

Politechnika Warszawska  
Wydział Elektryczny

---

SYSTEM DO ANALIZY SENTYMENTÓW SPÓŁKI  
NA PODSTAWIE WPISÓW W MEDIACH  
SPOŁECZNOŚCIOWYCH.

---

*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

12 czerwca 2020

---

## Spis treści

<b>1</b>	<b>Etap 1 - Zbieranie danych</b>	<b>4</b>
1.1	Opis . . . . .	4
1.2	Zadania . . . . .	4
1.2.1	Analiza dostępności API mediów społecznościowych .	4
1.2.2	Analiza dostępnej infrastruktury do przetwarzania danych. . . . .	4
1.2.3	Analiza i wybór sposobu wdrożenia aplikacji (rozwiązania chmurowe) . . . . .	4
1.2.4	Wybór odpowiedniej bazy do składowanych danych (porównanie SQL i noSQL) . . . . .	5
1.2.5	Stworzenie programu pobierającego dane z Twittera .	5
1.2.6	Stworzenie programu pobierającego dane z Facebooka	6
1.2.7	Instalacja wybranej bazy . . . . .	6
1.2.8	Połączenie bazy danych z programami pobierającymi dane . . . . .	6
1.2.9	Pobranie danych do bazy danych . . . . .	6
1.2.10	Analiza dostępnych zbiorów opisanych klasami, pozwalających na testowanie algorytmu . . . . .	6
1.3	Wykonanie zadań z pierwszego kamienia milowego . . . . .	7
<b>2</b>	<b>Etap 2 - Przetwarzanie danych</b>	<b>7</b>
2.1	Opis . . . . .	7
2.2	Zadania . . . . .	7
2.2.1	Budowa infrastruktury do przetwarzania danych . . .	7
2.2.2	Instalacja narzędzi do przetwarzania danych . . . . .	8
2.2.3	Baza Danych . . . . .	9
2.2.4	Potok przetwarzania danych w projekcie . . . . .	9
2.2.5	Udoskonalenie programu do pobierania danych z Twittera . . . . .	10
2.2.6	Stworzenie modelu do analizy sentymentów . . . . .	11
2.2.7	Stworzenie programu do pobierania danych z Facebooka	12
2.3	Wykonanie zadań z drugiego kamienia milowego . . . . .	13
<b>3</b>	<b>Etap 3 - Prace końcowe</b>	<b>13</b>
3.1	Opis . . . . .	13
3.2	Zadania . . . . .	13
3.2.1	Pobieranie tweetów w czasie rzeczywistym . . . . .	13
3.2.2	Pobieranie i przetwarzanie tweetów z Kafki . . . . .	13
3.2.3	Przetworzenie danych historycznych . . . . .	14
3.2.4	Budowa aplikacji do wizualizacji . . . . .	14
3.3	Wykonanie zadań z trzeciego kamienia milowego . . . . .	15



---

# 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

### 1.2.1 Analiza dostępności API mediów społecznościowych

Odpowiedzialni: **Piotr Rosa, Aleksei Haidukevich**

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**. W przypadku Facebooka korzystamy z programu **Face-Pager**. Jest to program pozwalający pobierać wszystkie posty z wybranej strony i zapisać je do pliku z rozszerzeniem csv. Opcjonalna jest dodatkowa konwersja do JSON. Posty można pobierać z całego okresu istnienia strony.

### 1.2.2 Analiza dostępnej infrastruktury do przetwarzania danych.

Odpowiedzialny: **Jakub Korczakowski**

W docelowej aplikacji przetwarzającej dane w czasie rzeczywistym planujemy użyć:

**Apache Hadoop** 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.

### 1.2.3 Analiza i wybór sposobu wdrożenia aplikacji (rozwiązania chmurowe)

Odpowiedzialni: **Jakub Korczakowski, Maciej Leszczyński**

Po analizie potrzeb naszego projektu zdecydowaliśmy się wykorzystać roz-

---

wiązanie chmurowe, ponieważ pozwala 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 platformie AWS zdecydowaliśmy się wybrać platformę Azure.

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

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

Pomimo tego, że te usługi znacznie ułatwiają 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.

