Politechnika Warszawska

WYDZIAŁ ELEKTRYCZNY

Instytut Elektrotechniki Teoretycznej i Systemów Informacyjno-Pomiarowych
Zakład Elektrotechniki Teoretycznej i Informatyki Stosowanej

Praca dyplomowa magisterska

na kierunku Informatyka
w specjalności Inżynieria oprogramowania

Analiza wiadomości z serwisów społecznościowych na użytek rozpoznawania nastrojów.

Jakub Rzepliński
Numer albumu: 233608

promotor
dr inż. Marcin Kołodziej

Warszawa, 2018

Streszczenie

Abstract

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że przedstawiona praca dyplomowa:

- została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami,
- nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego lub stopnia naukowego w wyższej uczelni

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

....
data podpis autora (autorów) pracy

Oświadczenie

Wyrażam zgodę / nie wyrażam zgody*1 na udostępnianie osobom zainteresowanym mojej pracy dyplomowej. Praca może być udostępniana w pomieszczeniach biblioteki wydziałowej. Zgoda na udostępnienie pracy dyplomowej nie oznacza wyrażenia zgody na jej kopiowanie w całości lub w części.
Brak zgody nie oznacza ograniczenia dostępu do pracy dyplomowej osób:

- reprezentujących władze Politechniki Warszawskiej,
- członków Komisji Akredytacyjnych,
- funkcjonariuszy służb państwowych i innych osób uprawnionych, na mocy odpowiednich przepisów prawnych obowiązujących na terenie Rzeczypospolitej Polskiej,

do swobodnego dostępu do materiałów chronionych międzynarodowymi przepisami o prawach autorskich. Brak zgody nie wyklucza także kontroli tekstu pracy dyplomowej w systemie antyplagiatowym.

....
data podpis autora (autorów) pracy

*1 - niepotrzebne skreślić

Spis treści

Rozdział 1

Wstęp

Czynnikiem powodującym rozwój świata są informacje i umiejętne ich wykorzystanie. Od jakiegoś czasu źródłem informacji stał się internet, a szczególnie portale społecznościowe, gdzie ludzie wymieniają się informacjami na różne tematy. Wiele globalnych instytucji korzysta z tego typu serwisów, wydając za ich pośrednictwem oficjalne komunikaty. Ilość informacji generowanych przez użytkowników takich serwisów jest tak duża, że wymaga użycia specjalnych narzędzi *Big Data*. Jednak decydującym czynnikiem jest umiejętne wykorzystanie wiedzy zawartej w zgromadzonych informacjach, do czego potrzebny jest czynnik ludzki.

Jednym z ciekawych sposobów wykorzystania informacji z serwisów takich jak np. *Twitter* jest badanie wydźwięku wpisów zamieszczanych przez użytkowników zwanego od angielskiego sformułowania *sentiment analysis* sentymentem. Aby było to możliwe potrzebne jest przetworzenie języka, którym posługują się ludzie czyli *języka naturalnego* na postać, którą mogą posługiwać się systemy NLP (ang. *Natural Language Processing*), a następnie określenie sentymentu za pomocą podejścia słownikowego lub technik maszynowego uczenia się (ang. *machine learning*). Wykorzystanie takich informacji pozwala zbadać opinie ludzi na praktycznie dowolny temat.

Cel pracy

Celem niniejszej pracy jest próba stworzenia narzędzia do analizy sentymentu wypowiedzi użytkowników serwisu społecznościowego Twitter. Stworzone powinno zostać rozwiązanie umożliwiające przetwarzanie w czasie rzeczywistym wiadomości publikowanych w tym serwisie zawierających wybrane słowo kluczowe. Napisany system powinien wyświetlać statystyki wydźwięku opinii użytkowników tego serwisu.

Zakres pracy

Pracę dyplomową można podzielić na trzy części: przegląd najistotniejszych informacji o poruszanych tematach, opis napisanej aplikacji oraz omówienie przeprowadzonych badań i wyciągnięte z nich wnioski.

W pierwszej części w rozdziale 2. opisano zasadę działania serwisu społecznościowego Twitter wraz z udostępnionym przez niego programowalnym interfejsem (ang. API - *Application Programming Interface*). W rozdziale 3. zostały podane informacje na temat analizy sentymentu wypowiedzi w systemach komputerowych.

