2023

Talk To Your Data - SQLamur

DATA SCIENCE CLUB PJATK

JACEK JACKOWSKI, PATRYK OWCZARZ, ADRIAN KORDAS, FILIP DZIECIOL, IGOR WINEK

Spis treści

Cel projektu

Celem wyzwania zaprezentowanego przez Ministerstwo Finansów oraz Krajową Administrację Skarbową jest umożliwienie bezpośredniego odpytywania bazy danych przez osoby niezwiązane z informatyką, które na codzień muszą obsługiwać wiele zgłoszeń. Benefitem takiego rozwiązania byłoby zmniejszenie czasu oczekiwania na odpowiedź ze strony urzędów, połączonym z jednoczesnym odciążeniem innych stanowisk technicznych w organizacji, co przełoży się na potencjalny wzrost efektywności pracy urzędników.

Poniżej jest pełny spis wymagań oraz informacja o tym, czy zostały spełnione, częściowo, czy nie.

✅ Wymaganie spełnione ☑️ Wymaganie częściowo spełnione ❌ Wymaganie niespełnione

Wymagania obowiązkowe:

✅ Aplikacja musi bazować na rozwiązaniach, których dalsze wykorzystanie nie będzie wiązało się z ponoszeniem dodatkowych kosztów licencyjnych. Zalecane jest korzystanie z rozwiązań typu open-source.

✅ Aplikacja musi być dostosowana do różnych struktur danych, w tym posiadać możliwość zdefiniowania schematu struktury danych.

✅ Definiowanie schematu struktur danych w aplikacji odbywa się za pomocą instrukcji DDL (dla dowolnej bazy danych: SQLite/Postgres itd.) lub poprzez wskazanie nazwy schematu i pobranie struktur z uruchomionej bazy danych.

✅ Aplikacja musi pozwalać na odpytywanie bazy danych za pomocą zapytań przetłumaczonych z języka naturalnego (angielskiego i/lub polskiego) na SQL.

✅ Aplikacja przetwarza zapytanie języka naturalnego na SQL, który może być modyfikowany przed jego wykonaniem.

✅ Aplikacja dostarcza odpowiedź w postaci wyników zapytania wraz z zapytaniem SQL.

✅ Aplikacja zapewnia ciągłość zapytań i/lub umożliwia odwoływanie się do wcześniejszych zapytań oraz ich wyników.

✅ Aplikacja musi być możliwa do uruchomienia na środowiskach typu on-premise (lokalnie, bez dostępu do sieci w trakcie przetwarzania).

✅ Aplikacja posiada interfejs w języku angielskim i/lub polskim.

✅ Do aplikacji musi zostać dołączona instrukcja instalacji pozwalająca na samodzielne uruchomienie wraz z informacją o wykorzystanym modelu, zależnościach i ich źródłach.

Wymagania dodatkowe:

✅ Aplikacja umożliwia pracę dla użytkownika polskojęzycznego tj. posiada interfejs w języku polskim i pozwala tworzyć zapytania w języku polskim.

✅ Interfejs aplikacji jest ergonomiczny i estetyczny wizualnie.

✅ Przygotowanie propozycji uruchomienia aplikacji na większą skalę (dla większej liczby użytkowników).

✅ Przekazanie aplikacji za pośrednictwem rozwiązania kontenerów docker (np. dockerfile, docker-compose).

Proponowane rozwiązanie

Koło Naukowe Data Science Club działające z ramienia Polsko-Japońskiej Akademii Technik Komputerowych w Warszawie proponuje rozwiązanie wykorzystujące złożone modele językowe (eng. LLM – Large Language Models), które ostatnimi czasy rozwinęły się tak bardzo, że ich udane implementacje można zauważyć w każdej sferze życia – od obsługi Klienta, przez pomoce naukowe, aż po wsparcie programistów.