#### 1.2.4 Wybór odpowiedniej bazy do składowanych danych (porównanie SQL i noSQL)

Odpowiedzialni: **Jakub Korczakowski, Maciej Leszczyński**

Do składowania historycznych danych w naszym projekcie będziemy wykorzystywać bazę SQL ze względu na szybkość dostępu do danych.

#### 1.2.5 Stworzenie programu pobierającego dane z Twittera

Odpowiedzialny: **Piotr Rosa**

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.

---

### 1.2.6 Stworzenie programu pobierającego dane z Facebooka

Odpowiedzialny: **Aleksei Haidukevich**

Pobieranie danych z Facebooka przy użyciu programu FacePager pozwala na dostęp do danych niezależnie od dat powstania postów. Dzięki temu, uruchamiając program co określony czas dla wybranych stron, otrzymujemy pliki csv, z których dane przetwarzane są przy użyciu języka Python. Interesujące nas dane są analogiczne do tych w przypadku Twittera.

### 1.2.7 Instalacja wybranej bazy

Odpowiedzialny: **Jakub Korczakowski**

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.

### 1.2.8 Połączenie bazy danych z programami pobierającymi dane

Odpowiedzialny: **Jakub Korczakowski**

Serwer bazodanowy działa na platformie **Azure**. Zarówno dla danych z Twittera, jak i z Facebooka, 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.

### 1.2.9 Pobranie danych do bazy danych

Odpowiedzialny: **Maciej Leszczyński**

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 zapisywane w bazie danych w czasie rzeczywistym za pomocą opracowanych programów.

### 1.2.10 Analiza dostępnych zbiorów opisanych klasami, pozwalających na testowanie algorytmu

Odpowiedzialna: **Marharyta Kruk**

Znaleziony został zbiór pozwalający na trenowanie oraz testowanie algo-

---

rytmu. Zbiór zawiera 1600000 opisanych tweetów. Dla każdego tweetu został obliczony sentyment.

Zbiór znajduje się pod adresem: [adres](#).

### 1.3 Wykonanie zadań z pierwszego kamienia milowego

Podczas tego etapu nie byliśmy w stanie wykonać wszystkich zadań, które założyliśmy w statucie projektu. Udało nam się rozpocząć budowę modelu oraz znaleźć odpowiedni zbiór treningowy, pozwalający na analizę sentymentu. Udało się nam również poznać API Twittera oraz rozpocząć pracę nad programami pobierającymi dane. Próba wykorzystania wiadomości z Reddita zakończyła się niepowodzeniem ze względu na brak odpowiednich danych w serwisie. Trudniejsze niż przewidywaliśmy okazało się również zbudowanie klastra na platformie MS Azure. Potrzebowaliśmy więcej czasu, aby zapoznać się z możliwościami platformy, co spowodowało przesunięcie się pracy.

## 2 Etap 2 - Przetwarzanie danych

czas: 7.01.2020 - 31.03.2020

### 2.1 Opis

W drugim etapie do najważniejszych celów należą cele niezrealizowane poprawnie w pierwszym etapie, czyli budowa infrastruktury do przetwarzania danych oraz stworzenie programów pobierających dane. Kolejnymi celami będzie stworzenie programów przetwarzających dane i uruchomienie modeli na przygotowanej infrastrukturze. W odniesieniu do pierwszego etapu dokonaliśmy zmiany bazy danych na bazę NoSQL, zdecydowaliśmy się wybrać bazę MongoDB.

