|  |  |
| --- | --- |
|  | **Wydział Finansów i Zarządzania Kierunek: Informatyka** |

Jakub Malerek

(numer albumu: 81704)

Temat pracy

Budowa aplikacji opierającej się na Retrieval-Augmented Generation (RAG) i sztucznej inteligencji dla kontekstowych odpowiedzi na zapytania użytkowników.

Opiekun merytoryczny:

**mgr inż. Wojciech Barczyński**

Wrocław 2024

Streszczenie

Współczesne rozwiązania oparte na sztucznej inteligencji i modelach językowych cieszą się ogromnym zainteresowaniem ze względu na ich innowacyjność i potencjał do optymalizacji procesów biznesowych. Jednym z najnowszych podejść jest technika Retrieval Augmented Generation (RAG), która umożliwia generowanie precyzyjnych odpowiedzi poprzez integrowanie modelu językowego z dynamicznie wyszukiwanymi informacjami z różnych zbiorów danych, w tym danych firmowych oraz zewnętrznych źródeł. RAG znajduje szerokie zastosowanie w automatyzacji pracy, usprawnianiu wyszukiwania informacji, generowaniu kontekstowych instrukcji oraz w budowie zaawansowanych systemów wspomagania decyzji, co znacząco przyczynia się do oszczędności czasu i poprawy efektywności operacyjnej przedsiębiorstw.

W tej pracy zaimplementowałem nowoczesną aplikację opartą na RAG, która specjalizuje się w dostarczaniu najnowszych informacji ze świata technologii. System zapewnia użytkownikowi możliwość uzyskania skondensowanej wiedzy na tematy takie jak kryptowaluty czy gry komputerowe, wspierając się stworzonym silnikiem wyszukiwania (search engine) i integracją danych z wielu źródeł. Aplikacja została zbudowana przy użyciu najnowocześniejszych technologii, takich jak wektorowe bazy danych, usługi chmurowe oraz zaawansowane modele językowe i machine learning. Implementacja bazuje na nowoczesnych bilbiotekach i językach, takich jak Python i React JS, zapewniając wysoką wydajność, skalowalność i nowoczesny interfejs użytkownika.

Projekt obejmuje również procesy inżynierii danych, takie jak zautomatyzowana ekstrakcja i przetwarzanie danych, oraz rozwiązuje wyzwania związane z pracą z wektorami, dużymi modelami językowymi (LLM) oraz przetwarzaniem języka naturalnego (NLP). Celem pracy było stworzenie kompleksowego i nowoczesnego rozwiązania, które prezentuje nowoczesne rozwiązania i dostarczy typ aplikacji aktualnie porządaną na rynku.

SPIS TREŚCI

[1. Wstęp 4](#_Toc182759485)

[1.1. Wprowadzenie do modeli językowych 4](#_Toc182759486)

[1.2. Zastosowanie modeli językowych 5](#_Toc182759487)

[1.3.Ograniczenia modeli językowych 7](#_Toc182759488)

[1.3.1. Wysokie koszty tworzenia i trenowania modeli 8](#_Toc182759489)

[1.3.2. Ograniczona aktualność wiedzy z powodu daty „cutoff” 8](#_Toc182759490)

[1.3.3. Wiedza ogólna zamiast specyficznej i brak dostępu do prywatnych danych 9](#_Toc182759491)

[1.3.4. Zgeneralizowane zachowanie i brak dostosowania do konkretnej roli 9](#_Toc182759492)

[1.3.5. Potencjalne rozwiązania ograniczeń modeli językowych 9](#_Toc182759493)

[1.4. Cel pracy 12](#_Toc182759494)

[2. Retrieval Augmented-Generation 13](#_Toc182759495)

[2.1. Czym jest RAG i jego architektura 13](#_Toc182759496)

[2.1.1. Wektorowa Baza Danych i Podobieństwo Semantyczne 14](#_Toc182759497)

[2.1.2. System Wyszukiwania Informacji (Information Retrieval, IR) 14](#_Toc182759498)

[2.1.3. Model Generatywny (LLM) 14](#_Toc182759499)

[2.1.4. Przepływ pracy w systemie RAG 15](#_Toc182759500)

[2.2. Przetwarzanie języka naturalnego 15](#_Toc182759501)

[2.2.1. Tokenizacja i Przygotowanie Tekstu do Wektoryzacji 15](#_Toc182759502)

[2.2.2. Wektoryzacja Całych Zdań lub Fragmentów Tekstu 16](#_Toc182759503)

[2.2.3. Chunkowanie – Dzielnie Tekstu na Fragmenty 16](#_Toc182759504)

[2.2.5. Reprezentacja Znaczenia Tekstu w Postaci Liczbowej 17](#_Toc182759505)

[2.3. Praca z Wektorami 18](#_Toc182759506)

[2.3.1 Wektorowe Bazy Danych a Tradycyjne Bazy SQL 18](#_Toc182759507)

[2.3.2. Porównywanie Wektorów i Podobieństwo Semantyczne 18](#_Toc182759508)

[3. Inżynieria Danych 20](#_Toc182759509)

[3.1. Dane jako złoto XXI wieku 20](#_Toc182759510)

[3.2. Źródło danych na potrzeby pracy 21](#_Toc182759511)

[3.2.1. Implementacja ekstrakcji danych w module extract.py. 21](#_Toc182759512)

[3.3.2. Przykładowa odpowiedź z API 22](#_Toc182759513)

[3.3. Przetwarzanie i wektoryzacja danych 23](#_Toc182759514)

[3.3.1 Logika przetwarzania danych w transform.py 24](#_Toc182759515)

[3.3.2. Ostateczna struktura przetworzonych danych 24](#_Toc182759516)

[3.4. Wektorowa baza danych 25](#_Toc182759517)

[3.4.1. Indeksowanie w Elasticsearch 25](#_Toc182759518)

[3.4.2. Mapping – struktura i organizacja indeksu 25](#_Toc182759519)

[3.5. Wgranie przetworzonych danych 28](#_Toc182759520)

[3.5.1. Moduł pipeline.py – Koordynacja Procesu ETL 28](#_Toc182759521)

[3.5.2. Struktura Dokumentów Wgrywanych do Elasticsearch 30](#_Toc182759522)

[3.6. Cykl życia danych w projekcie 30](#_Toc182759523)

**1. Wstęp**

## 1.1. Wprowadzenie do modeli językowych

Historia modeli językowych zaczyna się od prób automatycznego przetwarzania tekstu w połowie XX wieku, gdy uczeni pracowali nad pierwszymi programami do tłumaczenia maszynowego. W latach 60. i 70. XX wieku rozwijano modele oparte na regułach językowych, które jednak były mało elastyczne i ograniczone. W kolejnych dekadach pojawiły się podejścia statystyczne, w których komputer analizował dane tekstowe pod kątem częstości występowania słów i fraz, a następnie na tej podstawie próbował przewidywać kolejne elementy w zdaniu. Takie modele, choć były przełomowe w tamtych czasach, miały ograniczoną zdolność do uchwycenia złożoności języka naturalnego.

Prawdziwy postęp nastąpił wraz z rozwojem sztucznych sieci neuronowych oraz metod uczenia głębokiego, a szczególnie od czasu wprowadzenia architektury transformatorów przez zespół badawczy Google w 2017 roku. Transformery umożliwiły przetwarzanie tekstu w kontekście większych fragmentów, analizując zależności między słowami w zdaniu bez konieczności przetwarzania ich po kolei. To odkrycie zrewolucjonizowało modele językowe i zapoczątkowało powstanie dużych modeli, takich jak GPT, BERT czy T5, które oferują niespotykaną wcześniej precyzję i płynność generowanych wypowiedzi.

Modele językowe to obecnie zaawansowane algorytmy, które mogą przewidywać kolejne słowa lub generować całe teksty na podstawie analizy języka naturalnego. Uczą się one na ogromnych zbiorach danych – od książek po strony internetowe – co pozwala im „poznawać” kontekst, znaczenie wyrażeń i strukturę języka. W ten sposób modele takie jak GPT-3, BERT czy T5 potrafią zrozumieć różne zadania językowe i dostosować swoje odpowiedzi do kontekstu, nawet jeśli nie mają bezpośredniego dostępu do „prawdziwego” zrozumienia świata.

Popularność modeli językowych wzrosła znacząco dzięki ich szerokim możliwościom, które znajdują zastosowanie m.in. w tworzeniu chatbotów, analizie opinii w mediach społecznościowych, systemach rekomendacyjnych oraz narzędziach do automatycznego tłumaczenia. Modele te można ogólnie podzielić na dwie główne kategorie:

**Modele generatywne**, takie jak GPT (Generative Pre-trained Transformer), które generują nowe treści. Potrafią one odpowiadać na pytania, pisać eseje czy podsumowywać informacje, bazując na tym, czego nauczyły się podczas treningu.

**Modele rozumiejące**, takie jak BERT (Bidirectional Encoder Representations from Transformers), które są specjalizowane w zrozumieniu i analizie kontekstu tekstu. BERT i podobne modele są stosowane w zadaniach takich jak klasyfikacja tekstu, ekstrakcja informacji czy identyfikacja nastroju wypowiedzi.

