Politechnika Warszawska
Wydział Elektryczny

Raport z pierwszej części projektu

Autorzy:
Aleksei Haidukevich, nr albumu 295233
Jakub Korczakowski, nr albumu 291079
Marharyta Kruk, nr albumu 295235
Maciej Leszczyński, nr albumu 291085
Piotr Rosa, nr albumu 291112

30 stycznia 2020

Spis treści

1 Etap 1 - Zbieranie danych

czas: 4.11.2019 - 15.12.2019

1.1 Opis

W pierwszym etapie projektu najważniejszym celem było utworzenie narzędzi do zbierania danych z Twittera i Reddita. Zostaną one użyte w celu stworzenia zbioru danych pozwalającego na naukę i testowanie algorytmów. Pobrane wiadomości i nagłówki będą składowane w bazie danych. Ten etap prac zawierał również przygotowanie infrastruktury zdolnej analizować i przechowywać zebrane dane.

1.2 Zadania

1. **Analiza dostępności API mediów społecznościowych.**
   Po przeanalizowaniu API zdecydowaliśmy się wykorzystać media społecznościowe **Twitter** i **Facebook**. API Twittera jest bardzo rozwinięte i pozwala na swobodny dostęp do tweetów, nawet w przypadku darmowej wersji konta. Zdecydowaliśmy się pobierać tweety wykorzystując bibliotekę Pythona - **Tweepy**. **COŚ O FACEBBOOKU? COŚ O FACEBBOOKU? COŚ O FACEBBOOKU? COŚ O FACEBBOOKU? COŚ O FACEBBOOKU? COŚ O FACEBBOOKU?**

2. **Analiza i wybór sposobu wdrożenia aplikacji (rozwiązania chmurowe).**
   Po analizie potrzeb naszego projektu zdecydowaliśmy się wykorzystać rozwiązanie chmurowe, ponieważ pozwalają one na zapewnienie dostępu do zasobów projektowych(takich jak bazy danych czy maszyny wirtualne) dla wszystkich pracujących nad projektem.

   Wśród dostawców infrastruktury chmurowej można wyróżnić dwie firmy, które oferują darmowe środki dla studentów. Są to Amazon (AWS) oraz Microsoft (Azure). Z powodu mniejszych możliwości oraz mniejszej ilości środków dostępnych na patformie AWS zdecydowaliśmy się wybrać platformę Azure.

   W obrębie tej platformy oprócz klasycznych maszyn wirtualnych dostępne są rozwiązania docelowe przeznaczone do przetwarzania dużych ilości danych, należą do nich:

   - HDInsight,
   - Azure Databricks,
   - Azure Data Lake Services.

Pomimo tego, że te usługi znacznie ułatwają budowę projektów nie zdecydowaliśmy się na ich użycie ze względu na wysoką cenę. Planujemy używać maszyn wirtualnych i za pomocą Dockera, a w przyszłości Kubertenesa zbudować infrastrukturę projektu.

3. **Analiza dostępnej infrasturktury do przetwarzania danych.** W docelowej aplikacji przetwarzającej dane w czasie rzeczywistym planujemy użyć:

   Apache Kafka do pobierania tweetów w czasie rzeczywistym,

   Apache Spark do przetwarzania danych i uruchamiania modeli,

   MS SQL do przechowywania tweetów.

   Elementy architektury aplikacji prawdopodobnie ulegną jeszcze zmianie podczas dalszego rozwoju projektu.

4. **Wybór odpowiedniej bazy do składowanych danych (porównanie SQL i noSQL).**
   Do składowania historycznych danych w naszym projekcie będziemy wykorzystywać bazę SQL ze względu na szybkość dostępu do danych.

5. **Instalacja wybranej bazy.**
   Pobrane tweety składujemy w bazie SQL znajdującej się na platformie Azure. Obecnie bazy danych pomieścić może do 2GB danych, jednak w przypadku gdy potrzebne nam będzie więcej przestrzeni możemy w każdej chwili zwiększyć pojemność bazy. Jest to jedna z zalet użycia chmury. Baza danych dostępna jest pod statycznym adresem IP, więc jest łatwo dostępna dla każdego. Konieczne jest jedynie dostosowanie zapory sieciowej w celu umożliwienia dostępu.

6. **Stworzenie programu pobierającego dane z Twittera.**
   Korzystając z biblioteki **Tweepy** udało nam się stworzyć skrypt pobierający wpisy z Twittera dotyczące wybranego słowa kluczowego. API Twittera pozwala na pobieranie naprawdę wielu szczegółów dotyczących wpisów, jednak do naszych celów nie potrzebujemy ich wszystkich. Zdecydowaliśmy się na pobieranie: ID tweeta, datę jego stworzenia, nazwę użytkownika oraz zawartość tweeta.
   Pobrane dane przechowujemy w tabelach, osobno dla każdej firmy, z dodatkową kolumną Sentyment, która mówi o tym, czy przesłanie wiadomości jest negatywne, czy pozytywne.
   Skrypt ten pobiera dane w interwałach czasowych, przy czym nie wszystkie tweety są zapisywane, a losowana jest jedynie część z nich. Wynika to z faktu, że często pojawia się więcej wpisów niż się spodziewaliśmy, a możemy nie poradzić sobie ze zbyt dużym zbiorem danych.

7. **Stworzenie programu pobierającego dane z Reddita.**

8. **Połączenie bazy danych z programami pobierającymi dane.**
   Serwer bazodanowy działa na platformie **Azure**. W przypadku danych z Twittera, łączenie z serwerem odbywa się przy wykorzystaniu biblioteki **pyodbc**. Serwer bazodanowy na Azure posiada statyczny adres, dlatego łączenie się z nią nie jest skomplikowane. W konfiguracji połączenie trzeba podać odpowiedni login, hasło, nazwę bazy danych oraz nazwę sterownika w naszym systemie, który zostanie wykorzystany do tego połączenia. Zazwyczaj sterownik ten jest już zainstalowany, wystarczy jedynie odnaleźć jego nazwę.

9. **Pobranie danych do bazy danych.**
   Baza danych z której korzystamy jest bazą SQL-ową. Po nawiązaniu połączenia, polecenia wykonywane są jak w przypadku korzystania z bazy danych lokalnie. Dane będą pobierane do bazy danych w czasie rzeczywistym za pomocą opracowanych programów.

10. **Analiza dostępnych zbiorów opisanych klasami, pozwalających na testowanie algorytmu.**

1.3 Kamienie milowe:

1. Utworzenie wyselekcjonowanego zbioru danych pozwalającego na naukę i testowanie algorytmów.

1.4 Parametry

1. Zbiór danych musi zawierać 250 tweetów, 150 nagłówków z Reddita zbieranych codziennie przez 30 dni dla 10 organizacji(razem 75000 tweetów i 45000 nagłówków).

2. Zbiór testowy musi zawierać 5000 tweetów i 3000 nagłówków, opisanych poprzez klasy pozwalające na sprawdzenie algorytmu.

1.5 Ryzyka:

1. Ograniczona dostępność API serwisów społecznościowych.
   Mitygacja: Automatyzacja zakładania kont dewloperskich, w celu ominięcia ograniczeń.

2. Brak zbioru pozwalającego na testowanie algorytmów analizujących sentyment.
   Mitygacja: Ręczne opisanie zbioru testowego.