### 2.2 Zadania

#### 2.2.1 Budowa infrastruktury do przetwarzania danych

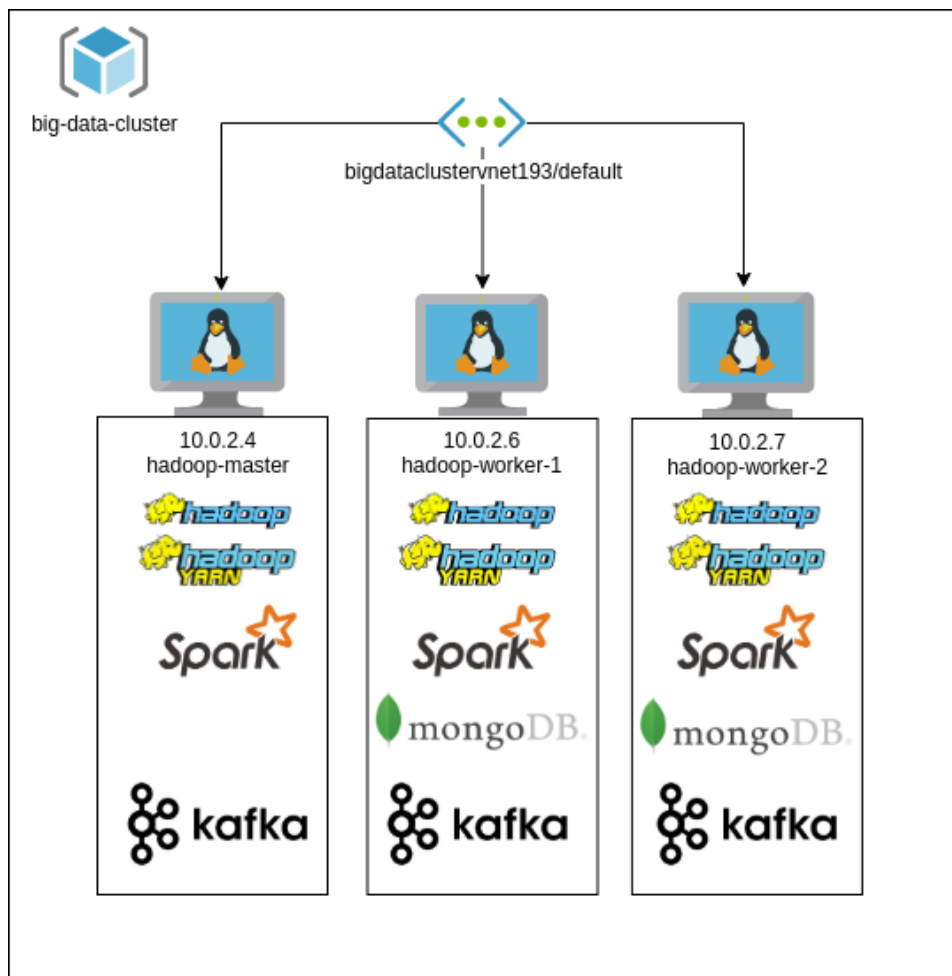
Odpowiedzialny: **Jakub Korczakowski**

W celu utworzenia infrastruktury zdolnej przetwarzać dane zostały utworzone 3 maszyny wirtualne w grupie zasobów **big-data-cluster** na platformie **Azure** w regionie **North Europe**. Są to odpowiednio:

**hadoop-master** Standard D2s v3 (2 vcpus, 8 GiB memory),

**hadoop-worker-1** Standard B2s (2 vcpus, 4 GiB memory),

**hadoop-worker-2** Standard B2s (2 vcpus, 4 GiB memory).



Rysunek 1: Diagram przygotowanej infrastruktury.

Ze względu na specyfikę platformy **Azure** wielkość i rodzaj maszyn może zostać przeskalowany w razie potrzeb. Wszystkie maszyny są dostępne pod publicznymi adresami IP poprzez protokół ssh. Dodatkowo maszyna **hadoop-master** posiada środowisko graficzne i można dostać się do niej również poprzez protokół RDP na porcie 3389.

Ilość maszyn jest zdeterminowana częściowo poprzez limit nałożony na subskrypcję studencką. Limit wynosi 6 vcpus na strefę regionalną maszyn.

### 2.2.2 Instalacja narzędzi do przetwarzania danych

Odpowiedzialny: **Jakub Korczakowski**

**Apache Hadoop** Został zainstalowany na klastrze. Wartość replikacji pli-



---

ków została zmieniona na **2** ze względu na ilość dostępnych DataNode.

**YARN** Został skonfigurowany w klastrze.

**Apache Spark** Został zainstalowany na klastrze oraz skonfigurowany w trybach działania **standalone** oraz poprzez **YARN**. Został połączony ze środowiskiem **Hadoop**.

**MongoDB** Baza została zainstalowana na maszynie **hadoop-master**. Została skonfigurowana w sposób umożliwiający odwołanie się do niej z sieci lokalnej (poprzez ip **10.0.2.4**). Poprzez odpowiednie opcje konfiguracyjne możliwe jest połączenie z **Apache Spark**.