Każdy z tych modeli ma swoje unikalne zastosowania i przyczynia się do ulepszania różnych aspektów przetwarzania języka. Współczesne modele językowe, będąc efektem lat badań i innowacji, oferują szeroki wachlarz możliwości zarówno dla użytkowników indywidualnych, jak i przedsiębiorstw. Są integralną częścią systemów informatycznych, wspomagając w automatyzacji, analizie danych oraz interakcji między ludźmi a maszynami w sposób bardziej naturalny niż kiedykolwiek wcześniej.

## 1.2. Zastosowanie modeli językowych

Modele językowe znalazły szerokie zastosowanie w różnych dziedzinach, dzięki ich zdolności do zaawansowanego przetwarzania i generowania języka naturalnego. Głębokie „rozumienie” kontekstu, jakie osiągają te modele, pozwala na skuteczne wykorzystanie ich w różnorodnych aplikacjach biznesowych, edukacyjnych, medycznych i codziennych, oferując rozwiązania, które jeszcze niedawno były trudne do zrealizowania technicznie.

**Obsługa klienta i automatyczne systemy konwersacyjne**

Jednym z najczęściej spotykanych zastosowań modeli językowych jest obsługa klienta, gdzie modele te pełnią rolę chatbotów i wirtualnych asystentów, odpowiadając na zapytania użytkowników, rozwiązując problemy oraz udzielając spersonalizowanych porad. Dzięki zaawansowanemu przetwarzaniu języka naturalnego modele mogą odpowiadać na pytania użytkowników w sposób konwersacyjny, rozumiejąc kontekst i intencje. Tego typu systemy są stosowane przez firmy na całym świecie, umożliwiając automatyzację obsługi klienta, co skutkuje znacznymi oszczędnościami czasu i kosztów oraz poprawą satysfakcji klientów.

**Wyszukiwanie i analiza informacji**

Modele językowe są również szeroko stosowane w zadaniach związanych z wyszukiwaniem informacji oraz analizą tekstów, co jest szczególnie przydatne dla branż opartych na dużej ilości danych, takich jak media, prawo i finanse. Dzięki zaawansowanej analizie semantycznej, modele te potrafią nie tylko wyszukiwać informacje na podstawie słów kluczowych, ale także „rozumieć” kontekst całych pytań, co pozwala na precyzyjne wyszukiwanie najbardziej odpowiednich treści. W dziedzinie analityki modeli językowych stosuje się narzędzia do analizy sentymentu, analizy opinii klientów czy monitorowania nastrojów w mediach społecznościowych, co pozwala firmom lepiej zrozumieć reakcje klientów na produkty i usługi.

**Tworzenie i automatyzacja treści**

Zaawansowane modele generatywne wykorzystywane są do automatycznego tworzenia treści, co ma zastosowanie w marketingu, redagowaniu artykułów czy przygotowywaniu raportów. Modele te mogą generować teksty na różne tematy, naśladując styl i ton komunikacji pożądany przez użytkownika. Automatyzacja treści staje się kluczowym narzędziem w marketingu cyfrowym, gdzie szybkie tworzenie tekstów reklamowych, newsletterów czy postów w mediach społecznościowych pozwala firmom reagować dynamicznie na potrzeby rynku i unikać kosztownego, ręcznego tworzenia materiałów.

**Edukacja i e-learning**

W dziedzinie edukacji modele językowe znajdują zastosowanie jako wirtualni nauczyciele i asystenci, wspierając procesy uczenia się na różnym poziomie, od szkół podstawowych po uniwersytety i szkolenia zawodowe. Modele te mogą odpowiadać na pytania uczniów, tłumaczyć trudne pojęcia, a nawet oceniać zadania pisemne. W kontekście samodzielnej nauki modele językowe pełnią funkcję wsparcia, odpowiadając na pytania i dostarczając wskazówek, co pozwala uczniom na bardziej interaktywne i spersonalizowane podejście do nauki.

**Tłumaczenie maszynowe i komunikacja wielojęzyczna**

Modele językowe wspierają automatyczne tłumaczenie tekstu, co znacząco ułatwia komunikację między osobami posługującymi się różnymi językami. W biznesie, gdzie międzynarodowa komunikacja jest na porządku dziennym, tłumaczenie maszynowe pozwala na szybsze i tańsze przygotowywanie dokumentacji, umów czy materiałów reklamowych. Dzięki zaawansowanemu rozumieniu kontekstu i stylu współczesne modele językowe generują tłumaczenia, które są nie tylko poprawne językowo, ale także zachowują naturalność wypowiedzi, co umożliwia bardziej płynne i zrozumiałe tłumaczenia niż poprzednie generacje narzędzi tłumaczeniowych.

**Diagnostyka i wsparcie medyczne**

W medycynie modele językowe mają potencjał wspomagania wstępnej diagnostyki oraz obsługi pacjenta. Mogą być używane w aplikacjach pomagających użytkownikom w ocenie objawów i sugestii co do dalszych kroków. Przetwarzanie języka naturalnego pomaga analizować notatki medyczne, historie chorób czy wyniki badań, wspomagając lekarzy w podejmowaniu decyzji i skracając czas potrzebny na analizę danych. Co więcej, w badaniach naukowych modele językowe są stosowane do przeszukiwania i przetwarzania literatury medycznej, pomagając lekarzom i naukowcom szybciej dotrzeć do kluczowych informacji i nowości.

**Przetwarzanie dokumentów i analiza treści prawnych**

W prawie i administracji modele językowe przyspieszają przetwarzanie dokumentów, ekstrakcję informacji oraz analizę treści prawnych. Dzięki możliwościom klasyfikacji dokumentów oraz wyodrębniania kluczowych informacji modele te są wykorzystywane przez kancelarie prawne i instytucje do efektywnego przetwarzania umów, analizowania spraw czy sporządzania raportów. Narzędzia te mogą przeszukiwać i analizować obszerne zbiory dokumentów, co pozwala na znaczne usprawnienie pracy w środowisku prawnym, które często wymaga dokładności i precyzji.

**Tworzenie rekomendacji i personalizacja**

Modele językowe odgrywają istotną rolę w tworzeniu systemów rekomendacyjnych, które są szeroko stosowane w handlu elektronicznym, serwisach streamingowych i mediach społecznościowych. Dzięki analizie preferencji i zachowań użytkowników modele te mogą rekomendować produkty, filmy czy artykuły, które najlepiej odpowiadają potrzebom użytkownika. Personalizacja oparta na modelach językowych umożliwia bardziej trafne dopasowanie oferty do indywidualnych zainteresowań, co przekłada się na wyższą jakość doświadczeń użytkownika oraz wzrost sprzedaży i zaangażowania.

Zastosowanie modeli językowych jest dziś niezwykle szerokie, a ich wpływ na wiele sektorów gospodarki jest znaczący. Dzięki rosnącej złożoności i zaawansowaniu w przetwarzaniu języka naturalnego, możliwości ich wykorzystania stale się poszerzają, a modele te zyskują status narzędzi niezbędnych w procesach automatyzacji, analizie danych oraz personalizacji usług.

## 1.3.Ograniczenia modeli językowych

Modele językowe, mimo ich imponujących możliwości, mają istotne ograniczenia, które wpływają na ich użyteczność w wielu zastosowaniach. Poniżej przedstawiono główne problemy, z którymi borykają się użytkownicy dużych modeli językowych (LLM), gdy próbują wdrożyć je w specyficznych kontekstach i zadaniach.

### 1.3.1. Wysokie koszty tworzenia i trenowania modeli

Trening dużych modeli językowych wymaga dostępu do zaawansowanej infrastruktury technologicznej i ogromnych zasobów obliczeniowych. Proces szkolenia takiego modelu polega na analizie setek terabajtów tekstu i miliardów parametrów, co wiąże się z koniecznością użycia wyspecjalizowanego sprzętu, takiego jak GPU (Graphics Processing Unit) czy TPU (Tensor Processing Unit). Trening tego typu modeli trwa tygodnie lub miesiące, a jego koszty mogą sięgać milionów dolarów.

Tak wysoki koszt sprawia, że tylko największe firmy i instytucje badawcze mogą sobie pozwolić na stworzenie i utrzymanie własnych modeli językowych. Dla większości mniejszych firm taki proces jest nieosiągalny, co zmusza je do korzystania z istniejących rozwiązań udostępnianych przez duże organizacje technologiczne. Takie gotowe modele, choć potężne, są najczęściej dostępne w ograniczonej formie komercyjnej i nie zawsze pozwalają na pełne dostosowanie do specyficznych potrzeb firmy

.

### 1.3.2. Ograniczona aktualność wiedzy z powodu daty „cutoff”

Modele językowe działają na podstawie wiedzy, którą zdobyły podczas procesu trenowania. Jednakże trening ten odbywa się w określonym okresie i obejmuje jedynie informacje dostępne do momentu zakończenia szkolenia, co oznacza, że model operuje wiedzą „zamrożoną” na dany moment, nazywaną datą „cutoff”. Wszystko, co wydarzyło się po tej dacie, jest poza zasięgiem modelu – nie ma on dostępu do bieżących informacji ani zdolności do samodzielnej aktualizacji swojej wiedzy.