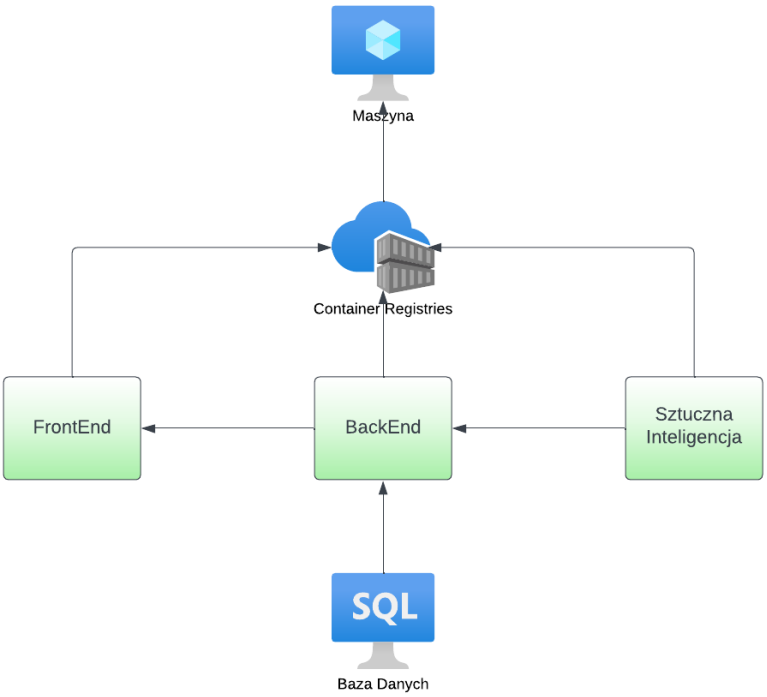
Wykorzystanie takiego modelu wiąże się z potrzebami dużych mocy obliczeniowych, zazwyczaj dostępnych tylko w środowiskach chmurowych, jednak istnieje wiele rozwiązań o otwartym kodzie źródłowym (eng. Open Source), które wcześniej zostały wytrenowane przez duże firmy i udostępnione w celu szerzenia sztucznej inteligencji. Użycie komercyjne jest w takim przypadku dozwolone i uważamy, że może być niezwykle pomocne w takim rozwiązaniu.

Istnieje wiele takich modeli na rynku, m.in. Llama2 (Facebook/Meta), Falcon lub Vanna. Potrzebują one jednak pomocy programistycznej, aby wykonywać swoje zadania z dużą prędkością oraz jakością, dlatego też czynniki te należy dobierać do aktualnie przeznaczonego budżetu oraz wymagań biznesowych. Dodatkowo sam model jest tylko cześcią produktu AI – wymagane jest ich odpowiednia integracja, wdrożenie, monitoring oraz ostatecznie opakowanie w pełnoprawną aplikację z czytelnym interfejsem, który zachęci użytkowników końcowych do korzystania z niej w swojej codziennej pracy.

W naszym przypadku proponowane rozwiązanie wykorzystuje 2 modele: Orca działający on-prem jako Open Source, oraz Vanna, który dzięki połączeniu się z relacyjną bazą danych jest w stanie zrozumieć jej strukturę oraz dane, a nastepnie odpowiedzieć na większość zadanych pytań wprowadzonych przez osoby nietechniczne. Wdrożenie ze strony technicznej wymaga:

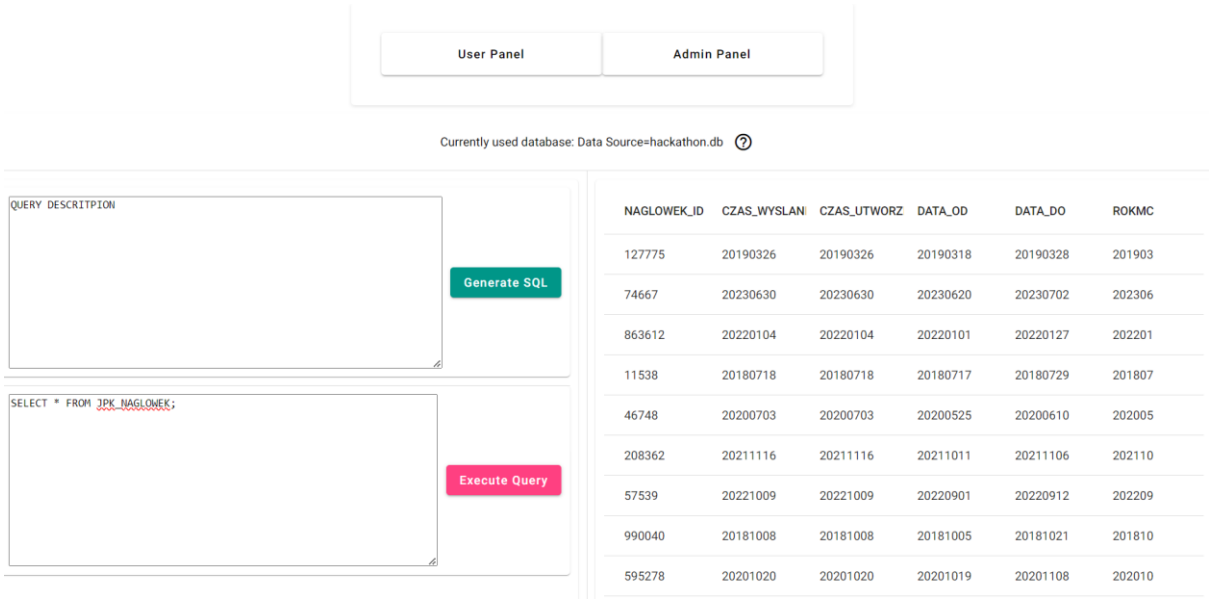
- Użycia aplikacji .NET (backend)
- Użycia aplikacji Angular (frontend)
- Użycia Flask (praca z modelem)
- Użycia bazy danych (dowolna relacyjna, w tym przypadku SQLite)
- Oprogramowania wspomagającego wdrożenia, tj. Docker, Docker-Compose
- Oprogramowania wspierającego programistów tj. Azure Pipelines oraz GitHub

Poniżej przedstawiona jest architektura stworzonego oprogramowania.



Ostatecznie otrzymujemy 3 kontenery komunikujące się ze sobą, zintegrowane poprzez API, wdrażane przez CI/CD na maszynie on-prem, z bazą danych SQLite, przy czym model może być dowolnym Produktem niezależnym od stylu wdrożenia, potrzebuje on jednak dosyć duży mocy obliczeniowych.

Cała aplikacja jest dostępna dla użytkownika końcowego przez elegancki i minimalistyczny interfejs.



Sztuczna inteligencja

Produkt posiada możliwość wykorzystania 2 modeli:

1. Orca Mini 13b – całkowicie prywatny model o 13 miliardach parametrów dostępny w wersji Open Source.
2. Vanna – model zapewniający o prywatności danych, które procesowane są na bazie Snowflake dostawcy modelu.

Model nr 1 został wybrany jako najlepszy dla aktualnych zasobów obliczeniowych, budżetowych oraz czasowych. Model nr 2 nie spełnia tylko wymogu bycia on-prem, ponieważ inferencja odbywa się po stronie dostawcy, jednak jest wyspecjalizowany w pracy na bazie danych, której tylko metadane są przetwarzane, dlatego też model ten może być łatwiejszy w użyciu, ale przede wszystkim posiada dużo wyższą jakość od Orca. Dodatkowo Vanna jest dotrenowywany lokalnie na dostarczonej bazie danych, dzięki czemu ma niemalże perfekcyjną jakość.

Model nr 1 jest użyty podstawowo jako główne wymaganie wyzwania, jednak model nr 2 można włączyć poprzez interfejs użytkownika za jego zgodą. Poniżej znajduje się krótkie porównanie obu modeli.

	ORCA MINI 13B	VANNA
ROZMIAR MODELU	7,5 GB	Nieznany
SPECJALIZACJA	Przetwarzanie tekstu	Przetwarzanie baz danych
CENA	Darmowy, Open Source	Darmowy, Open Source
SPOSÓB WYKORZYSTANIA	Pobrany model	Inferencja na serwerach dostawcy lub własne pobranie i wytrenowanie od zera
BEZPIECZEŃSTWO DANYCH	Kompletne – model w pełni lokalny	Zapewnione przez dostawcę – przetwarza tylko strukturę bazy, ale nie dane
JAKOŚĆ	Dostateczna	Bardzo wysoka
CZAS INFERENCJI	Kilkadziesiąt sekund	Kilka sekund
MOŻLIWOŚĆ DOTRENOWANIA	Do własnej implementacji	Zaimplementowana
TŁUMACZENIE Z I NA ANG	Google Translate API	Zaimplementowana

Model Vanna posiada taką przewagę, że bardzo łatwo jest go dotrenować na dostarczonej bazie danych, jej strukturze, ale także częstych zapytaniach, co może znacząco zwiększyć jego jakość i stworzyć kulturę reużywalności technicznej, która idzie w parze z kulturą konteneryzacji.

