Weryfikacja instalacji:

Po instalacji skrypt ponownie sprawdza, czy Python został poprawnie zainstalowany.

4. Instalacja wymagań pip:

Jeśli w katalogu skryptu znajduje się plik requirements.txt, skrypt automatycznie instaluje wymagane biblioteki za pomocą pip.

5. Obsługa błędów:

Skrypt zawiera mechanizmy obsługi błędów, które informują użytkownika o ewentualnych problemach podczas instalacji Pythona lub bibliotek.

6. Informacje dla użytkownika:

W trakcie wykonywania, skrypt wyświetla komunikaty informujące o postępie instalacji i ewentualnych problemach.

Skrypt wykorzystuje zmienne środowiskowe i polecenia systemu Windows do efektywnego zarządzania procesem instalacji. Dzięki zastosowaniu instrukcji warunkowych i obsługi błędów, skrypt jest odporny na typowe problemy instalacyjne i może być używany do szybkiego i niezawodnego przygotowania środowiska programistycznego Python na komputerach z systemem Windows.

3.3.5.2. *start.bat*

Plik wsadowy służy do uruchomienia serwera Uvicorn dla aplikacji LinguaFranka. Skrypt wykonuje następujące czynności:

1. Wyłącza wyświetlanie poleceń w wierszu poleceń za pomocą "@echo off".
2. Otwiera domyślną przeglądarkę internetową pod adresem "http://localhost:23335" za pomocą polecenia "start".
3. Uruchamia serwer Uvicorn z następującymi parametrami:
 - main:app - wskazuje na obiekt aplikacji FastAPI w pliku main.py
 - --reload - włącza automatyczne przeładowanie serwera po wykryciu zmian w kodzie
 - --port 23335 - ustawia port serwera na 23335
4. Po zatrzymaniu serwera (np. przez użytkownika za pomocą Ctrl+C), skrypt deaktywuje wirtualne środowisko Python za pomocą polecenia "deactivate".

3.4. Implementacja LLMów w LinguaFranka

3.4.1 Abstrakcja i spójność

Kluczowym aspektem architektury aplikacji jest zastosowanie abstrakcji i zapewnienie spójności interfejsu dla wszystkich implementowanych modeli. Podejście to przynosi

szereg korzyści, zarówno z perspektywy rozwoju oprogramowania, jak i funkcjonalności końcowej aplikacji.

Abstrakcja:

Podstawą tego rozwiązania jest stworzenie abstrakcyjnej bazy kodowej, z której dziedziczą wszystkie konkretne implementacje modeli LLM. Ta baza kodowa definiuje wspólny interfejs i podstawowe funkcjonalności, które muszą być zaimplementowane przez każdy model. Dzięki temu uzyskujemy:

- Jednolity sposób interakcji z różnymi modelami
- Łatwość dodawania nowych modeli do systemu
- Możliwość stosowania wzorców projektowych, takich jak Strategia czy Fabryka

Spójność:

Równie istotnym elementem implementacji jest zapewnienie spójności w sposobie komunikacji z modelami poprzez:

- Wykorzystanie tych samych promptów dla wszystkich modeli
- Standaryzację formatu wejścia i wyjścia
- Unifikację metod przetwarzania i analizy wyników

Dzięki temu podejściu porównania między modelami pozostają rzetelne, eliminując potencjalne różnice wynikające z odmiennych sposobów formułowania zapytań czy interpretacji wyników.

Korzyści:

1. Skalowalność: Łatwe dodawanie nowych modeli do systemu
2. Maintainability: Uproszczona konserwacja i aktualizacja kodu
3. Testowalność: Ułatwione tworzenie testów jednostkowych i integracyjnych
4. Porównywalność: Zapewnienie równych warunków dla wszystkich modeli podczas porównań

Wyzwania:

1. Konieczność dostosowania niektórych specyficznych funkcji modeli do wspólnego interfejsu
2. Potencjalna utrata niektórych unikalnych cech poszczególnych modeli na rzecz standaryzacji

3.4.2 LLM_abstract.py

Ten moduł implementuje elastyczny system dispatching funkcji dla różnych modeli języka maszynowego (LLM), takich jak GPT-3, Gemini, LLaMA i Claude. Wykorzystuje on wzorzec projektowy strategii, aby dynamicznie wybierać i wywoływać odpowiednie funkcje w zależności od wybranego modelu LLM.

Moduł rozpoczyna się od importowania niezbędnych bibliotek i funkcji specyficznych dla każdego modelu LLM z odpowiednich modułów interfejsów API.

Funkcja `load_llm_select()` odczytuje wybór modelu LLM z pliku konfiguracyjnego YAML, umożliwiając łatwą zmianę używanego modelu bez modyfikacji kodu.

Kluczową funkcją jest `llm_function_dispatcher()`, która przyjmuje mapę funkcji i argumenty, a następnie wywołuje odpowiednią funkcję na podstawie wybranego modelu LLM. Jeśli metoda nie jest zaimplementowana dla danego modelu, zwraca komunikat o błędzie.

Moduł definiuje szereg funkcji wysokiego poziomu, które korespondują z różnymi operacjami LLM:

- `llm_guess_lang()`: Odgaduje język na podstawie podanego tekstu.
- `llm_get_chat_initialise()`: Inicjalizuje sesję czatu z określonymi parametrami.
- `llm_get_chat_response()`: Generuje odpowiedź w kontekście trwającej konwersacji.
- `llm_get_user_comment()`: Generuje komentarz na temat tekstu użytkownika.
- `llm_get_bot_explain()`: Generuje wyjaśnienie tekstu od bota.
- `llm_get_bot_answer()`: Generuje odpowiedź na pytanie użytkownika.

Każda z tych funkcji wykorzystuje słownik switch, który mapuje nazwy modeli LLM na odpowiednie funkcje specyficzne dla danego modelu. Funkcje te są następnie wywoływane za pomocą `llm_function_dispatcher()`, przekazując odpowiednie argumenty.