**Apache Kafka** Broker Apache Kafka został zainstalowany jako klaster na wszystkich maszynach. Dodatkowo został zainstalowany Apache Zookeeper, który nadzoruje działanie brokera.

**Rola maszyn w systemach.**

	Apache Hadoop	YARN	Apache Spark
hadoop-master	NameNode	ResourceManager	Master
hadoop-worker-1	DataNode	NodeManager	Slave
hadoop-worker-2	DataNode	NodeManager	Slave

### 2.2.3 Baza Danych

Odpowiedzialny: **Jakub Korczakowski**

Baza danych została zainstalowana na maszynach **hadoop-woker-1** oraz **hadoop-woker-2**. Pierwsza z maszyn odpowiada za Twittera, a druga za Facebooka.

### 2.2.4 Potok przetwarzania danych w projekcie

Odpowiedzialny: **Jakub Korczakowski**

Podczas tego etapu udało nam się również zaplanować dokładniej potok przetwarzania danych w naszym projekcie. Planowany potok widoczny jest na poniższym rysunku.

Do pobierania danych w czasie rzeczywistym w naszym projekcie użyjemy Apache Kafka, który za pomocą skryptów napisanych w języku Python łączył się z Twitter API i pobierał dane. Przetwarzanie strumieniowe planujemy przeprowadzić za pomocą Apache Spark Streaming, a wyniki analiz zapisać w MongoDB. Przetwarzanie strumieniowe będzie obejmować głównie analizę sentymentu pobieranych tweetów. Wyniki z bazy będą na końcu prezentowane przez napisaną przez nas aplikację.



Rysunek 2: Diagram potoku danych w projekcie.

### 2.2.5 Udoskonalenie programu do pobierania danych z Twittera

Odpowiedzialny: **Piotr Rosa**

Program pobierający dane działa w języku Python z wykorzystaniem biblioteki Tweepy. Pobrane dane zapisywane są z wykorzystaniem systemu Avro. Pliki zapisane w tym formacie mają strukturę podobną, umożliwiającą łatwe przekształcenie, do formatu JSON. Główną wadą zapisu danych w JSON jest fakt, że nazwy pól zapisywane są dla każdego rekordu, co znacznie zwiększa zużycie pamięci, przez co nie nadaje się on dla dużych zbiorów danych. W przypadku Avro, schemat definiujący każdy rekord, zapisany jest jeden raz dla każdego pliku. Aby umożliwić współpracę programu z platformą Hadoop, tzn. zapis zebranych plików z danymi do systemu plików HDFS, wykorzystujemy bibliotekę Pythona `hdfs`, a zapis plików w formacie Avro umożliwia pakiet z tej biblioteki `hdfs.ext.avro`. Program, ze względu na brak możliwości nieustannego działania maszyn, uruchamiany jest domyślnie co dwa dni. Początkowo planowaliśmy pobieranie danych w odstępach tygodniowych, ponieważ Twitter pozwala na pobranie danych historycznych z okresu tygodnia, jednak istnieją również limity zapytań. Limit ten to 1800 zapytań w ciągu 15 minut, przy czym każde zapytanie zwraca maksymalnie 100 tweetów. Nie wystarcza to do naszych celów i konieczne byłoby czekanie około godziny na pobranie wszystkich interesujących nas wpisów, dlatego zdecydowaliśmy się pobierać je częściej. Dane zapisywane są w HDFS w strukturze: `data/<nazwa_firmy>/<nazwa_firmy>#.avro`, gdzie `#` oznacza kolejny numer pobrania. Program wykorzystuje pomocniczy plik, w którym zapisywana jest liczba dotychczas wykonanych pobrań danych, zdefiniowane są nazwy firm, które śledzimy oraz numer ID ostatniego tweeta dla każdej z nich, na podstawie którego określamy jak długo pobierać tweety, aby ich nie duplikować. Wybraliśmy osiem firm, kurierskich oraz dostarczających żywność, które będziemy śledzić.