W drugiej części w rozdziale 4. przedstawiono wymagania funkcjonalne i niefunkcjonalne

postawione przed zbudowaną aplikacją. W rozdziale 5. opisano wybrane do jej implementacji narzędzia, biblioteki oraz języki programowania. W rozdziale 6. znajduje się omówienie stworzonego systemu *Twitter Analyser*.

W ostatniej części w rozdziale 7. omówiono badania sentymentu wypowiedzi przeprowadzone podczas dwóch wydarzeń - meczu piłki nożnej FC Barcelona - Real Madryt oraz Święta Dziękczynienia. W ostatnim rozdziale 8. znajduje się podsumowanie całej pracy dyplomowej.

Rozdział 2

Serwis społecznościowy Twitter

Serwis społecznościowy Twitter jest globalnym serwisem internetowym służącym głównie do zamieszczania wiadomości tzw. *tweet*, które użytkownicy tego serwisu mogą czytać, komentować lub przekazywać dalej. Od kilku lat Twitter jest serwisem, gdzie dochodzi do wymiany zdań na różny temat dotyczących np. polityki, sportu, produktów, wydarzeń społecznych, a profile posiada wiele osób znanych publicznie oraz instytucji.

Serwis ten, nazywany SMS-em internetu, został założony w 2006 r. w Stanach Zjednoczonych przez Jacka Dorseya, Ev Williamsa, Noah Glassa oraz Biza Stone'a i od początku powstania sukcesywnie zwiększał swoją popularność poprzez wzrost liczby użytkowników odwiedzających jego witrynę oraz wysyłających wiadomości [1]. W 2012 r. osiągnął ponad 100 milionów użytkowników, którzy zamieszczali łącznie ponad 340 milionów wiadomości dziennie oraz obsługiwał średnio około 1.6 miliarda wyszukujących zapytań dziennie. W 2013 r. Twitter stał się jedną z najczęściej odwiedzanych stron w całym internecie. W tym samym roku inżynierowie Twittera podali informację, że serwis ten obsługuje ok. 143 tys. wiadomości na sekundę. Na początku 2016 r. serwis posiadał ponad 319 milionów użytkowników aktywnych podczas każdego miesiąca. Od listopada 2013 r. akcje Twittera są obecne na nowojorskiej giełdzie.

Tweet, czyli krótka wiadomość tekstowa, była początkowo ograniczona do 140 znaków, ale limit ten został podwojony w 2017 r. dla wszystkich języków oprócz chińskiego, japońskiego i koreańskiego. Użytkownicy mają możliwość wyróżniania wybranych przez siebie tematów przez dodanie do nich znaku '#', co czyni takie wyrażenie tagiem. Inną możliwością oferowaną przez Twittera jest odpowiadanie innym użytkownikom lub zamieszczenie referencji do nich przez dodanie znaku '@' poprzedzającego nazwę profilu innej osoby.

Serwis społecznościowy Twitter opierał się początkowo o typową architekturę trójwarstwową składająca się z warstwy prezentacji, logiki biznesowej oraz warstwy danych. Do napisania tej aplikacji został użyty framework Ruby on Rails wykorzystujący język Ruby, a warstwa bazodanowa opierała się o technologię MySQL. Jednak wraz ze wzrostem ilości przetwarzanych danych inżynierowie Twittera podjęli decyzję w 2011 r. o zmianie technologii na język Scala, który działa na maszynie wirtualnej Javy oraz zrezygnowano z dotychczasowej architektury na rzecz budowy rozproszonych serwisów komunikujących się między sobą. Wraz z przeprowadzonymi zmianami zanotowano ponad 10-krotne polepszenie obsługi twittów.