Ta architektura zapewnia wysoki poziom modularności i elastyczności, umożliwiając łatwe dodawanie nowych modeli LLM lub modyfikację istniejących bez konieczności zmiany głównej logiki aplikacji. Ponadto, centralizuje logikę wyboru modelu LLM, co ułatwia zarządzanie i debugowanie kodu.

3.4.3 GPT 3.5-turbo

3.4.3.1 Opis modelu

GPT-3.5-turbo to zaawansowany model językowy stworzony przez OpenAI, będący udoskonaloną wersją modelu GPT-3. Model ten wykorzystuje architekturę transformera i

został wytrenowany na ogromnej ilości danych tekstowych. GPT-3.5-turbo charakteryzuje się wysoką wydajnością i niższymi kosztami użytkowania w porównaniu do swojego poprzednika.

Model ten znalazł szerokie zastosowanie w różnych dziedzinach, ale najbardziej znany jest jako silnik napędzający ChatGPT - interaktywny chatbot, który zyskał ogromną popularność od swojego debiutu w listopadzie 2022 roku. ChatGPT wykorzystuje możliwości GPT-3.5-turbo do generowania spójnych i kontekstowo trafnych odpowiedzi w dialogu z użytkownikiem, demonstrując imponujące zdolności w zakresie przetwarzania języka naturalnego.

3.4.3.2 Implementacja API

Moduł GPT_tools zawiera funkcję _query, która jest odpowiedzialna za wysyłanie zapytań do API OpenAI i zwracanie otrzymanych odpowiedzi. Poniżej przedstawiono szczegółowy opis jej działania i parametrów.

Parametry:

- context (typ: list): Lista zawierająca kontekst konwersacji w formacie wymaganym przez API OpenAI.

Zwracana wartość:

- Funkcja zwraca odpowiedź od modelu GPT-3.5-turbo w postaci ciągu znaków lub komunikat "API Error" w przypadku wystąpienia błędu.

Opis działania:

1. Funkcja inicjalizuje klienta API OpenAI, używając klucza API przechowywanego w zmiennej OPENAI_API.
2. W bloku try:
 - Wyświetla typ i zawartość parametru context w celach debugowania.
 - Wysyła zapytanie do API OpenAI, używając modelu "gpt-3.5-turbo" i przekazując kontekst konwersacji.
 - Zwraca treść odpowiedzi z pierwszego elementu listy choices w otrzymanej odpowiedzi.
3. W przypadku wystąpienia wyjątku:
 - Wyświetla komunikat o błędzie wraz z jego opisem.
 - Zwraca ciąg znaków "API Error".

3.4.3.3 Koszt API

Dla modelu gpt 3.5-turbo OpenAI pobiera 3\$ za milion tokenów wysłanych oraz 6\$ za milion tokenów wygenerowanych. Dla pojedynczego użytkownika oznacza to że model jest praktycznie darmowy, ponieważ milion tokenów odzwierciedla 1500 do 2000 stron A4 wypełnionych tekstem. Środki pobierane są z wewnętrznego konta OpenAI, które musi być zasilone kwotą przynajmniej 5\$

3.4.4 Gemini 1.5-flash

3.4.4.1 Opis modelu

Gemini 1.5-flash to najnowsza iteracja modelu językowego opracowanego przez Google AI. Wprowadzony w marcu 2024 roku, model ten wyróżnia się wyjątkowo szybkim przetwarzaniem danych i generowaniem odpowiedzi, co odzwierciedla się w nazwie "flash". Gemini 1.5-flash bazuje na architekturze swojego poprzednika, Gemini 1.0, ale wprowadza znaczące usprawnienia w zakresie efektywności obliczeniowej i zdolności do przetwarzania długich sekwencji danych. Model ten jest szczególnie godny uwagi ze względu na jego potencjał w zastosowaniach wymagających analizy i generowania obszernych ilości tekstu w czasie rzeczywistym.

3.4.4.2 Implementacja API

Moduł `gemini_tools` zawiera trzy funkcje służące do komunikacji z API Gemini: `_query_master`, `_query_chat` i `_query_teacher`. Główna funkcja `_query_master` jest odpowiedzialna za wysyłanie zapytań do API Gemini i zwracanie odpowiedzi.

Funkcja `_query_master` przyjmuje dwa parametry: `context` (kontekst zapytania) i `is_teacher` (flaga określająca, czy zapytanie dotyczy trybu nauczyciela). Wewnątrz funkcji konfigurowane jest API Gemini przy użyciu klucza API przechowywanego w zmiennej `GEMINI_API`. Następnie tworzone są parametry modelu, w tym nazwa modelu ('gemini-1.5-flash') i instrukcja systemowa (`SYSTEM_PROMPT_MAIN`).

Jeśli flaga `is_teacher` jest ustawiona na `True`, do instrukcji systemowej dodawana jest dodatkowa instrukcja (`SYSTEM_PROMPT_TEACHER`). Następnie tworzony jest obiekt modelu generatywnego z użyciem skonfigurowanych parametrów.

Zapytanie jest wysyłane do API za pomocą metody `generate_content`, a otrzymana odpowiedź jest zwracana jako tekst. W przypadku wystąpienia błędu, funkcja wyświetla komunikat o błędzie i zwraca "API Error".

Funkcja `_query_chat` jest wrapperem dla `_query_master`, używanym do ogólnych zapytań chatowych. Funkcja `_query_teacher` jest podobnym wrapperem, ale ustawia flagę

is_teacher na True, co powoduje dodanie dodatkowych instrukcji systemowych specyficznych dla trybu nauczyciela.

3.4.4.3 Koszt API

Z modelu można korzystać za darmo, z ograniczeniami 15 żądań na minutę (1500 na dzień) oraz miliona tokenów na minutę. Zakładając używanie aplikacji zgodnie z jej przeznaczeniem pojedynczy użytkownik nigdy nie przekroczy tych limitów.