Kroki przy tworzeniu programu: Na maszynie, którą posłużymy się do pobierania danych zainstalowaliśmy potrzebne biblioteki Pythona:

```
pip install tweepy
```

---

```
pip install hdfs[avro]
```

Następnie stworzyliśmy plik konfiguracyjny `/.hdfscli.cfg`, potrzebny do łączenia się z HDFS. Później został stworzony testowy program zapisujący dane do pliku w formacie `.avro` do HDFS oraz program odczytujący ten plik. Po uruchomieniu programu zapisującego plik, sprawdziliśmy zawartość HDFS. Wykonaliśmy kolejno polecenia: `hdfscli -alias=dev CLIENT.list('test/')` `CLIENT.status('test/testfile.avro')`

Oto ich wyniki:

```
stud@hadoop-worker-1:~/scripts/tests$ hdfscli --alias=dev
```

```
Welcome to the interactive HDFS python shell.  
The HDFS client is available as CLIENT.
```

```
CLIENT.list('test/')  
[u'testfile.avro']  
CLIENT.status('test/testfile.avro')  
{u'group': u'supergroup', u'permission': u'644',  
u'blockSize': 134217728, u'accessTime': 1586095573925,  
u'pathSuffix': u'', u'modificationTime': 1586095574082,  
u'replication': 2, u'length': 283, u'childrenNum': 0,  
u'owner': u'stud', u'storagePolicy': 0, u'type': u'FILE',  
u'fileId': 16815}  
Oraz wynik programu odczytującego ten plik z HDFS:  
{u'username': u'user1', u'value': 1, u'title': u'test1'}  
{u'username': u'user2', u'value': 2, u'title': u'test2'}
```

Jak widać plik został poprawnie zapisany, istnieje możliwość odczytania go oraz widać, że został on zduplikowany. Gdy wszystko działało poprawnie stworzyliśmy właściwy program.

Obserwowane firmy to: DHL, Uber Eats, FedEx, Grubhub, DoorDash, Postmates, Woolworths.

## 2.2.6 Stworzenie modelu do analizy sentymentów

Odpowiedzialna: **Marharyta Kruk**

Model został stworzony na zbiorze Twitter Sentiments, zawierającym 1.6 milion tweetów o *negatywnych*, *pozytywnych* oraz *neutralnych* sentymentach. Model do analizy sentymentów Reddita nie został stworzony z powodu trudności zebrania niezbędnych danych. Model, nauczony na zbiorze tweetów, został przyznany jako przygodny do analizy dowolnych zdań. Zbiór

---

danych został początkowo przetworzony, zostały wyrzucone stop listy, zostawione stemmery. Wejściem do sieci służą przetworzone słowa, każde słowo jest przedstawione liczbą. Sieć przedstawia sobą prostą sieć sekwencyjną i składa się z warstw:

- Embedding, podstawą dla której jest nauczony i zapisany model Word2Vec.
- Dropout
- LSTM
- Dense

Na wyjściu otrzymujemy się liczbą, określającą stopień sentymentu. W algorytmu liczy się, że wynik poniżej 0.4 odnosi się do negatywnego sentymentu, powyżej 0.7 – do pozytywnego, reszta jest określana jako sentyment neutralny. Algorytm osiąga precyzję 80%

### 2.2.7 Stworzenie programu do pobierania danych z Facebooka

Odpowiedzialni: **Maciej Leszczyński, Aleksei Haidukevich**

Dokonano różnych prób pobierania komentarzy z-pod postów na Facebook, lista narzędzi które probowaliśmy:

- <https://github.com/strohne/Facepager> (ma tylko graficzną powłokę)
- <https://github.com/sriniskanda/Extracts-comments-and-posts-from-facebook>
- <https://github.com/rugantio/fbcrawl>
- <https://github.com/piocalderon/facebook-comment-scraper>
- <https://github.com/weijiekoh/scrape-fb-posts-and-comments>
- <https://github.com/chribsen/facebook-pages-scraper>
- <https://github.com/jpryda/facebook-multi-scraper>

Nie udało się wyciągnąć dane z różnych powodów (głównie nie są te programy dostosowane do nowych wersji Facebook Graph API, albo po prostu nie działają). Użycie większości z nich oraz ręczna próba wysyłania requestu w sposób opisany w dokumentacji API Facebook kończy się komunikatem o tym, że do pobierania danych z Facebook aplikacja ma przejść przez app review (może trwać miesiące) oraz posiadać "terms of service" i temu podobne rzeczy.