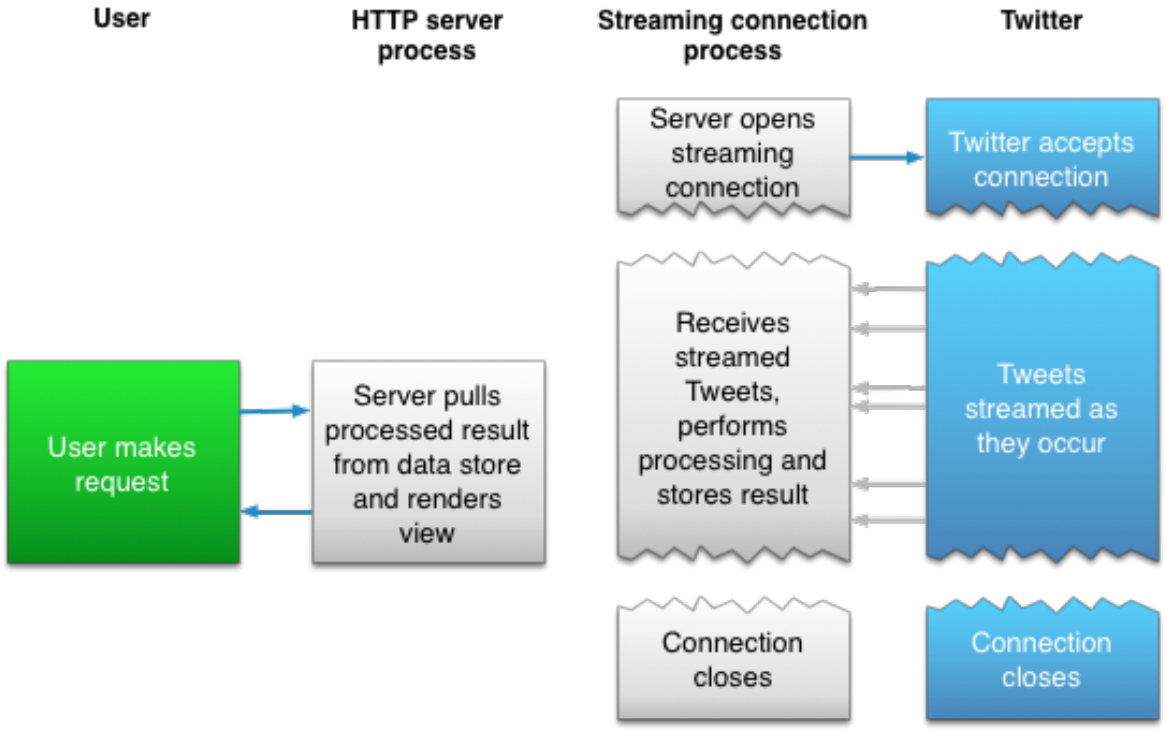
Twitter jest platformą otwartą i udostępnia programowalny interfejs (ang. API - *Application Programming Interface*) w dwóch postaciach: Search API oraz Streaming API.

Programiści korzystający z Search API są w stanie uzyskać dostęp tylko do danych histo-

rycznych, które zostały już wcześniej zamieszczone na łamach serwisu Twitter. Natomiast w przypadku Streaming API dostajemy możliwość śledzenia strumienia danych, które są do naszego dostępu nawet już kilka sekund po zamieszczeniu w serwisie Twitter. Po podłączeniu do takiego strumienia możemy cały czas obserwować nowe wiadomości. Przyjęło się stosować nazywnictwo, że analiza Search API to analiza *back in time*, a Streaming API to śledzenie *real time*.

Obie formy API wymagają wcześniejszej rejestracji na stronie przeznaczonej dla programistów zainteresowanych wykorzystywaniem Twitter API. Po pomyślnym przejściu rejestracji dostajemy dane, które po nawiązaniu połączenia z serwisem Twitter umożliwiają mu jednoznacznie określić, że możemy uzyskać dostęp do API.

Search API powstało z wykorzystaniem standardu REST - *Representational State Transfer*. Oba rodzaje API wykorzystują protokół HTTP: do poprawnego działania Streaming API potrzebne jest ciągłe połączenie HTTP, a w przypadku drugiego z nich każda operacja jest wykonywana przy nawiązaniu oddzielnego połaczenia. Schemat działania obu wersji API został przedstawiony na rysunku 2.1..



Rysunek 2.1: Schemat działania dwóch rodzajów programistycznego interfejsu API udostępnianego przez serwis społecznościowy *Twitter*: *Search API* i *Streaming API* [2].

Search API posiada ściśle określone parametry, które mogą być przesłane w żądaniu. W tabeli 2.3 zaprezentowano ich wykaz.

Streaming API nie posiada takich ograniczeń. W języku programowania Java dostępny jest pakiet *twitter4j* zawierający interfejsy *User* oraz *Status*, na które mapowane są przychodzące ze strumienia informacje. W tabelach 2.1 i 2.2 zamieszczono ich dokumentację.