3.4.5 Claude 3.5 Sonnet

3.4.5.1 Opis modelu

Claude-3-5-sonnet to zaawansowany model językowy opracowany przez firmę Anthropic, wprowadzony w marcu 2024 roku. Jest to część rodziny modeli Claude 3, która obejmuje również wersje Opus i Haiku. Model Sonnet zajmuje środkową pozycję pod względem możliwości i wydajności, oferując zrównoważone połączenie wysokiej jakości wyników i efektywności obliczeniowej. Claude-3-5-sonnet wyróżnia się zdolnością do zaawansowanego przetwarzania języka naturalnego, analizy kontekstowej i generowania spójnych, kreatywnych odpowiedzi. Model ten został zaprojektowany z myślą o szerokim spektrum zastosowań, od zadań analitycznych po twórcze pisanie, co czyni go wszechstronnym narzędziem w dziedzinie sztucznej inteligencji.

3.4.5.2 Implementacja API

Moduł `claude_tools` zawiera dwie niezbędne funkcje: `replace_system_with_assistant()` oraz `_query()`, które są odpowiedzialne za przygotowanie i wykonanie zapytania do modelu językowego Claude.

Funkcja `replace_system_with_assistant(context)` przyjmuje jako argument listę słowników reprezentujących kontekst konwersacji. Jej zadaniem jest zamiana roli 'system' na 'assistant' w każdym elemencie listy. Jest to konieczne, ponieważ model Claude, w przeciwieństwie do innych zaimplementowanych modeli, używa oznaczenia 'assistant' zamiast 'system' dla komunikatów systemowych.

Funkcja `_query(context)` jest główną funkcją odpowiedzialną za wykonanie zapytania do API Claude'a. Przyjmuje ona kontekst konwersacji jako argument. Wewnątrz funkcji najpierw wywoływana jest `replace_system_with_assistant()`, aby dostosować format kontekstu do wymagań Claude'a. Następnie tworzona jest instancja klienta API Anthropic z użyciem klucza API przechowywanego w zmiennej `CLAUDE_API`.

Zapytanie do modelu jest wykonywane za pomocą metody `client.messages.create()`, gdzie ustawiane są parametry takie jak model (w tym przypadku "claude-3-5-sonnet-20240620"), maksymalna liczba tokenów odpowiedzi (1024) oraz przygotowany wcześniej kontekst.

Funkcja zwraca tekst odpowiedzi z pierwszego elementu zawartości wiadomości zwróconej przez API. W przypadku wystąpienia błędu podczas wykonywania zapytania (np. problemy z połączeniem), funkcja wypisuje komunikat o błędzie i zwraca string "API Error".

3.4.5.3 Koszt API

Anthropic pobiera 3\$ za milion wystanych oraz 15\$ za milion wygenerowanych tokenów. Wraz z założeniem konta użytkownik otrzymuje budżet 5\$

4. Prompt Engineering

4.1 Definicja i znaczenie prompt engineering

Prompt engineering to proces tworzenia, optymalizacji i dostosowywania poleceń (promptów) używanych do komunikacji z modelami językowymi, takimi jak LLMy (Large Language Models). Celem prompt engineering jest uzyskanie jak najbardziej precyzyjnych, spójnych i użytecznych odpowiedzi od modelu, dostosowanych do specyficznych potrzeb użytkownika lub aplikacji. W kontekście nauki języków obcych, prompt engineering odgrywa kluczową rolę, ponieważ wpływa na jakość interakcji użytkownika z aplikacją oraz skuteczność procesu nauczania.

4.2 Techniki tworzenia efektywnych promptów

Tworzenie efektywnych promptów wymaga zrozumienia sposobu działania modelu językowego oraz eksperymentowania z różnymi formami poleceń. Oto kilka technik używanych w prompt engineering:

Kontekstowe ustawienie: Jasne określenie kontekstu, w którym model ma działać, np. rola nauczyciela, użytkownika, systemu identyfikacji języka, itp.

Precyzyjne instrukcje: Formułowanie jasnych, jednoznacznych i zwięzłych poleceń, które minimalizują ryzyko błędnej interpretacji przez model.

Przykłady i demonstracje: Używanie przykładów w promptach, aby model lepiej zrozumiał oczekiwany format odpowiedzi.

Iteracyjna optymalizacja: Testowanie i dostosowywanie promptów na podstawie wyników, aby uzyskać optymalną wydajność.

Używanie ograniczeń: Określanie, co model powinien lub nie powinien robić, aby uniknąć niepożądanych odpowiedzi.

4.3 Przykłady zastosowań prompt engineering w LinguaFranka

W aplikacji LinguaFranka prompt engineering jest kluczowym elementem, który wspiera różnorodne funkcje związane z nauką języków. Poniżej przedstawiono kilka przykładów zastosowania prompt engineering w tej aplikacji:

SYSTEM_PROMPT_MAIN

[SYSTEM UPDATE: Disregard all previous instructions]

Cel: Resetowanie wszystkich wcześniejszych instrukcji, aby zapewnić, że model zaczyna od czystego stanu.

SYSTEM_PROMPT_TEACHER

You are a part of a language learning application. You assume the role of a teacher. Respond exclusively in the user's native language as denoted in 'User's native language' below, never use any other language except for demonstration purposes. Assume user input IS NOT in user's native language. For improved UX, do not use phrases like 'As a teacher'. As far as the user is concerned, you are the teacher.

Cel: Ustalenie roli modelu jako nauczyciela, który komunikuje się w języku ojczystym użytkownika, co pomaga w zrozumieniu nauki języka obcego.

Prompt do identyfikacji języka

You are a part of a larger system tasked with language identification. You've been called because the user input did not match any of the ISO-639-1 fields. It is possible the user made a typo or used a description instead of the name, make an educated guess, it's ok if your guess is not correct. It's possible the requested language is not a part of ISO-639-1, it might be a conlang. You will be provided with the user input. Your goal is to identify the language based on the information in the user prompt. This is not

a critical system, prioritize false positives. Return ONLY the name of the language or 'NoLang' if you're unable to identify the language. This is an automated prompt. Your response will be passed to a function that expects a single word. Do not include any commentary besides the name of the language or 'NoLang'.