Na przykład, jeśli model został przeszkolony do końca 2022 roku, to nie będzie świadomy żadnych wydarzeń, nowości czy zmian, które zaszły w 2023 roku. Taki brak aktualności ogranicza użyteczność modelu w szybko zmieniających się branżach, gdzie aktualność danych ma kluczowe znaczenie – dotyczy to m.in. branży finansowej, prawnej, medycznej czy technologicznej. Modele te nie potrafią także samodzielnie zdobywać nowych informacji, co powoduje, że odpowiedzi mogą być przestarzałe lub nieadekwatne, szczególnie w kontekście najnowszych wydarzeń czy innowacji.

### 1.3.3. Wiedza ogólna zamiast specyficznej i brak dostępu do prywatnych danych

Większość dużych modeli językowych jest trenowana na szerokich, ogólnodostępnych zbiorach danych, takich jak artykuły z Internetu, książki, czy encyklopedie. W rezultacie model posiada wiedzę ogólną, a jego odpowiedzi są często zbyt szerokie lub nieprecyzyjne, gdy trzeba dostarczyć informacje specyficzne dla danej firmy, branży lub dziedziny. Modele językowe zazwyczaj nie mają dostępu do danych prywatnych, takich jak dokumenty firmowe, dane wewnętrzne czy zamknięte bazy danych.

Przykładowo, firma farmaceutyczna może oczekiwać, że model pomoże jej w analizie wyników badań klinicznych, ale bez dostępu do wewnętrznych danych model nie będzie w stanie dostarczyć odpowiednich informacji ani kontekstu. Podobnie, w branży finansowej modele mogą nie posiadać wiedzy o specyficznych procedurach raportowania czy wskaźnikach stosowanych w danej firmie. W efekcie modele mogą generować odpowiedzi, które są ogólne, mało użyteczne lub pozbawione kluczowego kontekstu.

### **1.3.4. Zgeneralizowane zachowanie i brak dostosowania do konkretnej** roli

Standardowe modele językowe są tworzone jako systemy uniwersalne, co oznacza, że ich odpowiedzi są neutralne i nie dostosowane do specyficznych ról, jakie mogą pełnić w różnych branżach. Takie modele nie potrafią naśladować stylu komunikacji typowego dla specjalistycznych asystentów czy ekspertów, chyba że zostaną odpowiednio przystosowane. Bez tego przystosowania model może odpowiadać w sposób formalny, uogólniony i nieadekwatny do wymogów, jakie stawia mu konkretny kontekst biznesowy.

Na przykład, w branży prawniczej odpowiedzi modelu muszą być precyzyjne, dostosowane do języka prawniczego i odpowiednie pod względem struktury dokumentów, co wymaga przystosowania modelu do roli „asystenta prawniczego”. W sektorze technologicznym odpowiedzi muszą odzwierciedlać specyficzny język używany przez ekspertów w dziedzinie IT, w tym terminologię i poziom szczegółowości odpowiedni dla specjalistów. Modele ogólne nie są w stanie sprostać takim wymaganiom bez dodatkowych metod ich ukierunkowania.

### 1.3.5. Potencjalne rozwiązania ograniczeń modeli językowych

Aby skutecznie wykorzystać modele językowe w konkretnych kontekstach i dostosować je do specyficznych potrzeb, stosuje się różne techniki pozwalające na personalizację modeli i poprawę ich efektywności. Do najczęściej używanych metod należą fine-tuning, inżynieria promptów (ang. *prompt engineering*) oraz Retrieval-Augmented Generation (RAG). Każda z tych metod ma swoje zalety i ograniczenia, które decydują o ich praktycznym zastosowaniu.

#### 1.3.5.1. Fine-tuning jako metoda dostosowania modeli

Fine-tuning to technika dodatkowego trenowania modelu na specyficznym, zazwyczaj węższym zbiorze danych, aby poprawić jego zdolność do odpowiadania na pytania z określonej dziedziny lub kontekstu. Dzięki fine-tuningowi model może dostosować swoje odpowiedzi do wymagań konkretnej branży, lepiej rozumieć język specjalistyczny i zachowywać się zgodnie z przyjętymi w firmie standardami komunikacji. Na przykład, kancelaria prawna może przeprowadzić fine-tuning modelu na zbiorze dokumentów prawniczych, co pozwoli na dokładniejsze rozumienie języka prawniczego i terminologii.

Fine-tuning może także rozwiązać problem braku aktualnych danych – przynajmniej częściowo – poprzez dostarczenie modelowi wiedzy specyficznej dla danej organizacji. Pozwala on także na „nauczanie” modelu nowego stylu lub roli, tak aby był w stanie pełnić funkcję eksperta lub doradcy w wybranej dziedzinie.

Ograniczenia fine-tuningu:

* **Wysokie koszty**: Proces fine-tuningu wymaga znacznych zasobów obliczeniowych i odpowiednich danych, co może generować wysokie koszty. Fine-tuning jest również czasochłonny i wymaga wiedzy specjalistycznej, przez co jest dostępny głównie dla organizacji dysponujących odpowiednimi zasobami technicznymi.
* **Ograniczona elastyczność**: Fine-tuning pozwala dostosować model do jednego kontekstu, co oznacza, że model stanie się bardziej przydatny w danej roli, ale jego zdolność do zmiany specjalizacji jest ograniczona. Jeśli firma będzie potrzebowała dostosować model do innego kontekstu, konieczne będzie przeprowadzenie nowego procesu fine-tuningu na innych danych.
* **Brak aktualizacji w czasie rzeczywistym**: Fine-tuning nie rozwiązuje problemu „zamrożonej” wiedzy modelu – po zakończeniu tego procesu model nadal nie będzie miał dostępu do najnowszych informacji ani możliwości samodzielnej aktualizacji.

Z uwagi na powyższe ograniczenia fine-tuning nie zawsze jest praktyczny i dostępny, zwłaszcza dla organizacji, które potrzebują elastycznego modelu, działającego w dynamicznym środowisku lub obsługującego różne konteksty. W odpowiedzi na te ograniczenia stosuje się alternatywne rozwiązania, takie jak inżynieria promptów oraz Retrieval-Augmented Generation (RAG).

#### 1.3.5.2. Inżynieria promptów (*Prompt Engineering*)

Inżynieria promptów to technika, która polega na formułowaniu zapytań i instrukcji w taki sposób, aby model odpowiedział zgodnie z oczekiwaniami użytkownika. Jest to metoda zmiany zachowania modelu poprzez odpowiednie przedstawienie mu kontekstu i roli, jaką ma pełnić, bez konieczności dodatkowego trenowania. Inżynieria promptów może być stosowana do tego, aby model działał jak wyspecjalizowany doradca, odpowiadał zgodnie z preferowanym stylem lub skupiał się na specyficznych aspektach zadania.

Na przykład, aby model zachowywał się jak asystent prawny, można sformułować prompt w stylu: „Jako doradca prawny przedstaw analizę prawną tego zagadnienia, biorąc pod uwagę obowiązujące przepisy”. Podobnie, w kontekście technologii, prompt może nakierować model, aby działał jak ekspert techniczny: „Jako specjalista ds. technologii wyjaśnij zasadę działania tego urządzenia, uwzględniając jego główne zalety i ograniczenia”.

Zalety inżynierii promptów:

* **Niski koszt**: Inżynieria promptów nie wymaga przeprowadzania dodatkowego treningu, więc jest bardziej dostępna i elastyczna niż fine-tuning.
* **Elastyczność**: Metoda ta pozwala na łatwe dostosowanie modelu do różnych ról poprzez zmianę promptów, co jest przydatne w środowiskach, które wymagają częstych zmian kontekstu.

Ograniczenia inżynierii promptów:

* **Brak pełnej kontroli nad modelem**: Model może różnie interpretować zapytania i nie zawsze odpowiedzieć w sposób zgodny z oczekiwaniami, zwłaszcza w bardziej skomplikowanych lub specjalistycznych zadaniach.
* **Brak dostępu do aktualnych danych**: Inżynieria promptów pozwala na ukierunkowanie zachowania modelu, ale nie rozwiązuje problemu przestarzałej wiedzy modelu ani braku dostępu do najnowszych danych.

#### 1.3.5.3. Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) to podejście, które łączy możliwości generatywne modeli językowych z dostępem do zewnętrznych źródeł danych. RAG umożliwia modelowi sięganie do aktualnych i specyficznych danych w czasie rzeczywistym, dzięki czemu model jest w stanie dostarczać odpowiedzi bardziej zgodnych z bieżącymi wymaganiami i opartych na najnowszych informacjach.

W systemie RAG model językowy generuje odpowiedzi, korzystając nie tylko z wiedzy zawartej w modelu, ale także z wyników wyszukiwania w zewnętrznych bazach danych, dokumentach firmowych czy internetowych zasobach informacji. Przykładowo, model w firmie finansowej może uzyskiwać dostęp do najnowszych danych rynkowych lub analiz ekonomicznych, co pozwala na generowanie odpowiedzi, które są aktualne i kontekstowo adekwatne. Dzięki temu RAG skutecznie rozwiązuje problem „daty cutoff” oraz braku dostępu do specyficznych danych, na które cierpią standardowe modele.