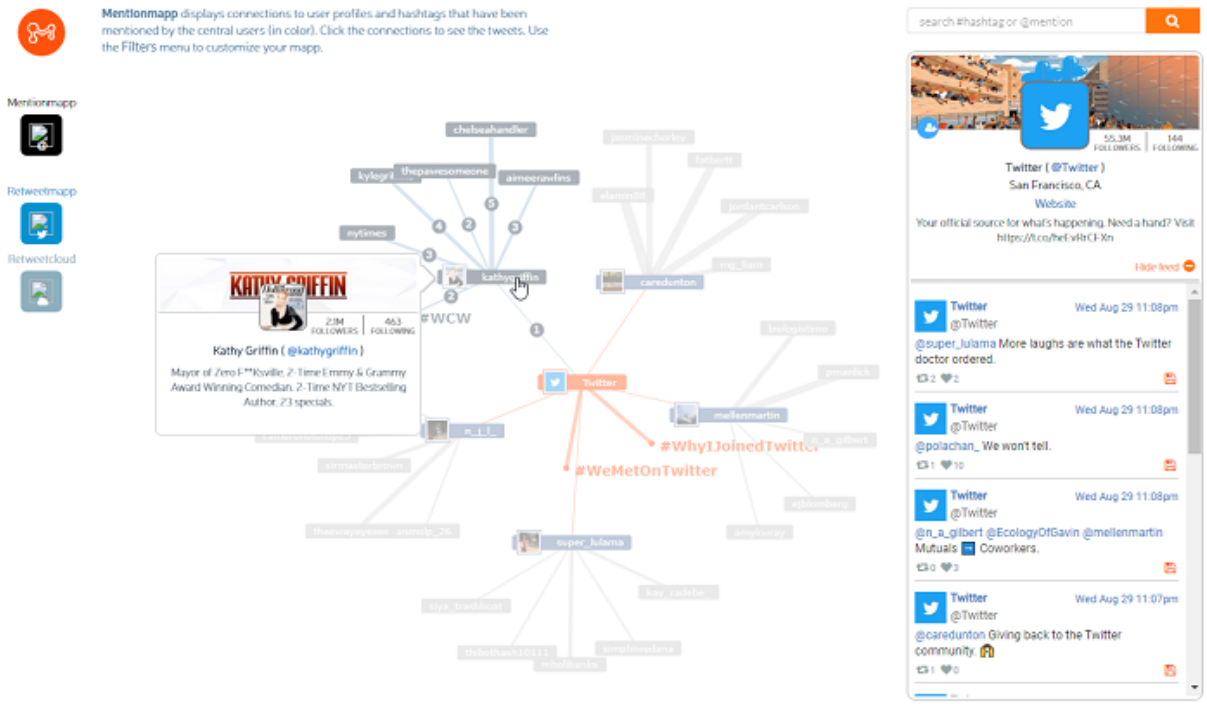
Korzystając z Search API mamy możliwość wysłania 720 zapytań na godzinę, a maksymalna ilość wiadomości jaka może być zwrócona na jedno zapytanie to 100 [4]. Jeśli wykorzystalibyśmy ten limit w maksymalny sposób to daje nam to 72 000 wiadomości na godzinę. W

przypadku Streaming API głównym ograniczeniem jest dostęp do ok. 1 % danych ze strumienia, a maksymalna ilość wiadomości w czasie jednej minuty to 3 000. W przypadku tego API w ciągu godziny możemy uzyskać 180 000 wiadomości. Są to ograniczenia, które obowiązują dla rozwiązań typu *open-source*.

Udostępnienie API przez serwis społecznościowy Twitter oraz rosnące znaczenie danych generowanych przez użytkowników tego serwisu spowodowało, że wiele firm oraz instytucji zaczęło przywiązywać dużą wagę do analizy opinii wyrażanych na swój temat lub na tematy pokrewne oraz kształtujących się trendów. Dlatego powstały aplikacje internetowe służące do wyświetlania takich informacji i przeprowadzające wstępną analizę zebranych danych.

Narzędziem wartym omówienia jest *Mentionmap*. Jest to aplikacja, która wyróżnia się spośród innych tym, że rysuje wykres powiązań pomiędzy użytkownikami wchodzącymi w interakcje z danym profilem, a także z ich profilami [5]. Posiada także podstronę umożliwiającą badanie nastrojów społecznych osób zamieszczających wiadomości z konkretnym słowem kluczowym. Nastroje te są przedstawione w postaci chmury nazw kont użytkowników, gdzie kolor nazwy zależy od nastoroju prezentowanego przez użytkownika pod kątem słowa kluczowego. Interfejs graficzny narzędzia Mentionmap został przedstawiony na rysunkach 2.2 i 2.3..