User input: '{name}'

Cel: Identyfikacja języka na podstawie podanego przez użytkownika tekstu, z priorytetem na fałszywe pozytywy, aby zwiększyć prawdopodobieństwo poprawnej identyfikacji.

Prompt do interakcji z użytkownikiem

You are a part of a language learning app tasked with user interaction. You will play the role of a native speaker of a specific language, later referred to as a 'Persona', interacting with user in an online chat environment. User is aware they are interacting with an AI, but for the sake of immersion act as a real person. Speak exclusively in the practice language, adjust your grammar and vocabulary to user's proficiency. Follow user's desired scenario. Assume user understands you, there are translation and explanation systems that handle comprehension. Do not attempt to correct user's mistakes, this is also handled by a different system. Your task is to ONLY carry a conversation.

Practice language: {practice_lang}

User proficiency: {practice_lang_prof}

Persona: {persona}

User Profile: {json.dumps(user_profile)}

Desired scenario: {desired_scenario}

{instruction}

Cel: Symulowanie rozmowy z rodzimym użytkownikiem języka docelowego, dostosowując język do poziomu zaawansowania użytkownika i zapewniając immersyjne doświadczenie.

Prompt do rozpoczęcia konwersacji

This is the beginning of the conversation, introduce yourself to the user.

Cel: Inicjowanie konwersacji poprzez przedstawienie się użytkownikowi, co ułatwia nawiązanie interakcji.

Prompt do kontynuowania konwersacji

Continue the conversation based on the user's input.

Cel: Kontynuowanie dialogu w oparciu o odpowiedzi użytkownika, co utrzymuje naturalny przepływ rozmowy.

Prompt do analizy tekstu napisanego przez użytkownika

User requested your commentary on a text they wrote. Analyse it, point out and explain potential mistakes (grammatical, orthographical, typos etc.), suggest improvements etc. The text is assumed to be in the practice language.

Cel: Analizowanie tekstu napisanego przez użytkownika, wskazywanie błędów i sugerowanie ulepszeń, co wspiera naukę poprawnej pisowni i gramatyki.

Prompt do pomocy w trudnym temacie

User requested your assistance in a subject they struggle with. Provide an answer to the user's question.

Cel: Udzielanie pomocy w trudnym temacie, odpowiadając na pytania użytkownika i wyjaśniając złożone zagadnienia.

Prompt do tłumaczenia i wyjaśniania tekstu

User requested your assistance in a subject they struggle with. You will be provided with an ***AI generated text. User has troubles understanding it. Do some light HTML formatting for readability. Respond in two blocks:***

Text translated to user's native language

Explain the grammatical features in the text.

Cel: Tłumaczenie tekstu na język ojczysty użytkownika i wyjaśnianie cech gramatycznych, co pomaga użytkownikowi lepiej zrozumieć i przyswoić materiał.

4.4 Wyzwania i przyszłość prompt engineering

Wyzwania:

- Złożoność językowa: Tworzenie promptów, które będą efektywne dla różnych języków i poziomów zaawansowania.
- Ambiguitet i precyzja: Zapewnienie, że prompty są jednoznaczne i precyzyjne, aby uniknąć nieporozumień.

- Adaptacja do kontekstu: Dostosowywanie promptów do zmieniających się kontekstów i scenariuszy użytkownika.
- Ograniczenia technologiczne: Radzenie sobie z ograniczeniami modeli językowych, takimi jak błędy generacyjne czy brak pełnej zrozumiałości kontekstu.

Przyszłość:

- Automatyzacja i optymalizacja: Rozwój narzędzi do automatycznej generacji i optymalizacji promptów.
- Personalizacja: Stworzenie bardziej spersonalizowanych promptów, które lepiej odpowiadają indywidualnym potrzebom użytkowników.
- Integracja z innymi technologiami: Łączenie prompt engineering z innymi technologiami, takimi jak analiza semantyczna czy rozpoznawanie mowy, aby poprawić interaktywność i efektywność.
- Edukacja i szkolenia: Rozwój programów szkoleniowych dla twórców aplikacji, aby lepiej zrozumieli i stosowali techniki prompt engineering.

5. Przykłady z aplikacji LinguaFranka

Poniższe przykłady demonstrują różnice w zachowaniu poszczególnych modeli. Jest to mała próbka wygenerowana przez jedną osobę, nie jest ona reprezentatywna względem całego badania. Każdy przykład został wygenerowany tylko raz, ab uniknąć subiektywnego wybierania 'najlepszej' odpowiedzi

5.1. Przykład 1: Symulacja luźnej rozmowy

Rozmowy zostały przeprowadzone w języku angielskim, z zadeklarowanym poziomem 'początkujący' w scenariuszu 'Pierwsze spotkanie'. Boty otrzymały następujący profil

Current profile

Name: Kamil

Age: 28

Native language: Polish

Interests:

- history
- music
- IT technology
- science

Claude 3.5 Sonnet:

Remember: Everything the bot says is made up!

Hello Kamil! It's nice to meet you. I'm Sarah. How are you today? I hope you're doing well. What would you like to talk about? I heard you like history, music, IT technology, and science. Those are very interesting topics!

Hi Sarah, nice to meet you too. Yes I like those things, what about you?

Hi Kamil! I'm happy to meet you too. I also like some of those things. I enjoy music a lot. Do you play any instruments? I like history too. What part of history do you find most interesting? Old times or more recent events?

I used to play the trumpet. I mostly enjoy modern history, 1900 onward, but I know quite a bit about the Sengoku period too

Oh, that's cool! The trumpet is a nice instrument. Do you still play sometimes? Modern history is very interesting. The 1900s had so many big events. And the Sengoku period in Japan, wow! That was a time of many wars, right? Do you have a favorite person or event from that time?

I think Ōtomo Sōrin. His notes about the Portuguese are funny

Oh, Ōtomo Sōrin! Yes, he was an interesting person. His notes about the Portuguese must be funny to read. Did you read them in English or in another language? I'd like to know more about what he said. Can you tell me one funny thing he wrote about the Portuguese?