#### 1.3.5.4. Połączenie RAG i inżynierii promptów jako elastyczne rozwiązanie

Kombinacja RAG i inżynierii promptów umożliwia tworzenie systemów, które jednocześnie dostarczają odpowiedzi oparte na aktualnych danych oraz przyjmują konkretną rolę w komunikacji. Inżynieria promptów pozwala modelowi pełnić funkcję specjalistycznego doradcy czy analityka, a RAG dostarcza mu na bieżąco informacji spoza modelu, czyniąc go bardziej trafnym i precyzyjnym w dynamicznych środowiskach.

Na przykład, system RAG dla doradcy prawnego może łączyć odpowiednie prompty, aby model zachowywał się jak asystent prawniczy, oraz korzystać z bieżących danych prawniczych czy dokumentów prawnych, co zapewnia aktualność i trafność odpowiedzi. Dzięki połączeniu RAG z inżynierią promptów organizacje mogą tworzyć systemy dostosowane do specyficznych ról i dynamicznych potrzeb, bez konieczności kosztownego fine-tuningu.

## 1.4. Cel pracy

Celem niniejszej pracy jest opracowanie rozwiązania, które przezwycięży kluczowe ograniczenia dużych modeli językowych (LLM) i zastosuje je w praktycznym, biznesowym kontekście. Projekt zakłada stworzenie chatbota – eksperta technologicznego – zdolnego do dostarczania najnowszych informacji ze świata technologii. Chatbot ten będzie szczególnie przydatny dla osób zainteresowanych technologią, twórców treści, programistów oraz innych specjalistów, automatyzując i usprawniając proces pozyskiwania informacji oraz śledzenia bieżących trendów i nowinek w branży.

Aplikacja zostanie zbudowana w oparciu o nowoczesny stos technologiczny, który umożliwi jej efektywne działanie i skalowalność:

* **Python** jako główny język programowania posłuży do stworzenia back-endu aplikacji, w tym implementacji logiki działania chatbota oraz obsługi modeli językowych i procesów związanych z przetwarzaniem języka naturalnego,
* **React (JavaScript)** zapewni interaktywny i intuicyjny front-end, dzięki któremu użytkownicy będą mogli wygodnie korzystać z chatbota i szybko uzyskiwać odpowiedzi,
* **Wektorowe bazy danych** będą przechowywać i wyszukiwać informacje, umożliwiając chatbotowi dostęp do najnowszych danych technologicznych i dynamiczne wzbogacanie odpowiedzi,
* **Technologie chmurowe** pozwolą na wdrożenie aplikacji w sposób skalowalny i dostępny online, co umożliwi obsługę dużej liczby zapytań użytkowników w czasie rzeczywistym i zapewni stabilność działania całego rozwiązania.

Realizacja tego projektu pozwoli nie tylko przezwyciężyć ograniczenia LLM poprzez zastosowanie Retrieval-Augmented Generation (RAG) i inżynierii promptów, ale także stworzyć aplikację, która wykorzysta najnowsze technologie w celu zapewnienia wysokiej wydajności i wartości użytkowej dla szerokiego grona odbiorców.

# 2. Retrieval Augmented-Generation

## 2.1. Czym jest RAG i jego architektura

Retrieval Augmented-Generation (RAG) to technika, która łączy możliwości dużych modeli językowych (LLM) z systemami wyszukiwania informacji, co pozwala na generowanie odpowiedzi opartych nie tylko na wiedzy modelu, ale również na aktualnych i specyficznych danych zewnętrznych. Kluczowym procesem w RAG jest „indeksowanie” treści – czyli przekształcanie tekstów (np. artykułów, dokumentów) na wektory liczbowe, które odzwierciedlają semantyczne znaczenie każdego tekstu. Te wektory przechowywane są w specjalnej bazie danych, co pozwala na szybkie odnajdywanie treści podobnych znaczeniowo, a nie tylko literalnie zgodnych z zapytaniem użytkownika. Dzięki temu RAG może wyszukiwać treści odpowiadające intencjom użytkownika.

### 2.1.1. Wektorowa Baza Danych i Podobieństwo Semantyczne

Wektorowa baza danych jest jednym z najważniejszych elementów systemu RAG, ponieważ przechowuje dane w postaci wektorów, czyli liczbowych reprezentacji znaczenia semantycznego tekstów. Dzięki temu możliwe jest wyszukiwanie podobieństwa semantycznego między treściami. W podobieństwie semantycznym chodzi o to, że teksty o podobnym znaczeniu mają wektory położone blisko siebie w przestrzeni wektorowej. Przykładowo, zdania takie jak „Zalety energii słonecznej” i „Dlaczego warto inwestować w panele słoneczne” będą miały podobne wektory, gdyż poruszają ten sam temat, choć są inaczej sformułowane.

### 2.1.2. System Wyszukiwania Informacji (Information Retrieval, IR)

Komponent odpowiedzialny za wyszukiwanie informacji w RAG to **Information Retrieval (IR)**, co można przetłumaczyć jako „wyszukiwanie informacji” lub „system wyszukiwania informacji”. IR analizuje zapytanie użytkownika, przekształca je na wektor, a następnie porównuje ten wektor z zapisanymi w wektorowej bazie danych, aby znaleźć semantycznie zbliżone treści. Działa to podobnie do wyszukiwarki, która zamiast szukać identycznych słów, odnajduje treści o podobnym znaczeniu.

Przykładowo, jeśli użytkownik pyta: „Jakie są aktualne trendy w kryptowalutach?”, IR przekształca zapytanie w wektor i porównuje go z wektorami przechowywanymi w bazie danych. Wyniki mogą obejmować treści takie jak:

* „Wzrost popularności DeFi wśród inwestorów.”
* „Ekologiczne alternatywy w kopaniu kryptowalut.”
* „Stablecoiny jako odpowiedź na zmienność rynku.”

IR zwraca te najbardziej pasujące wyniki do LLM, które wykorzystuje je w generowaniu odpowiedzi.

### 2.1.3. Model Generatywny (LLM)

Model językowy (LLM) odpowiada za stworzenie ostatecznej odpowiedzi. Po otrzymaniu wyników z IR, LLM łączy je z własną, wbudowaną wiedzą, dzięki czemu jest w stanie stworzyć kontekstową i precyzyjną odpowiedź, która odpowiada na bieżące potrzeby użytkownika. Działanie modelu generatywnego w RAG zwiększa jego elastyczność i trafność odpowiedzi, gdyż LLM „uczy się” kontekstu zapytania od wyników IR.

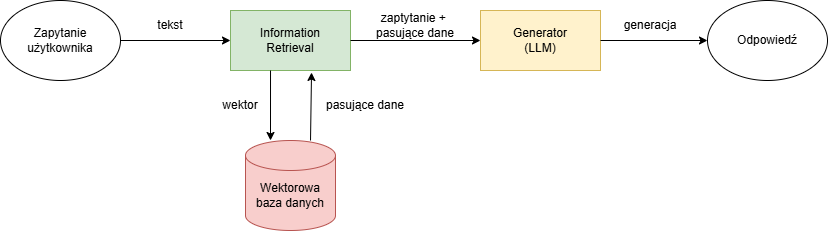
### 2.1.4. Przepływ pracy w systemie RAG

Przepływ pracy w RAG obejmuje cztery główne kroki:

1. Użytkownik wprowadza zapytanie, które trafia do IR.
2. IR konwertuje zapytanie na wektor i przeszukuje bazę danych, zwracając wyniki semantycznie zbliżone do zapytania. Wyniki te są przekazywane do LLM, który integruje je z własną wiedzą.
3. LLM generuje odpowiedź opartą na wynikach wyszukiwania i przekazuje ją użytkownikowi.

Taki przepływ zapewnia, że odpowiedzi RAG są zarówno precyzyjne, jak i aktualne.

Przykładowy przepływ pracy:



Stworzony przeze mnie system RAG będzie miał jeszcze inne usprawnienia.

## 2.2. Przetwarzanie języka naturalnego

Przetwarzanie języka naturalnego (NLP) jest kluczowym elementem systemu Retrieval-Augmented Generation (RAG), który umożliwia przekształcenie tekstu na formę liczbową, czyli wektor. Dzięki NLP, system RAG jest w stanie zrozumieć znaczenie tekstu na poziomie semantycznym, co pozwala mu na trafne odnajdywanie fragmentów danych, które najlepiej odpowiadają na zapytania użytkowników.

### 2.2.1. Tokenizacja i Przygotowanie Tekstu do Wektoryzacji

Pierwszym krokiem przetwarzania tekstu jest **tokenizacja**, czyli podział tekstu na mniejsze jednostki zwane tokenami. Tokeny są podstawowymi fragmentami języka – najczęściej pojedynczymi słowami lub ich częściami. Tokenizacja jest niezbędna dla działania modelu, ponieważ umożliwia przekształcenie tekstu w postać zrozumiałą dla komputera.

Na przykład, zdanie „Energia słoneczna jest przyjazna dla środowiska” zostanie rozbite na tokeny, takie jak „Energia”, „słoneczna”, „jest”, „przyjazna”, „dla”, „środowiska”. Następnie każdy token jest przetwarzany przez model NLP, aby przygotować tekst do wektoryzacji.