Podsumowując warto zauważyć, że nie ma obecnie na rynku aplikacji, która umożliwiałaby śledzenie występowania dowolnego słowa w wiadomościach zamieszczanych w czasie rzeczywistym w serwisie Twitter, rysowałaby wykres zależności pomiędzy użytkownikami tego serwisu, analizowałaby nastroje społeczne użytkowników z wyświetleniem informacji o nastroju wyrażanym w poszczególnych wiadomościach oraz pozwalałaby na analizowanie danych historycznych i zamieszczanych w czasie rzeczywistym. Taka sytuacja pozwala na utworzenie nowej aplikacji, która udostępniałaby te funkcjonalności.



Rysunek 2.2: Narzędzie Mentionmap - wykres powiązań [5].

Tablica 2.1: Dokumentacja interfejsu *Status* z pakietu *twitter4j* [7].

Typ zwracany	Nazwa metody	Opis
long[]	getContributors()	
java.util.Date	getCreatedAt()	zwraca datę utworzenia wiadomości
long	getCurrentUserRetweetId()	zwraca id użytkownika, którego wiadomość została podana dalej
int	getDisplayTextRangeEnd()	
int	getDisplayTextRangeStart()	
int	getFavoriteCount()	zwraca informację ile razy została polubiona wiadomość
GeoLocation	getGeoLocation()	zwraca lokalizację użytkownika zamieszczającego tą wiadomość
long	getId()	zwraca id wiadomości
java.lang.String	getInReplyToScreenName()	zwraca nazwę użytkownika, do którego kierowana jest odpowiedź
long	getInReplyToStatusId()	zwraca id wiadomości, do którego kierowana jest odpowiedź
java.lang.String	getLang()	zwraca język zamieszczonej wiadomości
Place	getPlace()	zwraca obiekt Place przypisany do tej wiadomości
Status	getQuotedStatus()	zwraca obiekt Status cytowanej wiadomości
long	getQuotedStatusId()	zwraca id cytowanej wiadomości
URLEntity	getQuotedStatusPermalink()	zwraca obiekt URLEntity reprezentujący bezpośredni odnośnik do cytowanej wiadomości
int	getRetweetCount()	zwraca ile razy wiadomość została podana dalej
Status	getRetweetedStatus()	zwraca oryginalny status, który jest podany dalej w tej wiadomości
Scopes	getScopes()	zwraca obiekt typu Scopes posiadający informację o id miejsc, do których odnosi się ta wiadomość
java.lang.String	getSource()	zwraca źródło wiadomości
java.lang.String	getText()	zwraca tekst wiadomości
User	getUser()	zwraca obiekt typu User powiązany z tą wiadomością
java.lang.String[]	getWithheldInCountries()	zwraca tablicę nazw krajów, w których wiadomość została wstrzymana

boolean	isFavorited()	zwraca informację czy wiadomość została polubiona
boolean	isPossiblySensitive()	zwraca informację czy wiadomość zawiera link do chronionych informacji
boolean	isRetweet()	zwraca informację czy tweet jest podaną dalej wiadomością
boolean	isRetweeted()	informuje czy wiadomość jest podana dalej
boolean	isRetweetedByMe()	informuje czy wiadomość jest podana dalej przez tego użytkownika
booleand	isTruncated()	informuje czy wiadomość jest skrócona (zakończona znakiem "...")

Tablica 2.2: Dokumentacja interfejsu *User* z pakietu *twitter4j* [6]. Zamieszczone zostały tylko najważniejsze z metod.

Typ zwracany	Nazwa metody	Opis
java.util.Date	getCreatedAt()	zwraca datę utworzenia profilu użytkownika
java.lang.String	getDescription()	zwraca opis konta użytkownika
java.lang.String	getEmail()	zwraca adres e-mail powiązany z tym kontem
int	getFavouritesCount()	zwraca liczbę wiadomości, którą polubił ten użytkownik
int	getFollowersCount()	podaje ilość użytkowników śledzących profil
int	getFriendsCount()	podaje ilość śledzonych profili
long	getId()	zwraca id użytkownika
java.lang.String	getLang()	zwraca język preferowany przez użytkownika
java.lang.String	getLocation()	zwraca lokalizację użytkownika
java.lang.String	getName()	podaje nazwę użytkownika
java.lang.String	getScreenName()	zwraca nazwę konta
Status	getStatus()	zwraca obiekt typu Status reprezentujący wiadomość wysłaną przez użytkownika
int	getStatusesCount()	podaje ilość wiadomości wysłanych przez użytkownika
java.lang.String	getTimeZone()	podaje strefę czasową użytkownika
java.lang.String	getURL()	zwraca URL do profilu
boolean	isVerified()	podaje informację czy profil jest zweryfikowany

Tablica 2.3: Parametry żądania *Twitter Search API* [3].