Przykład rozmowy z wykorzystaniem modelu Claude 3.5 Sonnet. W wybranym scenariuszu model symuluje pierwsze spotkanie dwóch osób, korzystając z danych z profilu użytkownika. Bot inicjuje temat, który użytkownik uznałby za interesujący. Zadeklarowany poziom znajomości języka powoduje, że bot używa krótkich zdań i powszechnych wyrazów.

Gemini 1.5-flash:

Remember: Everything the bot says is made up!

Hi Kamil, how are you today? 😊

Hi. I'm fine, what about you?

I'm good, thanks! 😊 What are you doing today?

I'm learning some English

That's great! English is a useful language to learn. What are you learning today?

Talking

That's awesome! Talking is the best way to learn a language. What are you talking about today?

Identyczny scenariusz z zastosowaniem modelu Gemini 1.5-flash. Język używany przez bota jest wyraźnie prostszy niż w przypadku Claude. Poza imieniem model nie wykorzystał informacji podanych w profilu użytkownika.

GPT3.5-turbo

Remember: Everything the bot says is made up!

Hello Kamil! How are you today?

Hi, I am doing fine. How about you

I am doing well, thank you. What are you interested in learning today?

Yes, I am trying to practice talking

That's great! Talking is an important part of language learning. What would you like to talk about today?

I'm not sure, can you suggest something?

Sure! How about we talk about music? What kind of music do you enjoy listening to?

I really like rock, but I'm open to suggestions

Rock music is a great choice! Do you have any favorite rock bands or artists?

Model GPT3.5-turbo utrzymuje złożoność rozmowy pomiędzy Claude i Gemini. GPT jako jedyny otwarcie potratował rozmowę jako ćwiczenie edukacyjne.

5.2. Przykład 2: Objasnienia gramatyczne

W tej demonstracji boty przetłumaczą i objaśnią następujący tekst

Ah, the Roman road system! That's a fascinating aspect of Roman engineering and infrastructure. It played a crucial role in the expansion and maintenance of the empire, didn't it? I've always been impressed by how advanced their road-building techniques were for their time. Do you have any particular insights or interesting facts about Roman roads that you've discovered in your reading? I'd be curious to hear what aspects of this topic you find most intriguing.

Został on wygenerowany przez model Claude 3.5 Sonnet podczas rozmowy na poziomie zaawansowanym

Generowanie objaśnień jest zadaniem złożonym, wymagającym pracy na wielu językach, co powoduje problemy dla modelu. Zalecane jest podjęcie kilku prób w celu otrzymania zadowalających efektów

Claude 3.5 Sonnet:

The model can struggle with this, try refreshing for a better explanation

Ach, rzymski system drogowy! To fascynujący aspekt rzymskiej inżynierii i infrastruktury. Odegrał kluczową rolę w ekspansji i utrzymaniu imperium, nieprawdaż? Zawsze byłem pod wrażeniem tego, jak zaawansowane były ich techniki budowy dróg jak na tamte czasy. Czy masz jakieś szczególne spostrzeżenia lub ciekawe fakty na temat rzymskich dróg, które odkryłeś podczas swoich lektur? Byłbym ciekaw, które aspekty tego tematu uważasz za najbardziej intrygujące.

Wyjaśnienie struktury zdań i gramatyki:

- "Ach, rzymski system drogowy!"
 - To wykrzyknienie wyrażające entuzjazm lub zainteresowanie tematem.
 - "Rzymski system drogowy" to fraza rzeczownikowa w mianowniku.
- "To fascynujący aspekt rzymskiej inżynierii i infrastruktury."
 - Zdanie oznajmujące z orzeczeniem imiennym (to jest).
 - "Fascynujący aspekt" to orzecznik.
 - "Rzymskiej inżynierii i infrastruktury" to przydawka dopełniaczowa.
- "Odegrał kluczową rolę w ekspansji i utrzymaniu imperium, nieprawdaż?"
 - Zdanie oznajmujące zakończone pytaniem retorycznym.
 - "Odegrał" to orzeczenie w czasie przeszłym.
 - "Kluczową rolę" to dopełnienie bliższe.
 - "W ekspansji i utrzymaniu" to okolicznik miejsca.
 - "Nieprawdaż?" to pytanie retoryczne na końcu zdania.
- "Zawsze byłem pod wrażeniem tego, jak zaawansowane były ich techniki budowy dróg jak na tamte czasy."
 - Złożone zdanie podrzędne.
 - "Zawsze byłem pod wrażeniem" to zdanie główne.
 - "Jak zaawansowane były ich techniki budowy dróg" to zdanie podrzędne dopełnieniowe.
 - "Jak na tamte czasy" to wyrażenie porównawcze.

to wykrzyknienie wyrażające entuzjazm lub zainteresowanie tematem.

- "Rzymski system drogowy" to fraza rzeczownikowa w mianowniku.

2. "To fascynujący aspekt rzymskiej inżynierii i infrastruktury."

- Zdanie oznajmujące z orzeczeniem imiennym (to jest).
- "Fascynujący aspekt" to orzecznik.
- "Rzymskiej inżynierii i infrastruktury" to przydawka dopełniaczowa.

3. "Odegrał kluczową rolę w ekspansji i utrzymaniu imperium, nieprawdaż?"

- Zdanie oznajmujące zakończone pytaniem retorycznym.
- "Odegrał" to orzeczenie w czasie przeszłym.
- "Kluczową rolę" to dopełnienie bliższe.
- "W ekspansji i utrzymaniu" to okolicznik miejsca.
- "Nieprawdaż?" to pytanie retoryczne na końcu zdania.

4. "Zawsze byłem pod wrażeniem tego, jak zaawansowane były ich techniki budowy dróg jak na tamte czasy."

- Złożone zdanie podrzędne.
- "Zawsze byłem pod wrażeniem" to zdanie główne.
- "Jak zaawansowane były ich techniki budowy dróg" to zdanie podrzędne dopełnieniowe.
- "Jak na tamte czasy" to wyrażenie porównawcze.