Instalacja produktu

Całość składa się z jasno separowalnych modułów, dzięki czemu mogą być one rozwijane i zarządzane niezależnie, wiąże się to jednak z większym trudnem administracyjnym, dlatego też niezbędne jest odpowiednie wyszkolenie zespołu utrzymującego rozwiązanie.

Środowisko Programistyczne

Kod umiejscowiony jest w systemie wersjonowania kodu – GitHub:

- Kod ML: https://github.com/PJWSTK-Data-Science-Dojo/hackyeah2023-ml
- Kod Backend: https://github.com/PJWSTK-Data-Science-Dojo/hackyeah2023-api
- Kod Frontend: https://github.com/PJWSTK-Data-Science-Dojo/hackyeah2023-ui
- Kod DevOps: https://github.com/PJWSTK-Data-Science-Dojo/hackyeah2023-devops

Repozytoria zostaną opublikowane po czasie oddania wszystkich projektów HackYeah 2023.

BackEnd oraz FrontEnd postawione są w środowisku skonteneryzowanym korzystającym z Azure Container Registry (ACR), jednak można też użyć innego systemu. Azure ACR został wybrany ze względu na prostą integrację z Azure Pipelines oraz Azure DevOps, które służą odpowiednio do szybkiego i niezawodnego wdrażania oraz zarzadzania rozwiązaniem i jego rozwojem, przy czym oba serwisy są darmowe.

W przypadku modelu nie są wymagane zewnętrzne zasoby ze względu na prostotę bazy danych. W dłuższym jednak okresie oraz na produkcyjnych hurtowniach jego regularne dotrenowywanie, skalowanie oraz monitoring techniczny będą kluczowymi elementami architektury, które powinny korzystać z najlepszych praktyk MLOps.

Instrukcja nanoszenia zmian w kodzie źródłowym:

1. Należy uzyskać dostęp do wyżej wymienionych repozytoriów oraz pobrać kod poprzez git clone
2. Każda zmiana w kodzie wypchnięta na repozytorium kodu wywołuje potok w Azure Pipelines, który buduje obraz dockera, wykonuje testy oraz, jeśli wszystko przeszło bez błędów, wrzuca nowy obraz na Azure Container Registry, z którego można wdrażać zmiany.

Wdrożenia

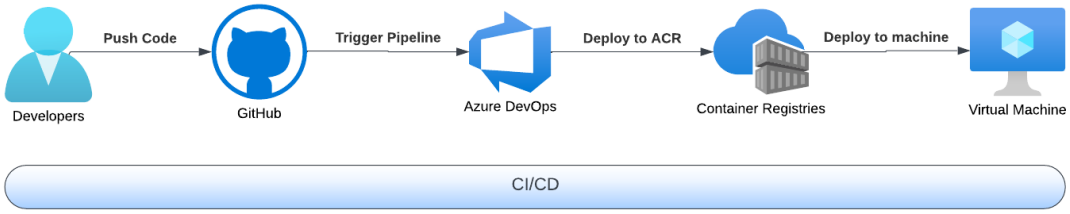
Bez względu na CI/CD działające na bazie Azure Pipelines, potrzebujemy wdrożyć każdy z modułów: Backend, FrontEnd oraz ML. Każdy jest oddzielnym kontenerem, który może być uruchomiony na dowolnym środowisku tj. DEV, TEST oraz PROD. Dzięki zamknięciu wszystkich zależności i wystawieniu interfejsów API, ich integracja oraz wdrożenie może odbywać się niemalże automatycznie, przy założeniu odpowiednich testów. Jest też ona opakowana w Docker-Compose, aby ułatwić wdrożenie i zarządzanie.

Kwestie bazy danych mogą być rozwiązane dowolnie tak długo, jak istnieje komunikacja pomiędzy Produktem, a bazą, przy czym baza nie powinna być w kontenerze ze względu na ulotność jej danych (ewentualnie można użyc zewnętrznych wolumenów).