### 2.2.2. Wektoryzacja Całych Zdań lub Fragmentów Tekstu

Po tokenizacji tekst może być reprezentowany przez wektor dla całego zdania lub fragmentu tekstu. Nowoczesne modele NLP, takie jak BERT, GPT-3, czy RoBERTa, przekształcają tekst na dwa sposoby: tworząc pojedyncze wektory dla każdego tokenu oraz generując zbiorczy wektor dla całego zdania lub fragmentu tekstu. W systemie RAG, to właśnie zbiorczy wektor dla całego zdania lub fragmentu jest wykorzystywany najczęściej, ponieważ pozwala na reprezentowanie całościowego znaczenia i kontekstu tekstu.

Przykładowo, zdanie „Energia słoneczna jest przyjazna dla środowiska” może być przekształcone w pojedynczy wektor, który zawiera zbiorczą reprezentację jego znaczenia. Taki wektor pozwala systemowi na efektywne porównywanie i wyszukiwanie semantycznie podobnych treści na poziomie całych zdań czy zapytań.

### 2.2.3. Chunkowanie – Dzielnie Tekstu na Fragmenty

W przypadku dłuższych dokumentów lub dużych tekstów, system RAG stosuje chunkowanie (ang. chunking), czyli dzielenie tekstu na mniejsze fragmenty, zwane „chunkami”. Chunkowanie pozwala systemowi na skuteczne przetwarzanie dużych objętości danych, unikając konieczności analizowania całego dokumentu jednorazowo.

Na przykład, długi raport o energii odnawialnej może zostać podzielony na fragmenty skupiające się na różnych aspektach, takich jak „zalety ekologiczne”, „koszty wdrożenia” czy „wyzwania technologiczne”. Każdy taki fragment zostaje przekształcony na osobny wektor, co umożliwia systemowi precyzyjniejsze dopasowanie kontekstu zapytania użytkownika do odpowiednich fragmentów dokumentu.

**2.2.4. Różne Metody Embedding i Ich Rola**

Embedding (czyli osadzanie tekstu w formie wektorowej) polega na przedstawieniu znaczenia tekstu jako zbioru liczb. Każdy wymiar tego wektora (ang. dimension) odzwierciedla pewien aspekt znaczenia. Modele embedding stosują różne podejścia, co wpływa na precyzję reprezentacji i jej zastosowanie w RAG.

* **Word2Vec i GloVe** – Te modele nadają każdemu słowu stały wektor, niezależnie od kontekstu, co jest szybkie, lecz mniej precyzyjne. Wektory te mają małą liczbę wymiarów (np. 50–300), co sprawdza się przy prostszych analizach, ale nie nadaje się do bardziej zaawansowanych zastosowań, gdzie kontekst zdania jest kluczowy.
* **ELMo** – Model ten generuje dynamiczne wektory zależne od kontekstu, co oznacza, że słowa zmieniają swoje reprezentacje zależnie od ich użycia w zdaniu. ELMo tworzy wektory o większej liczbie wymiarów (np. 1024), które dokładniej odwzorowują znaczenie słów w zależności od kontekstu.
* **BERT i transformery** – Modele te przekształcają każde zdanie lub fragment na pojedynczy, zbiorczy wektor, który uwzględnia relacje między wszystkimi słowami w zdaniu. Dzięki wyższej liczbie wymiarów (768 lub więcej), BERT i inne modele transformerowe są szczególnie przydatne w RAG, ponieważ dokładnie odwzorowują kontekst i znaczenie całego tekstu.
* **LLM, takie jak GPT-3** – Modele te generują bardzo szczegółowe wektory (nawet powyżej 2000 wymiarów), co pozwala na przechwycenie złożonych zależności semantycznych. Taka precyzja pomaga w zadaniach NLP, ale może też prowadzić do tzw. **przekleństwa wymiarowości** (ang. **curse of dimensionality**), gdzie zbyt duża liczba wymiarów sprawia, że wektory stają się trudniejsze do porównania i mniej efektywne w wyszukiwaniu.

### 2.2.5. Reprezentacja Znaczenia Tekstu w Postaci Liczbowej

Dzięki embeddingom RAG może przełożyć znaczenie całego zdania na wektor, który odzwierciedla jego sens na poziomie liczbowym. Teksty o podobnym znaczeniu mają wektory położone blisko siebie w wielowymiarowej przestrzeni, co pozwala RAG na efektywne wyszukiwanie treści o podobnym znaczeniu. Na przykład, zdania „Energia słoneczna jest przyjazna dla środowiska” i „Energia odnawialna sprzyja ochronie przyrody” będą miały wektory blisko siebie, mimo różnic w słowach. Dzięki temu RAG może rozpoznać, że oba zdania dotyczą podobnych zagadnień.

To podejście umożliwia RAG generowanie precyzyjnych, semantycznie zgodnych odpowiedzi na zapytania użytkowników, analizując znaczenie na poziomie całych zdań i fragmentów tekstu, a nie jedynie dopasowania słów kluczowych.

## 2.3. Praca z Wektorami

W systemie Retrieval Augmented-Generation (RAG) praca z wektorami jest kluczowym elementem, który umożliwia efektywne przeszukiwanie danych i dostarczanie precyzyjnych odpowiedzi. Wektorowe bazy danych, które przechowują treści w formie liczbowych reprezentacji, różnią się znacząco od tradycyjnych baz danych SQL. Wektorowe bazy danych są zaprojektowane tak, aby przechowywać semantyczne reprezentacje tekstu i umożliwiać szybkie porównywanie znaczenia fragmentów tekstu, nawet jeśli nie zawierają identycznych słów.

### 2.3.1 Wektorowe Bazy Danych a Tradycyjne Bazy SQL

W tradycyjnych bazach danych SQL dane są przechowywane w strukturze tabel, a wyszukiwanie opiera się na dokładnym dopasowaniu wartości w tabelach. Na przykład, jeśli baza zawiera kolumnę „produkt” z wartościami takimi jak „telefon” i „laptop”, wyszukiwanie zwróci jedynie dokładne wyniki dla zapytań zawierających te same słowa. Ten sposób przechowywania i wyszukiwania jest szybki i skuteczny dla prostych danych, ale nie radzi sobie z analizą znaczenia lub intencji użytkownika.

W wektorowych bazach danych, takich jak Pinecone czy FAISS, każdy element danych jest przekształcany na wektor – liczbową reprezentację odzwierciedlającą jego semantyczne znaczenie. W przeciwieństwie do baz SQL, wektorowe bazy danych umożliwiają wyszukiwanie na podstawie podobieństwa semantycznego. Oznacza to, że zapytanie użytkownika jest przekształcane w wektor, który porównywany jest z wektorami w bazie danych, aby odnaleźć najbliższe znaczeniowo wyniki.

### 2.3.2. Porównywanie Wektorów i Podobieństwo Semantyczne

Aby porównywać wektory na poziomie semantycznym, wektorowe bazy danych stosują różne metody wyznaczania podobieństwa między wektorami. Podstawowe metody to:

1. **Odległość kosinusowa (Cosine Similarity)** – Odległość kosinusowa mierzy kąt między dwoma wektorami, co pozwala na ocenę podobieństwa niezależnie od ich długości. Im mniejszy kąt między wektorami, tym większe podobieństwo semantyczne. Jest to jedna z najczęściej stosowanych metod, ponieważ efektywnie identyfikuje podobne znaczeniowo teksty.
2. **Odległość euklidesowa (Euclidean Distance)** – Oblicza bezpośrednią odległość między punktami wektorów w wielowymiarowej przestrzeni. Im mniejsza odległość euklidesowa między wektorami, tym większe ich podobieństwo. Ta metoda jest bardziej wrażliwa na różnice w długości wektorów, dlatego sprawdza się, gdy liczba wymiarów jest stosunkowo mała.
3. **Odległość manhattańska (Manhattan Distance)** – Jest to suma wartości bezwzględnych różnic między współrzędnymi wektorów. Metoda ta jest mniej popularna w wyszukiwaniu semantycznym, ale bywa użyteczna w przypadku konkretnych algorytmów.

W wektorowych bazach danych porównywanie wektorów często odbywa się za pomocą algorytmów wyszukiwania przybliżonego najbliższego sąsiada (ang. Approximate Nearest Neighbor Search, ANN). Algorytmy te, takie jak HNSW (Hierarchical Navigable Small World) czy LSH (Locality-Sensitive Hashing), pozwalają na szybkie odnalezienie wektorów najbliższych do wektora zapytania, co znacząco przyspiesza przeszukiwanie bazy przy dużej liczbie danych.

* **HNSW (Hierarchical Navigable Small World)** – Jest to algorytm strukturyzujący wektory w wielopoziomową sieć, gdzie najbliżsi sąsiedzi są szybko odnajdywani na podstawie relacji przestrzennych między nimi. HNSW jest szczególnie skuteczny dla dużych zbiorów danych, ponieważ pozwala na szybkie wyznaczanie podobieństwa w wielowymiarowej przestrzeni.
* **LSH (Locality-Sensitive Hashing)** – Jest to technika haszowania wektorów o podobnych wartościach do tych samych przedziałów, co umożliwia szybkie wyszukiwanie bez konieczności dokładnego porównywania każdego wektora. LSH pozwala na optymalizację wyszukiwania przy dużej liczbie wektorów, choć nie jest tak dokładny jak HNSW.