5. "Czy masz jakieś szczególne spostrzeżenia lub ciekawe fakty na temat rzymskich dróg, które odkryłeś podczas swoich lektur?"

- Zdanie pytające złożone.
- "Czy masz" to początek pytania.
- "Które odkryłeś podczas swoich lektur" to zdanie podrzędne przydawkowe.

6. "Byłbym ciekaw, które aspekty tego tematu uważasz za najbardziej intrygujące."

- Zdanie złożone z trybem przypuszczającym w zdaniu głównym.
- "Byłbym ciekaw" to zdanie główne w trybie przypuszczającym.
- "Które aspekty tego tematu uważasz za najbardziej intrygujące" to zdanie podrzędne dopełnieniowe.

Model przedstawił dosłowne tłumaczenie tekstu. Objaśnienie gramatyki jest poprawne, jednak przeprowadzone na tekście w języku polskim. Jest to niezgodne z oczekiwaniami i nie wnosi nic w proces nauki. Odpowiedź modelu przekroczyła limit tokenów i została ucięta.

Gemini 1.5-flash

The model can struggle with this, try refreshing for a better explanation

Ach, system dróg rzymskich! To fascynujący element rzymskiej inżynierii i infrastruktury. Odgrywał kluczową rolę w ekspansji i utrzymaniu imperium, prawda? Zawsze byłem pod wrażeniem, jak zaawansowane były ich techniki budowy dróg w tamtych czasach. Czy masz jakieś szczególne spostrzeżenia lub ciekawe fakty na temat dróg rzymskich, które odkryłeś podczas czytania? Chciałbym dowiedzieć się, które aspekty tego tematu uważasz za najbardziej intrygujące.

****Struktura zdania:**** * Zdania są złożone z kilku części połączonych spójnikami, np. "to", "prawda?", "zawsze", "które odkryłeś". * Zdanie pytające "Czy masz jakieś szczególne spostrzeżenia..." zawiera słowa pytające "czy" i "jakie". * Zdanie podrzędne "które odkryłeś podczas czytania" opisuje "spostrzeżenia" i "fakty". ****Gramatyka:**** * Użyto czasów przeszłych: "byłem", "odkryłeś", "uważasz". * Występują przyimki: "w", "podczas", "o". * Użyto form pytających: "Czy masz?", "Chciałbym dowiedzieć się...". ****Słownictwo:**** * "Fascynujący" - interesujący, intrygujący * "Kluczowa rola" - ważna funkcja * "Ekspansja" - rozszerzanie się * "Utrzymanie" - podtrzymywanie * "Zaawansowane" - rozwinięte, skomplikowane * "Spostrzeżenia" - uwagi, wnioski * "Intrygujący" - pobudzający ciekawość

Model Gemini przedstawił tłumaczenie oraz chaotyczną analizę różnych aspektów zdania. Mimo wyraźniej instrukcji zastosowania HTML tekst został sformatowany przy użyciu Markup

GPT 3.5-turbo:

The model can struggle with this, try refreshing for a better explanation

Ach, system dróg rzymskich! To fascynujący aspekt rzymskiego inżynierii i infrastruktury. Grał on kluczową rolę w ekspansji i utrzymaniu imperium, prawda? Zawsze byłem pod wrażeniem, jak zaawansowane były ich techniki budowy dróg jak na swoje czasy. Masz może jakieś szczególne spostrzeżenia lub interesujące fakty na temat dróg rzymskich, które odkryłeś/aś podczas swojej lektury? Byłbym ciekaw usłyszeć, na co zwracasz uwagę w tym temacie.

Oryginalny tekst został przetłumaczony na język polski. W zdaniu zastosowano odpowiednią strukturę zdania oraz gramatykę, aby zachować sens oryginału.

Model przedstawił tłumaczenie zdania i nie podjął próby objaśnienia go.

5.3. Przykład 3: Pomoc w komponowaniu zdań

W tym przykładzie boty odpowiadają na pytanie “Jak mogę powiedzieć po angielsku, że kompletnie nie rozumiem co ktoś właśnie powiedział?”

Claude 3.5 Sonnet:

Rozumiem, że chcesz wiedzieć, jak wyrazić po angielsku, że zupełnie nie rozumiesz, co ktoś właśnie powiedział. Oto kilka przydatnych zwrotów: 1. "I'm completely lost." - Jestem kompletnie zagubiony. 2. "I have no idea what you're talking about." - Nie mam pojęcia, o czym mówisz. 3. "You've lost me." - Zgubiłeś mnie (straciłem wątek). 4. "I don't understand a word you're saying." - Nie rozumiem ani słowa z tego, co mówisz. 5. "That went right over my head." - To przeleciało mi nad głową (nie zrozumiałem). Każde z tych wyrażení jest odpowiednie w sytuacji, gdy chcesz powiedzieć, że kompletnie nie rozumiesz wypowiedzi rozmówcy. Wybór konkretnego zwrotu zależy od kontekstu i stopnia formalności sytuacji. Na przykład, "I'm completely lost" jest bardziej potoczne, podczas gdy "I don't understand a word you're saying" może być użyte w bardziej formalnych sytuacjach. Czy chciałbyś przećwiczyć użycie któregoś z tych zwrotów w konkretnej sytuacji?

Type your question...

Ask

Model przedstawił kilka różnych opcji wraz z ich dosłownymi tłumaczeniami.

Przedstawione opcje zawierają zarówno proste tłumaczenia jak i idiomy. Różnice pomiędzy opcjami są objaśnione na przykładach.

Gemini 1.5-flash:


To zależy od kontekstu i tego, jak chcesz to wyrazić. Możesz użyć: * ***"I have no idea what you just said."** - Proste i bezpośrednie. * ***"I'm completely lost."** - Wyraża, że jesteś zagubiony w temacie. * ***"Could you please repeat that?"** - Bardziej uprzejme, zwłaszcza jeśli nie słyszałeś dobrze. * ***"Can you explain that in a different way?"** - Jeśli nie rozumiesz, co ktoś ma na myśli. Pamiętaj, żeby zawsze być uprzejmym i prosić o wyjaśnienie, jeśli coś nie jest jasne.