Wymagania dotyczące poszczególnych modułów:

- Kontener Backendowy - .NET 6+
- Kontener Frontendowy – Angular 16+
- Kontener ML – Python 3.9+, biblioteki gpt4all, vanna, flask

Całość wdrożenia polega na ustawieniu środowisk oraz uruchomieniu potoku w Azure Pipelines.



Instrukcja pełnego uruchomienia:

1. Należy zainstalować oprogramowanie Docker
2. Należy pobrać obrazy Dockerowe z Azure Container Registry

```
docker login -u hack-yeah -p <password> pexoacr.azurecr.io
docker pull “pexoacr.azurecr.io/hackyeah-ui:latest”
docker pull “pexoacr.azurecr.io/hackyeah-api:latest”
docker pull “pexoacr.azurecr.io/hackyeah-ml:latest”
```

3. Należy uruchomić kontenery z pobranych obrazów na jeden z dwóch poniższych sposobów
    a. Uruchomienie całego zestawu modułów na jeden z dwóch poniższych sposobów
        i. Z lokalnych repozytoriów
        ii. Poprzez Docker Compose (wymaga instalacji docker-compose)

```
docker-compose -f .\docker-compose-local.yml up
```

    b. Uruchomienie kontenerów pojedyczno

```
docker run -d --name "hackyeah-api" -p 5012:80 pexoacr.azurecr.io/hackyeah-api
docker run -d --name "hackyeah-ui" -p 4200:80 pexoacr.azurecr.io/hackyeah-ui
docker run -d --name "hackyeah-ml" -p 7000:105 pexoacr.azurecr.io/hackyeah-ml
```

Konfiguracja środowisk

Głównym wymaganiem będzie tu zainstalowane środowisko Docker postawione na systemie Windows lub Unix. Dodatkowo, jeśli chcemy zapewnić orkiestrację oraz większa automatyzację, możemy wykorzystać Kubernetes, jednak nie jest to wymagane i wymagałoby to dalszych przygotowań.

Ze względu na duże moce obliczeniowe wymagane dla modułu ML będzie to najcięższa część aplikacji i może wymagać osobnej warstwy skalowania. Konteneryzacja daje bardzo dużą skalowalność horyzontalną, jednak w przypadku tego typu uczenia i inferencji, może być wymagana bardzo duża maszyna z solidnymi zasobami GPU oraz pamięci. Budżet w uczeniu maszynowym, a zwłaszcza w modelach LLM, w głównej ilości jest spożytkowany przez moc obliczeniową.

Jedną ze zmian, które nalezy zrobić to podmiana ścieżki do modelu ORCA w 2 plikach repozytorium DevOps:

- docker-compose-localyml → services.volumes
- complete-setup-from-acr.yml → services.volumes

Konfiguracja modelów podlega również odpowiedniemu tuningowi. W przypadku tego produktu konfiguracja Vanna jest po stronie dostawcy, jednak Orca posiada swoją w repozytorium ML:

- llm/config/config.yaml

Plany na dalszy rozwój

Umożliwienie łączność z dowolną bazą danych

Jednym z głównych problemów uczenia maszynowego jest dostęp do danych. W przypadku tego Produktu mamy otrzymać dostęp do hurtowni danych, struktura której ma być wyuczona. W teorii model jest w stanie dotrenować się na dowolnej strukturze, zwłaszcza taki wyspecjalizowany w procesowaniu zapytań SQL, dzięki czemu jedną z potencjalnych funkcjonalności jest podanie połączenia (eng. Connection String) do dowolnej bazy danych lub podanie pliku z definicją struktury dowolnej bazy, dzięki czemu można uzyskać pełną niezależność od typu i systemu bazy danych.