Parametr	Wymagany/Opcjonalny	Opis	Przykład
q	wymagany	zapytanie wyszukujące o maksymalnej długości 500 znaków	nasa
geocode	opcjonalny	zwraca wiadomości użytkowników oddalonych o podany promień od podanej szerokości i długości geograficznej, promień może być podany w milach lub kilometrach	37.781157 -122.398720 1mi
lang	opcjonalny	ogranicza wiadomości do wybranego języka spośród dostępnych kodów ISO 639-1	pl
locale	opcjonalny	specyfikuje język wysyłanego zapytania, obecnie tylko *j*a jest skuteczny	ja
result_type	opcjonalny	określa typ zwracanych wiadomości, obecnie dostępne są trzy wartości tego parametru: *recent* (zwracane są najnowsze wiadomości), *popular* (zwracane są najbardziej popularne wiadomości) *mixed* (wartość domyślna, zwracane wyniki obejmują najnowsze i najbardziej popularne wiadomości)	mixed
count	opcjonalny	specyfikuje ilość zwracanych wiadomości; maksymalna wartość to 100, a domyślna to 15	100

until	opcjonalny	ten parametr odpowiada za zwracanie wiadomości, których data utworzenia jest starsza o maksymalnie tydzień niż podana; obowiązuje format YYYY-MM-DD	2015-07-19
since_id	opcjonalny	dzięki temu parametrowi zwracane są wiadomości o ID większym niż podane czyli nowsze wiadomości niż określono	12345
max_id	opcjonalny	dzięki temu parametrowi zwracane są wiadomości o ID mniejszym lub równym niż podane czyli starsze lub takie same wiadomości niż określono	54321

Rysunek 2.3: Narzędzie Mentionmap - badanie nastrojów społecznych [5].

Rozdział 3

Analiza sentymentu wypowiedzi

Zagadnieniem dobrze ilustrującym postęp technologiczny jest prędkość rozprzestrzeniania się informacji. W starożytności informacja była przekazywana zazwyczaj przez posłańców, od których losów zależało czy i kiedy zostanie ona dostarczona do nadawcy. Tak było w przypadku posłańca Filippidesa, który według legendy podążał po wygranej bitwie pod Maratonem do Aten, aby uprzedzić Greków przed zbliżającym się atakiem Persów [8]. W dzisiejszych czasach coraz częściej informacja jest dostępna na ekranach urządzeń mobilnych takich jak telefon komórkowy w czasie rzeczywistym zaraz po wystąpieniu jakiegoś zdarzenia. Dlatego bardzo ważne stało się monitorowanie opinii i nastojów społecznych. Dostępne obecnie narzędzia pozwalają klasyfikować wydźwięk tekstu jako pozytywny, negatywny albo neutralny.

Podstawową operacją języka naturalnego jest ustalanie znaczenia zdania zwane także semantyką. Ustalenie znaczenia wyrażenia w języku naturalnym polega na wyznaczeniu występujących w nim obiektów oraz zachodzących między nimi relacji [9]. Do analizy pełnego znaczenia wymagana jest szeroka wiedza o świecie odnosząca się do konretnego kontekstu. Na tym etapie przetwarzania tekstu pomocne są informacje o powiązaniach między słowami. W zależności od zastosowania przydatna może być na przykład wiedza o tym, że kot jest ssakiem, a rafineria rodzajem przedsiębiorstwa.