Wniosek po tej pracy: choć komentarze mają duży potencjał pod względem analizy sentymentu, musimy zrezygnować z Facebooka i używać tweetów jako jedyne źródło danych, co z kolei pozwoli skupić się raczej na architekturze systemu i narzędziach do przechowywania i przetwarzania danych.

---

## 2.3 Wykonanie zadań z drugiego kamienia milowego

W statucie błędnie założyliśmy, że utworzenie infrastruktury oraz potoku danych zajmie nam tylko czas na początku projektu. Dzięki wcześniejszemu poznaniu platformy Azure udało nam się stworzyć klastrer zdolny przetwarzać nasze dane. Poczyniliśmy również duże postępy w kierunku skończenia potoku danych. Udało nam się również stworzyć model analizy sentymentu. W statucie nie przewidzieliśmy głównie nakładu pracy potrzebnego do wykonania zadań z pierwszego etapu, mocno go zaniżając. Z drugiej strony zakładaliśmy, że na budowę algorytmu potrzebna będzie nam uwaga całego zespołu przez cały drugi etap, co okazało się przecenieniem ilości czasu potrzebnego na wykonanie tego zadania. W kolejnym, ostatnim etapie kluczowe będzie dokończenie przetwarzania danych oraz budowa aplikacji prezentującej nasze wyniki.

## 3 Etap 3 - Prace końcowe

**czas: 4.11.2019 - 15.12.2019**

### 3.1 Opis

Ostatni etap projektu to dopracowanie istniejących rozwiązań, stworzenie ostatnich potrzebnych programów oraz połączenie wszystkich komponentów. W tym etapie zajęliśmy się przetwarzaniem strumieniowym oraz aplikacją prezentującą.

### 3.2 Zadania

#### 3.2.1 Pobieranie tweetów w czasie rzeczywistym

Odpowiedzialny: **Piotr Rosa**

Został stworzony potok przetwarzania dla tweetów w czasie rzeczywistym. Wykorzystując bibliotekę Pythona Tweepy powstał skrypt który otwiera połączenie z Twitterem i nasłuchuje tweetów na wybrane przez nas tematy. Następnie z postów które otrzymujemy wyciągamy interesujące nas informacje - id posta, użytkownik który go dodał, data publikacji oraz jego treść. Następnie informacje te publikowane są na serwer Kafka, skąd mogą zostać pobrane do dalszego przetwarzania. Publikacja na serwer Kafka odbywa się z wykorzystaniem biblioteki Pythona kafka-python.

#### 3.2.2 Pobieranie i przetwarzanie tweetów z Kafki

Odpowiedzialny: **Piotr Rosa, Aleksei Haidukevich**

Kolejny program który został stworzony w ramach tego kamienia milowego

---

odpowiedzialny jest za pobieranie wiadomości z Kafki, a następnie ich przetwarzanie. Do tego celu wykorzystana została biblioteka `pyspark.streaming` oraz `pyspark.streaming.kafka`. Tweety pobierane są z tematu w dziesięciosekundowych interwałach, a następnie są przetwarzane. Pierwszym krokiem przetwarzania jest rozdzielenie tweetów tematycznie, ze względu na firmy których one dotyczą. Krok ten mógł zostać zrealizowany również podczas pobierania tweetów z Twittera i publikacji ich do Kafki - mogły być one publikowane do różnych tematów. Podczas pobierania nie było jednak informacji które tweety dotyczą których firm, więc musiałyby zostać wykonana ta sama operacja sprawdzania. Uznaliśmy, że efektywniej będzie sprawdzać to po pobraniu z Kafki, z wykorzystaniem Sparka. Po rozdzieleniu tweetów dokonywana jest predykcja ich sentymentu przy użyciu stworzonego wcześniej modelu. Dodawane są informacje o wyniku tweeta oraz sentymencie, który został określony na jego podstawie. Dane w takiej postaci zapisywane są do bazy danych Mongo, przy czym każda firma ma oddzielną kolekcję.