Kiedy użytkownik zadaje zapytanie, jest ono przekształcane na wektor za pomocą modelu NLP (np. BERT). Ten wektor reprezentuje znaczenie zapytania i jest porównywany z wektorami zindeksowanymi w bazie danych. Algorytmy wyszukiwania (np. HNSW) szybko odnajdują wektory najbliższe do wektora zapytania na podstawie wybranej miary podobieństwa (np. odległości kosinusowej). Wynikiem jest zestaw fragmentów tekstu, które najlepiej pasują znaczeniowo do pytania użytkownika, nawet jeśli używają innych słów.

Takie porównanie na poziomie semantycznym umożliwia RAG generowanie trafnych odpowiedzi, bazując na rzeczywistym znaczeniu zapytania, a nie na dokładnym dopasowaniu słów kluczowych, co jest ograniczeniem tradycyjnych baz SQL.

# 3. Inżynieria Danych

## 3.1. Dane jako złoto XXI wieku

Wszystkie najważniejsze elementy i procesy związane z działaniem systemu Retrieval Augmented-Generation (RAG) – takie jak NLP, wektoryzacja tekstu, wektorowe bazy danych oraz metody wyszukiwania – zostały już omówione. Jednak fundamentem każdego z tych etapów jest posiadanie odpowiednich danych. Dane stanowią podstawę i punkt wyjścia do budowy całego systemu, determinując jego możliwości oraz jakość uzyskiwanych wyników. W dzisiejszych czasach dane bywają określane mianem "złota XXI wieku", co podkreśla ich wartość oraz kluczową rolę w rozwoju nowoczesnych technologii, w tym również sztucznej inteligencji.

Porównanie danych do „złota” nawiązuje nie tylko do ich wartości, ale również do wyzwań związanych z ich pozyskaniem i przetwarzaniem. Dane są niezbędnym zasobem, jednak dostęp do wartościowych, aktualnych i dobrze zorganizowanych danych bywa trudny. Firmy i instytucje często muszą inwestować znaczne środki w ich zbieranie, weryfikowanie, oczyszczanie i aktualizowanie. Co więcej, odpowiednio przygotowane dane muszą być różnorodne, szczegółowe oraz precyzyjnie dostosowane do specyfiki zastosowania – w przeciwnym razie model będzie podatny na błędy, niepełne odpowiedzi i brak kontekstu.

Dostęp do wysokiej jakości danych staje się tym trudniejszy, im bardziej specyficzna jest tematyka, jakiej dotyczą. W przypadku omawianego projektu, którego tematem jest chatbot dostarczający informacji ze świata technologii i najnowszych wiadomości, niezbędne będą źródła stale aktualizowane, aby model mógł odpowiadać na bieżące pytania użytkowników. W szczególności potrzebne są dane dotyczące nowości w branży technologicznej, takich jak kryptowaluty, gry komputerowe, czy nowe rozwiązania technologiczne.

W kontekście RAG, pozyskane dane muszą zostać nie tylko zebrane, ale także odpowiednio przetworzone, aby mogły zostać zapisane w wektorowej bazie danych i wykorzystane do analizy semantycznej. Procesy te wymagają zastosowania technik ekstrakcji, czyszczenia, a także wektoryzacji, co pozwoli na uzyskanie precyzyjnych reprezentacji semantycznych każdego fragmentu tekstu.

## 3.2. Źródło danych na potrzeby pracy

Aby zapewnić dostęp do najbardziej aktualnych wiadomości technologicznych, źródłem danych dla systemu jest News API (<https://newsapi.org/>), które umożliwia dostęp do milionów artykułów z ponad 150,000 serwisów informacyjnych, w tym zarówno głównych wydawców, jak i mniejszych blogów. Korzystanie z News API pozwala na przeszukiwanie artykułów według słów kluczowych, dat publikacji i innych parametrów, co ułatwia tworzenie zestawienia najnowszych informacji w wybranych obszarach technologicznych.

News API oferuje bezpłatny plan dostępu dla deweloperów, który umożliwia korzystanie z zasobów w ograniczonym zakresie, co jest wystarczające dla celów badawczych i akademickich, takich jak praca inżynierska. Polityka użytkowania serwisu pozwala na wykorzystywanie danych w projektach naukowych, dzięki czemu można legalnie pobierać i przetwarzać treści w ramach tego projektu

.

### 3.2.1. Implementacja ekstrakcji danych w module extract.py.

Ekstrakcja danych realizowana jest w języku Python, w specjalnym module o nazwie **extract.py**. Moduł ten zawiera trzy główne funkcje, które wspólnie odpowiadają za pobieranie, organizowanie i przygotowanie danych do dalszego przetwarzania. Proces ten obejmuje przeszukiwanie News API pod kątem wybranych tematów technologicznych, iteracyjne pobieranie wyników dla każdego dnia z ostatniego tygodnia

.

Główne funkcje modułu extract.py:

1. **get\_tech\_news** – Ta funkcja stanowi główny element modułu odpowiedzialny za pobieranie artykułów na podstawie wcześniej zdefiniowanych tematów technologicznych. Działa w następujący sposób:
   1. **Przygotowanie zapytań**: Dla każdego z tematów (np. „blockchain”, „sztuczna inteligencja”) funkcja przygotowuje zapytanie do News API z ustalonymi parametrami. Parametr q odpowiada za słowa kluczowe, from i to ustalają zakres dat dla konkretnego dnia, a sortBy określa sposób sortowania wyników (np. według trafności lub daty publikacji).
   2. **Wysyłanie zapytań**: Funkcja iteracyjnie wysyła zapytania do News API, pobierając artykuły powiązane z każdym z tematów na wybrany dzień z ostatniego tygodnia. Każdego dnia funkcja wykonuje zapytania dla każdego tematu osobno, co umożliwia systematyczne przeszukiwanie całego tygodnia dla wybranych obszarów technologicznych.
   3. **Deduplikacja**: Aby uniknąć zapisania duplikatów, funkcja sprawdza adres URL każdego artykułu. Jeśli URL jest unikalny, artykuł zostaje zapisany, a jego URL dodany do listy sprawdzonych. W ten sposób baza danych zawiera tylko unikatowe artykuły.
   4. **Obsługa błędów**: Funkcja uwzględnia obsługę błędów, takich jak błędy HTTP czy problemy z połączeniem, i zapewnia komunikaty w przypadku problemów z dostępem do serwera API.
2. **iterate\_days** – Funkcja pomocnicza, której zadaniem jest wygenerowanie przedziałów czasowych dla każdego dnia z ostatniego tygodnia.

Funkcja tworzy listę dat reprezentujących każdy dzień z ostatniego tygodnia, co umożliwia systematyczne i jednolite ustawianie zakresu dat w zapytaniach do News API. Każda wygenerowana data jest używana jako wartość parametrów from i to, aby zawęzić wyniki dla poszczególnych dni.

Dzięki temu rozwiązaniu dane są pobierane z rozdzieleniem na dni, a artykuły zebrane w ciągu tygodnia są uporządkowane według daty publikacji, co ułatwia ich późniejszą analizę.

1. **recent\_week\_etl** – Ta funkcja koordynuje pobieranie artykułów dla każdego dnia w tygodniu i łączy wyniki z różnych tematów.
   1. **Przeszukiwanie po dniach**: Funkcja iteruje przez listę dat wygenerowanych przez iterate\_days i dla każdego dnia wywołuje funkcję get\_tech\_news, pobierając artykuły dla każdego tematu oddzielnie. Dzięki takiemu rozwiązaniu każdy temat jest analizowany na podstawie danych z każdego dnia tygodnia.
   2. **Kompilacja wyników**: Po pobraniu danych dla wszystkich tematów i dni z ostatniego tygodnia, funkcja łączy wszystkie artykuły w jeden zbiór wyników.Wyniki te są gotowe do dalszego przetwarzania.

### 3.3.2. Przykładowa odpowiedź z API

News API zwraca dane w formacie JSON, co umożliwia łatwe przetwarzanie i analizę pozyskanych informacji. Każdy artykuł jest reprezentowany jako dokument JSON, zawierający kluczowe informacje, takie jak źródło publikacji, autor, tytuł, opis, link do artykułu oraz data publikacji. Dzięki temu struktura jest czytelna i ułatwia późniejszą wektoryzację oraz integrację danych w systemie RAG.

Dla przykładu, odpowiedź API dla pojedynczego artykułu zawiera następująco:

* **source** – informacje o źródle artykułu, takie jak ID i nazwa serwisu,
* **author** – nazwisko autora,
* **title** – tytuł artykułu,
* **description** – krótki opis lub fragment artykułu,
* **url** – bezpośredni link do artykułu,
* **urlToImage** – link do grafiki powiązanej z artykułem,
* **publishedAt** – data i czas publikacji,
* **content** – nieformatowany fragment treści artykułu (do 200 znaków).