Model LLM

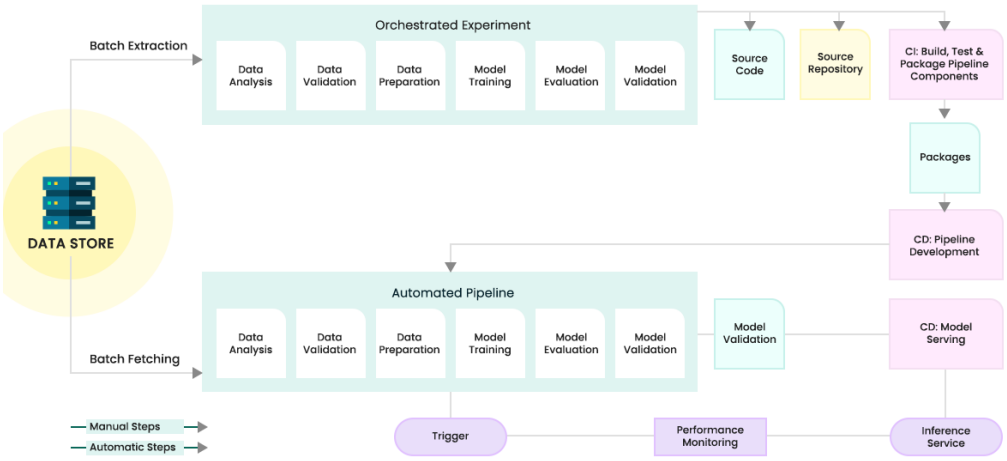
Modele używane w środowiskach on-prem bardzo często są ograniczone do zasobów organizacji, stąd prawdopodobnie najlepszym sposobem byłoby wykorzystanie zewnętrznych modeli postawionych na ogromnych klastrach obliczeniowych. Duża ich część pobiera i składuje wysyłane dane, jednak są też takie, które zapewniają prywatność przesyłanych informacji, tj. PrivateGPT, Vanna lub Azure OpenAI Service.

Środowisko uruchomieniowe

Największym ograniczeniem aktualnego rozwiązania, pod kątem jakościowym i wydajnościowym, jest moc obliczeniowa wykorzystywana przy każdej inferencji z modelem. Niestety przetwarzanie tekstu, zwłaszcza z długim zachowaniem kontekstu, jest bardzo zasobochłonne, stąd rekomendowane jest wykorzystanie dużych maszych wielkości setek GB RAM oraz wielu GPU.

Zapewnienie skalowalności i niezawodności poprzez MLOps

MLOps jest zestawem dobrych praktyk dla pracy nad sztuczną inteligencją w dużej skali. Zgodnie z nimi należy zapewnić możliwość ciągłego dotrenowywania (automatycznie oraz na żądanie), wykorzystanie procesów CI/CD oraz odpowiedniego monitorowania. Ponadto każdy moduł powinien być jasno separowalny z możliwością skalowania horyzontalnego oraz wertykalnego. Część z tego procesu już działa, dlatego też może być to jedno z pierwszych ulepszeń Produktu.



Rysunek 1: [MLOps: A set of essential practices for scaling ML-powered applications]

Personalizacja produktu pod użytkownika

Ze względu na złożoność zadań wykonywanych przez ludzi, modele potrzebują pomocy pod kątem tego, w jakim kierunku mają odpowiadać. Działa to podobnie do obsługi klienta, gdzie obsługujący powinien specjalizować się w zgłaszanym typie zgłoszeń, w zależności od poziomu linii wsparcia.

Większe modele potrafią trzymać kontekst, nawet do kilku tysięcy słów wstecz i na bazie tego potrafią coraz lepiej na bieżąco poznawać tego, z kim rozmawia. Wymaga to jednak znacząco większych mocy obliczeniowych, oraz odpowiednich technik modelowania tekstu, tj. Prompt Engineering oraz analiza behawioralna. Jest to dużo bardziej złożony proces, który musi być stworzony na bazie person oraz możliwości modeli LLM.

Zwiększenie jakości i bezpieczeństwa rozwiązania

Wykorzystanie narzędzi AI zawsze wiąże się z niechęcią do ich wykorzystania, nie tylko ze względu na świadomość sztuczności, ale przede wszystkim przez zagrożenia idące za takimi rozwiązaniami.

Przede wszystkim chodzi tu o szkodliwe treści, które mogą być potencjalnie wygenerowane, kiedy atakujący w jakiś sposób niebezpiecznie zasili model swoimi danymi – należy przemyśleć takie zagrożenia, oraz jak im zapobiegać.

Kolejnym problemem jest metryka jakości modelu – może to być przykładowo % zapytań, które zwróciły akceptowalny poziom odpowiedzi. Akceptowalność może być ustalona przez osoby techniczne, większe modele, lub przez samych użytkowników końcowych (np. przez mierzenie zadowolenia).