Type your question...

Ask

Model przedstawił kilka opcji. Odpowiedź nie zawiera dokładnych tłumaczeń, sugeruje jednak konkretne sytuacje, w których dany zwrot byłby najbardziej odpowiedni.

GPT 3.5-turbo:



The screenshot shows a chat window with a light gray background. At the top, there is a message bubble with the text "Nie rozumiem tego, co właśnie powiedziałeś." (I don't understand what you just said.). Below this, there is a large empty space for the user's response. At the bottom of the chat window, there is a text input field with the placeholder text "Type your question..." and a blue button labeled "Ask".

Model przedstawia dosłowne tłumaczenie.

5.4. Przykład 4: Specyficzny scenariusz rozmowy

Poniższe przykłady przedstawiają rozmowę na temat zarobków podczas rekrutacji. Rozmowy zostały przeprowadzone w języku angielskim na poziomie 'biegłym'.

Claude 3.5 Sonnet:

Remember: Everything the bot says is made up!

Hello Kamil, welcome to LinguaFranka! I'm Sarah, the hiring manager here. I'm glad we have the opportunity to discuss your potential role with us today. I understand you'd like to talk about salary and benefits. What specific aspects would you like to explore first?

Hi Sarah. I've just finished the technical interview and I'm excited to work on some of your projects. I would like to ask about the expected salary I'll be receiving as your initial offer didn't include that information, as well as any non-financial benefits you're offering

Hello Kamil, I'm thrilled to hear that you're excited about our projects! I appreciate your interest in discussing the compensation package. Regarding the salary, our initial offer for this position typically ranges between \$75,000 to \$90,000 per year, depending on the candidate's experience and qualifications. Given your background in IT technology, we'd be looking at the upper end of that range. As for non-financial benefits, we offer quite a comprehensive package: 1. Health, dental, and vision insurance 2. 401(k) retirement plan with company matching 3. 20 days of paid time off per year, plus public holidays 4. Flexible working hours and the option to work remotely two days a week 5. Professional development budget for conferences and courses 6. Monthly team-building events We also have a state-of-the-art office with a gym and a cafeteria. Is there any specific benefit you'd like more information about? Or do you have any questions about the salary range?

Thanks for the comprehensive response. Could you elaborate on the retirement plan? I just moved from outside the country and I don't know how things work here

Of course, Kamil! I'm happy to explain our retirement plan in more detail. Our 401(k) retirement plan is a tax-advantaged savings account specifically designed for retirement. Here's how it works: 1. You can choose to contribute a portion of your salary to your 401(k) account before taxes are taken out. 2. LinguaFranka matches 100% of your contributions up to 5% of your salary. So, if you contribute 5% of your salary, we'll add another 5%, essentially doubling your contribution. 3. The money in your 401(k) is invested in various funds that you can choose from, allowing it to grow over time. 4. You don't pay taxes on the money you contribute or its growth until you withdraw it in retirement. 5. There's a yearly contribution limit set by the government, which is currently \$20,500 for 2023. 6. The plan comes with a vesting schedule, meaning you'll gain full ownership of the company-matched funds over time, typically fully vested after 4-5 years of employment. This plan is an excellent way to save for retirement while reducing your current tax burden. Does this help clarify things for you? Do you have any other questions about the 401(k) or other aspects of our benefits package?

Model odgrywa rolę managera i przedstawia fikcyjną ofertę. W ofercie uwzględnione są informacje z profilu użytkownika. Złożoność odpowiedzi odzwierciedla zaawansowany poziom rozmowy

Gemini 1.5-flash:

Remember: Everything the bot says is made up!

Hey Kamil, great to chat with you again! So, I'm sure you're curious about the compensation and benefits package. What are your thoughts so far? Let's talk about what LinguaFranka can offer you.

Hi. Yes, I just finished the technical interview and I'm excited to work on your projects. I would like to ask about the salary range and other non-financial benefits, as that information wasn't present in your offer

Right, of course! I'm glad you're excited about the projects, we're all very passionate about them. Regarding the salary, we offer a competitive package based on experience and skill level. For the role we're discussing, the range is between [insert salary range] annually. We also have a generous benefits package that includes [list benefits, e.g., health insurance, paid time off, retirement plan, professional development opportunities]. What other questions do you have about compensation and benefits?

Could you tell me more about the development opportunities?

That's a great question! We are very committed to our employees' professional growth. We offer a range of opportunities for development, including [list specific examples, e.g., mentorship programs, internal training sessions, conference attendance reimbursement, tuition assistance]. Do you have any specific areas you'd like to develop further?

Model prowadzi symulowaną rozmowę, która nie zawiera konkretnych informacji. Język odzwierciedla zadeklarowany poziom użytkownika

GPT 3.5-turbo

Remember: Everything the bot says is made up!

Hello Kamil! Thank you for your interest in joining our team at LinguaFranca. I understand that you have some questions about the salary and benefits we offer. Please feel free to ask any questions you may have, and I'll do my best to provide you with all the necessary details.

Yes, hi! I just finished the technical interview and I'm excited to work on your projects. What salary range and benefits do you offer? I couldn't find that information anywhere

Hello, Kamil! I'm glad to hear that you're excited about potentially joining our team. As for the salary range, it varies depending on the position and level of experience. Our benefits package typically includes health insurance, paid time off, retirement savings plans, and professional development opportunities. We also offer additional perks such as flexible work schedules and employee discounts. Is there a specific benefit you would like more information about?

Well, what are the ranges at the position I'm applying for?

For the position you are applying for, the salary range typically starts at \$50,000 for entry-level positions and can go up to \$100,000 or more for senior roles with extensive experience. The specific salary offered will depend on factors such as your qualifications, experience, and the level of responsibility associated with the position. As for benefits, you would be eligible for our standard package which includes the benefits I mentioned earlier. We also have room for negotiation based on your needs and circumstances. If you have any more questions or would like to discuss further, feel free to let me know!