Przy pozyskiwaniu artykułów na wybrane tematy technologiczne z ostatniego tygodnia, proces ekstrakcji zwraca około 3,000 dokumentów, które są gotowe do dalszego przetwarzania. W celu wzbogacenia metadanych, do każdego dokumentu dodawane jest również pole „topic”, które określa temat technologiczny, z jakiego pochodzi dany artykuł. Wartość tego pola jest ustalana na podstawie parametru „q” użytego w zapytaniu, który definiuje poszukiwany temat.

## 3.3. Przetwarzanie i wektoryzacja danych

W module **transform.py** rozpoczyna się kluczowy proces przekształcania pobranych artykułów w numeryczne reprezentacje, czyli wektory. Wektoryzacja umożliwia systemowi przetwarzania informacji (IR) porównywanie semantycznego znaczenia artykułów i dopasowywanie ich do zapytań użytkowników w systemie Retrieval-Augmented Generation (RAG). Wcześniej omówione techniki NLP oraz rola wektorów w RAG znajdują tu praktyczne zastosowanie.

Do generowania wektorów używam osobnego modułu embedding.py, który zawiera klasę TextEmbedder. Moduł ten korzysta z biblioteki transformers i wykorzystuje model NoInstruct-small-Embedding-v0 dostępny na platformie Hugging Face. Model ten generuje wektory o 384 wymiarach, co umożliwia precyzyjne odwzorowanie semantycznego znaczenia tekstów przy zachowaniu wydajności. Metoda „Asymmetric Pooling”, stosowana przez ten model, optymalizuje wyniki wyszukiwania i dopasowywania treści, co jest szczególnie istotne w systemach opartych na RAG.

### 3.3.1 Logika przetwarzania danych w transform.py

Moduł **transform.py** wykorzystuje klasę **TextEmbedder** do generowania wektorów dla wybranych pól artykułów, takich jak **title** (tytuł), **description** (opis) i **content** (pełna treść artykułu). Wektory tych pól umożliwią systemowi IR odnajdywanie dokumentów, które najlepiej odpowiadają semantycznie na zapytania użytkowników, ponieważ stanowią one najbardziej istotny kontekst artykułów.

Oprócz wektorów dla kluczowych pól, pozostawiam dodatkowe informacje, takie jak **author** (autor artykułu), **publishedAt** (data publikacji), **source\_name** (nazwa źródła), **url** (link do artykułu) oraz **topic** (tematyka artykułu), które są zachowane w oryginalnej, niezwektoryzowanej formie. Te metadane, pochodzące z wcześniejszego procesu ekstrakcji, są istotne dla zapewnienia użytkownikowi pełnego obrazu kontekstu artykułu, jego źródła oraz szczegółów publikacji.

### 3.3.2. Ostateczna struktura przetworzonych danych

Po zakończeniu procesu transformacji, każda jednostka danych zawiera zarówno wygenerowane wektory kluczowych pól, jak i oryginalne metadane, które nie zostały zwektoryzowane. Ostateczna struktura każdego artykułu obejmuje:

* **title** – tytuł artykułu,
* **description** – opis artykułu,
* **content** – pełną treść artykułu,
* **title\_vector** – wektor tytułu,
* **description\_vector** – wektor opisu,
* **content\_vector** – wektor pełnej treści,
* **author** – autora artykułu
* **publishedAt** – datę publikacji
* **source\_name** – nazwę źródła
* **url** – bezpośredni link do artykułu
* **topic** – temat artykułu

Ważne jest, aby oryginalne, niezwektoryzowane pola zostały zachowane, ponieważ model LLM, który generuje odpowiedzi, nie wykorzystuje wektorów jako kontekstu. Zamiast tego opiera się na faktycznej, tekstowej reprezentacji artykułu. Wektory służą wyłącznie do wyszukiwania semantycznie powiązanych treści, natomiast rzeczywista treść tekstowa dostarcza kontekstu dla odpowiedzi generowanych przez model. Po zakończeniu tego etapu dane są gotowe do przechowywania w wektorowej bazie danych,

## 3.4. Wektorowa baza danych

W systemie Retrieval Augmented-Generation (RAG) jednym z kluczowych elementów jest indeksowanie danych w sposób, który umożliwia szybkie i efektywne wyszukiwanie treści. W tym projekcie zdecydowano się na bazę Elasticsearch, ponieważ łączy ona tradycyjne wyszukiwanie pełnotekstowe z zaawansowanym wyszukiwaniem semantycznym opartym na wektorach. Proces indeksowania w Elasticsearch to nie tylko wgranie danych, ale także ich analiza, strukturyzacja i optymalizacja pod kątem wyszukiwania, co zapewnia wysoką precyzję i wydajność w odnajdywaniu treści dopasowanych do zapytania.

### 3.4.1. Indeksowanie w Elasticsearch

Indeksowanie to proces, w którym dane są przetwarzane i zapisywane w sposób umożliwiający ich szybkie odnajdywanie. W Elasticsearch indeks można porównać do tabeli w tradycyjnych bazach danych, jednak jego struktura i funkcjonalność są bardziej zaawansowane. Każdy indeks tworzy tzw. indeks odwrócony (ang. inverted index), który pozwala na szybkie wyszukiwanie dokumentów przez analizę treści.

Indeks odwrócony przypisuje każde słowo do dokumentów, w których się ono pojawia, co pozwala szybko zidentyfikować dokumenty zawierające dane słowo lub frazę, zamiast przeszukiwać każdy dokument od początku do końca. Dzięki temu możliwe jest wyszukiwanie pełnotekstowe oraz wyszukiwanie semantyczne, które pozwala dopasowywać dokumenty na podstawie ich znaczenia. W przypadku wyszukiwania semantycznego Elasticsearch wykorzystuje reprezentacje wektorowe, które oddają znaczenie treści i pozwalają na dopasowanie dokumentów według podobieństwa.

### 3.4.2. Mapping – struktura i organizacja indeksu

Mapping to schemat, który określa strukturę danych w indeksie. Można go porównać do struktury tabeli w tradycyjnych bazach danych. Mapping definiuje, jakie pola będą przechowywane w dokumentach, jakiego będą typu oraz jak powinny być przetwarzane. Dzięki mappingowi Elasticsearch „rozumie”, jakie dane znajdują się w poszczególnych polach dokumentu i w jaki sposób powinny być analizowane oraz indeksowane.

Mapping używany w projekcie obejmuje następujące kluczowe typy danych i ich konfigurację:

* **Typy danych**: Każde pole jest przypisane do określonego typu, co umożliwia optymalizację przechowywania i wyszukiwania:
  + text – dla pól, które mają być przeszukiwane pełnotekstowo, takich jak title, description i content. Tekst jest przekształcany na mniejsze jednostki (tokeny), co pozwala na wyszukiwanie na poziomie słów i fraz.
  + keyword – dla pól, które przechowują dokładne wartości, np. source\_name, author, topic. Są one używane do filtrowania lub sortowania i nie podlegają dalszej analizie.
  + dense\_vector – dla pól wektorowych, takich jak title\_vector, description\_vector i content\_vector, które przechowują liczbową reprezentację znaczenia treści, kluczową w wyszukiwaniu semantycznym.
  + date – np. data publikacji danego źródła informacji
* **Analizator treści**

Dla pól tekstowych (title, description, content) wykorzystano domyślny analizator Elasticsearch, który automatycznie przetwarza tekst, dzieląc go na tokeny (np. słowa lub krótsze wyrażenia). Analizator wykonuje kilka istotnych operacji:

* + **Tokenizacja** – dzieli tekst na mniejsze fragmenty, takie jak słowa, co umożliwia dokładne dopasowanie do zapytań.
  + **Normalizacja** – przekształca tekst na małe litery, co zapobiega błędnemu dopasowaniu wynikającemu z różnic w wielkości liter.
  + **Usuwanie znaków specjalnych** – eliminuje zbędne znaki, które nie wpływają na znaczenie tekstu, co upraszcza proces wyszukiwania.

Analizator sprawia, że Elasticsearch efektywnie przeszukuje dokumenty na podstawie pojedynczych słów lub fraz, zapewniając przy tym elastyczność w dopasowywaniu treści.

* **Pola wektorowe – konfiguracja i optymalizacja**

Pola wektorowe (title\_vector, description\_vector, content\_vector) odgrywają kluczową rolę w wyszukiwaniu semantycznym, ponieważ reprezentują znaczenie treści w formie liczb. W projekcie każde z tych pól przechowuje wektory o 384 wymiarach, które zostały wygenerowane na etapie przetwarzania tekstu. Liczba wymiarów odpowiada liczbie cech wyodrębnionych przez model embeddingowy, co pozwala na dokładne uchwycenie kontekstu i znaczenia treści w dużej przestrzeni wektorowej.