### 3.2.3 Przetworzenie danych historycznych

Odpowiedzialny: **Piotr Rosa**

Stworzony został program który przetwarza tweety zbierane już od początkowych faz projektu. Ze względu na brak modelu, tweety nie mogły zostać przetworzone i nie mieliśmy informacji o ich sentymencie. Postanowiliśmy nie zapisywać tych danych w bazie Mongo, a w postaci plików z rozszerzeniem `.avro`. Teraz, gdy model już istnieje możliwe jest przetworzenie tych danych. Aby usprawnić działanie tego procesu wykorzystujemy Sparka. Program iteruje po plikach w systemie HDFS, zawierających historyczne wpisy, wczytuje je oraz dokonuje predykcji sentymentu dla każdego wpisu. Podobnie jak w przypadku przetwarzania strumieniowego, do każdego tweeta dołączana jest informacja o jego wyniku oraz sentymencie, a następnie w takiej formie zapisywany jest do odpowiedniej kolekcji w Mongo.

### 3.2.4 Budowa aplikacji do wizualizacji

Odpowiedzialny: **Aleksei Haidukevich, Maciej Leszczyński, Marharyta Kruk**

Aplikacja przedstawiająca wyniki naszych działań została stworzona w języku Python z wykorzystaniem biblioteki `Plotly Dash`. Została ona uruchomiona korzystając z serwera webowego `Flask` na maszynie `hadoop-worker-2`. Aplikacja obejmuje dostanie się do danych z bazy danych oraz przekazaniem ich do odpowiednich funkcji odpowiedzialnych za powstanie wykresów.

Aplikacja łączy się z bazą `MongoDB` za pomocą biblioteki `pymongo`, i pobiera dane, znajdujące się w kolekcjach `"histogramtestoraz"` oraz `"tweetsentimentz"`

worker-1. Co 20 sekund zbiera wszystkie tweety dopisane do bazy w tym czasie, aby zliczać ich ilość, oraz robi to samo co minutę aby przedstawić dystrybucję ilości tweetów w trakcie ostatnich 5 minut.

Wykresy, które stworzyliśmy obejmują zarówno wizualizację na żywo, jak i przedstawienie danych historycznych. Wykresy live przedstawiają liczbę i sentyment tweetów w zależności od firmy (od czasu uruchomienia aplikacji), sentyment tweetów w ostatnich 5 minutach oraz wykresy kołowe z podziałem na firmy i sentyment wszystkich tweetów. Wykresy obejmujące dane historyczne pokazują liczbę i sentyment tweetów w zależności od dnia (od początku zbierania danych) oraz ze względu na dzień tygodnia. Dodatkowo w aplikacji zostały umieszczone przyciski pozwalające wybrać firmę, która nas interesuje; dla tej firmy zostaną pokazane wyżej wymienione wykresy szczegółowe.



Rysunek 3: Zrzut ekranu przedstawiający działającą aplikację. Firma dla której przedstawione są wykresy to FedEx. Widoczny jest wzrost negatywnych wpisów w okolicy 22 maja.

### 3.3 Wykonanie zadań z trzeciego kamienia milowego

W ostatnim etapie prac zwiększyliśmy czas przeznaczony na pracę nad projektem. Dzięki temu wykonaliśmy wszystkie zadania wyznaczone w trzecim etapie prac. Praca nad aplikacją internetową przebiegała sprawnie, ponieważ część zespołu miała już doświadczenie w technologii Dash, której używaliśmy.



Rysunek 4: Zrzut ekranu przedstawiający działającą aplikację. Firma dla której przedstawione są wykresy to DHL.

## 4 Podsumowanie

Podczas pracy nad tym projektem naprawdę dużo się nauczyliśmy. Poznaliśmy wiele narzędzi oraz technologii. Poznaliśmy technologie Big Data oraz uczenia maszynowego. Poszerzyliśmy swoją wiedzę techniczną o wdrażanie aplikacji internetowych oraz korzystanie z chmury MS Azure. Uważamy jednak, że najważniejszą rzeczą, którą nauczyliśmy się podczas pracy nad projektem to zarządzanie projektami oraz sprawy z nim powiązane. Nie mieliśmy wcześniej szansy pracy przy tak dużym i długoterminowym projekcie, więc była to dla nas możliwość zdobycia cennego doświadczenia. Staraliśmy się planować pracę oraz dbać o dobrą komunikację, jednak musieliśmy zmierzyć się z brakiem doświadczenia. Uważamy, że zrealizowaliśmy założenia projektowe oraz wiele nauczyliśmy się podczas pracy nad projektem.