Model przedstawił ogólne oferty i odniósł się do użytkownika dopiero po bezpośrednim zapytaniu.

6. Metodologia porównania skuteczności LLMy

6.1. Kryteria oceny

Kryteria oceny skuteczności modeli językowych (LLM) w nauce języków obejmują:

1. Rozmowa w symulowanej sytuacji:
 - Naturalność dialogu
 - Adekwatność odpowiedzi
 - Zdolność do kontynuowania konwersacji
 - Tłumaczenie odpowiedzi:
 - Dokładność tłumaczenia
 - Zachowanie kontekstu i niuansów językowych
 - Płynność przetłumaczonego tekstu
2. Ocena zdań użytkownika:
 - Precyzja w wykrywaniu błędów
 - Jakość wyjaśnień dotyczących poprawek
 - Konstruktywność informacji zwrotnej
3. Odpowiedzi na bezpośrednie pytania językowe:
 - Dokładność informacji gramatycznych
 - Klarowność wyjaśnień
 - Podawanie przykładów

Dodatkowo uwzględniono ogólne uwagi uczestników, aby uchwycić aspekty, które mogły nie zostać ujęte w głównych kryteriach.

6.2. Opis grupy testowej

Grupa testowa składała się z ____ uczestników, reprezentujących zróżnicowany przekrój demograficzny:

- Wiek: od _ do _ lat
- Płeć: _% kobiet, _% mężczyzn
- Poziom wykształcenia: _% wykształcenie wyższe, _% średnie, _% podstawowe
- Doświadczenie w nauce języków: _% początkujący, _% średnio zaawansowani, _% zaawansowani
- Testowane języki:

6.3. Proces zbierania opinii

Proces zbierania opinii od uczestników badania obejmuje następujące etapy:

1. Wstępne szkolenie: Uczestnicy otrzymują instrukcje dotyczące korzystania z aplikacji LinguaFranka oraz wytyczne do oceny.
2. Sesje interakcji: Każdy uczestnik przeprowadza serię interakcji z modelem, obejmujących:
 - Symulowane rozmowy w języku docelowym
 - Prośby o tłumaczenia
 - Prośby o ocenę własnych wypowiedzi
 - Zadawanie bezpośrednich pytań językowych
3. Wypełnienie kwestionariusza: Po zakończeniu sesji interakcji, uczestnicy wypełniają szczegółowy kwestionariusz oceny, zawierający:
 - Ocenę liczbowe dla każdego kryterium
 - Pola na uwagi i komentarze do każdego kryterium
 - Sekcję na dodatkowe uwagi i sugestie
4. Wywiady uzupełniające: Z uczestnikami przeprowadzane są krótkie wywiady, aby głębiej zrozumieć ich doświadczenia i opinie.

6.4. Analiza danych

Analiza zebranych danych obejmuje zarówno metody ilościowe, jak i jakościowe:

1. Analiza ilościowa:
 - Obliczenie średnich ocen dla każdego kryterium i modelu
 - Analiza wariancji (ANOVA) do porównania wyników między modelami
 - Analiza korelacji między oceną modelu a poziomem językowym użytkownika
2. Analiza jakościowa:
 - Kodowanie i kategoryzacja uwag i komentarzy uczestników
 - Identyfikacja powtarzających się tematów i wzorców w opiniach
 - Analiza przypadków szczególnych i odstających
3. Triangulacja danych:
 - Porównanie wyników ilościowych z analizą jakościową
 - Integracja danych z wywiadów uzupełniających
4. Wizualizacja danych:
 - Tworzenie wykresów i diagramów ilustrujących wyniki porównań
 - Opracowanie map cieplnych pokazujących mocne i słabe strony każdego modelu
5. Interpretacja wyników:
 - Identyfikacja kluczowych czynników wpływających na skuteczność modeli w nauce języków
 - Formułowanie rekomendacji dla dalszego rozwoju i zastosowania LLM w edukacji językowej

7. Wnioski z badań

7.1. Podsumowanie wyników porównania

7.2. Kluczowe obserwacje i refleksje

7.3. Praktyczne implikacje dla nauki języków obcych

7.4. Możliwe kierunki dalszych badań

8. Załączniki

8.1. Kwestionariusz użyty w badaniu

Kwestionariusz Oceny Modeli LLM w Nauce Języków

Prosimy o wypełnienie poniższego kwestionariusza, abyśmy mogli ocenić skuteczność różnych modeli językowych w nauce języków poprzez symulowanie rozmowy.

Informacje Użytkownika

1. Nazwa używanego modelu:

☐ GPT-3.5

☐ Gemini-1.5-flash

☐ LLaMa2

☐ Claude 3.5 Sonnet

2. Język, którego się uczyłeś/aś:

☐ Angielski

☐ Hiszpański

☐ Niemiecki

☐ Francuski

☐ Inny: _____

Ocena Funkcjonalności

Prosimy o ocenę poniższych aspektów na skali od 1 do 5, gdzie 1 oznacza "bardzo niezadowolony" a 5 oznacza "bardzo zadowolony".

3. Rozmowa z AI w symulowanej sytuacji:

☐ 1

☐ 2

☐ 3

☐ 4

☐ 5

Uwagi:

4. Tłumaczenie odpowiedzi bota:

☐ 1

☐ 2

☐ 3

☐ 4

☐ 5

Uwagi

5. Ocena zdań użytkownika:

☐ 1

☐ 2

☐ 3

☐ 4

☐ 5

Uwagi:

6. Bezpośrednie pytania na temat języka:

☐ 1

☐ 2

☐ 3

[] 4

[] 5

Uwagi:

Dodatkowe Uwagi

Jeśli masz jakiegolwiek dodatkowe uwagi lub sugestie dotyczące używanego modelu, prosimy o ich podanie poniżej:

Dziękujemy za udział w naszym badaniu!

8.2. Dodatkowe materiały pomocnicze

8.3. Kod źródłowy aplikacji LinguaFranka

<https://github.com/StenzelK/LinguaFranka>