W polach wektorowych zastosowano następujące konfiguracje:

* + **Typ dense\_vector** – Pozwala na przechowywanie liczbowych reprezentacji znaczenia treści, które mogą być porównywane w procesie wyszukiwania.
  + **Indeksowanie (index: true)** – Indeksowanie jest niezbędne, aby umożliwić szybkie porównanie wektorów podczas wyszukiwania semantycznego. Dzięki temu Elasticsearch może przetwarzać zapytania w czasie rzeczywistym.
  + **Metryka podobieństwa (similarity: "cosine")** – Wybór metryki kosinusowej do porównywania wektorów umożliwia dopasowanie dokumentów na podstawie kąta między wektorami, co pozwala na znalezienie najbardziej podobnych semantycznie treści.
  + **Opcje indeksowania HNSW (index\_options)** – Pola wektorowe są zoptymalizowane do wyszukiwania z użyciem algorytmu HNSW (Hierarchical Navigable Small World), który zapewnia wydajne odnajdywanie najbliższych sąsiadów w dużych zbiorach danych:
  + **Parametr m: 16** – Określa liczbę połączeń (krawędzi) w grafie HNSW, co pozwala na znalezienie równowagi między szybkością wyszukiwania a precyzją.
  + **Parametr ef\_construction: 100** – Kontroluje dokładność indeksu, co pozwala na optymalizację szybkości i efektywności wyszukiwania.

Podsumowanie struktury mappingu

Poniżej przedstawiono strukturę mappingu zastosowaną w indeksie:

* **source\_name** – typ keyword, przechowuje nazwę źródła publikacji.
* **author** – typ keyword, przechowuje nazwisko autora artykułu.
* **title** – typ text, przeznaczony do wyszukiwania pełnotekstowego.
* **title\_vector** – typ dense\_vector, przechowuje reprezentację wektorową o 384 wymiarach.
* **description** – typ text, przeznaczony do wyszukiwania pełnotekstowego.
* **description\_vector** – typ dense\_vector, przechowuje reprezentację wektorową o 384 wymiarach.
* **content** – typ text, przeznaczony do wyszukiwania pełnotekstowego.
* **content\_vector** – typ dense\_vector, przechowuje reprezentację wektorową o 384 wymiarach.
* **url** – typ keyword, przechowuje link do artykułu, nieindeksowany do wyszukiwania.
* **publishedAt** – typ date, przechowuje datę publikacji.
* **topic** – typ keyword, przechowuje temat artykułu.

Ta konfiguracja zapewnia odpowiednią strukturę danych, precyzję i wydajność, umożliwiając skuteczne wyszukiwanie informacji zarówno na podstawie treści, jak i podobieństwa semantycznego.

## 3.5. Wgranie przetworzonych danych

Ostatnim elementem inżynierii danych w projekcie jest proces wgrania przetworzonych danych do bazy Elasticsearch, który odbywa się za pomocą techniki **ETL**. Skrót ETL oznacza trzy główne etapy przetwarzania danych: **Extract** (ekstrakcja danych), **Transform** (transformacja), i **Load** (wgranie). W ETL dane są najpierw pobierane ze źródła, następnie przekształcane do odpowiedniej formy, a na końcu ładowane do docelowego systemu, gdzie mogą być efektywnie przechowywane i przeszukiwane. W projekcie poszczególne kroki ETL są realizowane przez dedykowane moduły w Pythonie, które automatyzują przetwarzanie artykułów pobranych z News API.

### 3.5.1. Moduł pipeline.py – Koordynacja Procesu ETL

Moduł pipeline.py pełni rolę głównego koordynatora całego procesu ETL, łącząc funkcje odpowiedzialne za ekstrakcję, transformację oraz wgranie przetworzonych dokumentów do indeksu Elasticsearch. Moduł ten zawiera dwie główne funkcje: run\_etl oraz run\_etl\_update, które sterują operacjami związanymi z wgrywaniem danych i ich aktualizacją.

Funkcja run\_etl obsługuje pełny proces ETL od pobrania artykułów, aż po ich załadowanie do indeksu w Elasticsearch:

* **Ekstrakcja danych**: Funkcja recent\_week\_etl (moduł extract.py) pobiera artykuły z ostatniego tygodnia z News API, wykorzystując wcześniej zdefiniowane tematy technologiczne.
* **Transformacja danych**: Po ekstrakcji, każdy artykuł jest przekształcany przy pomocy funkcji transform\_data (moduł transform.py). Funkcja ta selekcjonuje kluczowe pola, tworzy ich wektory oraz dodaje odpowiednie metadane.
* **Wgranie danych**: Po przekształceniu, przetworzone dokumenty są przekazywane do bulk\_load\_documents (moduł elastic\_utils.py), która ładuje dane do indeksu Elasticsearch, umożliwiając ich dalsze przeszukiwanie.

Funkcja run\_etl\_update

Funkcja run\_etl\_update jest wariantem funkcji run\_etl, rozszerzonym o możliwość usunięcia wcześniejszych dokumentów z indeksu przed wgraniem nowych danych. Funkcja ta może być używana w przypadku aktualizacji danych:

1. **Czyszczenie indeksu**: Przed ponownym załadowaniem danych funkcja usuwa wszystkie dokumenty w indeksie, co zapewnia brak duplikacji i dostosowanie zawartości indeksu do bieżących danych.
2. **Aktualizacja ETL**: Po wyczyszczeniu indeksu, funkcja ponownie uruchamia run\_etl, wykonując wszystkie etapy ETL dla najnowszych artykułów.

Funkcja bulk\_load\_documents – Wgranie Danych do Elasticsearch

Zasadnicze wgranie danych realizuje funkcja bulk\_load\_documents, znajdująca się w module **elastic\_utils.py**. Funkcja ta odpowiada za zbiorcze załadowanie wszystkich przetworzonych dokumentów do wskazanego indeksu Elasticsearch:

1. **Przygotowanie dokumentów**: Funkcja bulk\_load\_documents organizuje przetworzone artykuły w formie akcji, gdzie każdy dokument jest przypisany do określonego indeksu i przygotowany do zbiorczego wgrania.
2. **Zbiorcze ładowanie danych**: W celu efektywności i optymalizacji działania, funkcja wykorzystuje metodę bulk z biblioteki helpers dla Elasticsearch. Ta metoda umożliwia jednorazowe załadowanie wielu dokumentów, co zmniejsza liczbę pojedynczych operacji przesyłania danych i przyspiesza proces wgrania.
3. **Obsługa błędów**: Dzięki zastosowaniu blokady try-except, funkcja monitoruje wszelkie błędy występujące podczas operacji wgrywania, takie jak problemy z połączeniem czy błędy transportowe, zapewniając stabilność działania całego procesu.

### 3.5.2. Struktura Dokumentów Wgrywanych do Elasticsearch

Przetworzone dokumenty ładowane do Elasticsearch posiadają zdefiniowaną strukturę, zawierającą zarówno pola tekstowe, jak i pola wektorowe:

* **Metadane artykułu**:
  + source\_name: Nazwa źródła, np. „Wired”.
  + author: Autor artykułu.
  + title, description, content: Odpowiednio tytuł, opis i pełna treść artykułu.
  + url: Link do artykułu.
  + publishedAt: Data publikacji artykułu.
  + topic: Tematyka artykułu (np. „blockchain”), wskazująca, z jakiego obszaru pochodzi artykuł.
* **Pola wektorowe**:
  + title\_vector, description\_vector, content\_vector: Wektory reprezentujące semantyczne znaczenie tytułu, opisu oraz pełnej treści artykułu. Wektory te mają 384 wymiary, a ich konfiguracja umożliwia wyszukiwanie semantyczne oparte na podobieństwie kosinusowym.

Wszystkie przetworzone dokumenty po wgraniu do indeksu Elasticsearch są gotowe do dalszych operacji wyszukiwania i analizy w systemie RAG, wspierając zarówno klasyczne wyszukiwanie pełnotekstowe, jak i zaawansowane wyszukiwanie semantyczne oparte na wektorach.

## 3.6. Cykl życia danych w projekcie

W projekcie zastosowano cotygodniowy cykl aktualizacji danych, aby zapewnić dostęp do najnowszych informacji ze świata technologii. Stałe odnawianie danych pozwala systemowi generować odpowiedzi na podstawie aktualnych wydarzeń i trendów, co jest kluczowe w kontekście dynamicznie zmieniających się branż, takich jak technologia i IT. Proces ten jest możliwy dzięki regularnie uruchamianemu, zautomatyzowanemu przepływowi ETL (Extract, Transform, Load).

Po wdrożeniu systemu w środowisku chmurowym cykl ETL będzie działał automatycznie, zgodnie z harmonogramem ustalonym dla aktualizacji. System samodzielnie:

1. **Usuwa stare dane**: Przed załadowaniem nowych danych, system czyści indeks, usuwając starsze artykuły, co zapewnia, że tylko aktualne informacje są przechowywane i dostępne do wyszukiwania.
2. **Ekstrakcja**: Pobiera nowe artykuły z News API dla określonych tematów technologicznych.
3. **Transformacja**: Przetwarza te artykuły, tworząc dla nich odpowiednie wektory oraz dodając metadane.
4. **Ładowanie**: Wgrywa zaktualizowane dokumenty do bazy Elasticsearch, zastępując stare dane.

Tak zorganizowany cykl życia danych w projekcie nie wymaga ingerencji manualnej, a system pozostaje na bieżąco dzięki automatycznym aktualizacjom danych.