Do reprezentacji semantyki języka naturalnego można wykorzystać mechanizmy formalne, które odpowiadają praktycznym potrzebom dokonania interpretacji. Jendym z przykładów takich mechanizmów jest *rachunek predykatów I rzędu*, który pozwala zapisać czy fakt jest prawdziwy lub fałszywy, umożliwia zapisywanie pytań za pomocą użycia zmiennych, a także posiada opracowane metody wnioskowania. Inną metodą reprezentowania znaczenia zdania jest *Teoria Reprezentacji Dyskursu - DRT* (ang. *Discourse Representation Theory*), która polega na przekształcaniu drzewa rozbioru znaczenia zdania na strukturę zwaną *DRS* (ang. *Discourse Representation Structure*). Niestety żaden ze znanych mechanizmów nie jest w stanie odzwierciedlić całej złożoności procesów powiązanych z rozumieniem języka naturalnego. Przyjmując jedną z istniejących metod lub tworząc nową musimy zmierzyć się z wybraniem mechanizmu, który wobec postawionych wymagań najlepiej poradzi sobie z rozległością skali znaczeń jakie chcemy reprezentować, stopniem skomplikowania semantyki oraz kosztem jej uzyskania. Dlatego większość systemów informatycznych nie korzysta z wyrafinowanych reprezentacji znaczenia, ale ogranicza ją do najprostszych scenariuszy.

Analiza tekstu jest złożonym zadaniem. Wymaga zrozumienia zależności pomiędzy występującymi faktami oraz do jego pełnego zrozumienia potrzebna jest wiedza, którą dysponuje człowiek. Dlatego analiza tekstu jest uznawana za problem *AI-zupełny* (ang. *AI - Artifical Intel-*

ligence lub po polsku *SI - Sztuczna Inteligencja*). Zrozumienie kolejnej części tekstu wymaga umiejętnego wyciągania wniosków z poprzedniej części tekstu i powiązania ich z wiedzą nabytą z innych źródeł.

Opis dłuższego tekstu wymaga odtworzenia powiązań pomiędzy kolejnymi zdaniami, ale z powodu ilości możliwych kombinacji sekwencji zdań można opisać tylko wybrane zjawiska oraz wykluczyć niektóre typy powiązań. Kolejną trudnością jest rozstrzyganie do jakich obiektów odnoszą się wyrażenia wskazujące, ponieważ w tekstach stosowane są sposoby opisu tego samego obiektu w różny sposób np. wszystkie frazy odnoszące się do tej samej osoby - *Janek, kolega Maćka, mały chłopiec, ten z prawej, najmłodszy w rodzinie*. Inną ważną kwestią są części wpływające na ciągłość tekstu - nawiązanie do poprzedniego tekstu lub wypowiedzi, ciągłość opisów.

Analiza nastrojów społecznych wypowiedzi jest możliwa po poddaniu tekstu zamianie na reprezentację, którą mogą posługiwać się systemy informatyczne. Poniżej przedstawiono opis kilku takich reprezentacji:

- **bag of words** [10] - najpopularniejszy sposób reprezentacji tekstu w postaci zestawu wyrazów z przyporządkowanymi im liczbami wystąpień w tekście np. zdanie *Jan lubi oglądać filmy, a Maria także lubi je oglądać* można zapisać w notacji *JSON* (ang. *JSON - JavaScript Object Notation*) jako *{"Jan": 1, "lubi": 2, "oglądać": 2, "filmy": 1, "a": 1, "Maria": 1, "także": 1, "je": 1}*; innym przykładem zastosowania takiej reprezentacji jest wykorzystanie jej w mechanizmach odpowiadających za filtrowanie wiadomości e-mail.; wadą takiego podejścia jest utrata informacji o kolejności wyrazów w zdaniu i powiązaniach między nimi.
- **reprezentacja wektorowa** (ang. *vector space model*) [11, 12] - dokument tekstowy reprezentowany jest w postaci wektorów częstości występowania słów; tworzona jest macierz w której termy (np. wyrazy) odpowiadają wierszom, a dokumenty (np. strony internetowe) kolumnom; wadą tego podejścia jest fakt, że ilość słów może być bardzo duża co skutkuje utworzeniem ogromnej macierzy; ta technika maszynowego uczenia się znajduje swoje zastosowanie m.in. przy określaniu kategorii badanych tekstów.
- **reprezentacja grafowa** [13] - podejście rozszerzające reprezentację wektorową, w której słowa są węzłami połączonymi ze sobą krawędziami jeśli występują razem w tekście; istnieje możliwość zaobrazowania kolejności występowania słów z wykorzystaniem krawędzi skierowanych.

Jako istoty ludzkie posiadamy zdolność do rozpoznawania cudzych emocji po treści wypowiedzi i towarzyszących jej czynników pozawerbalnych, której wydźwięk określa się jako stosunek lub postawa pozostająca w korelacji do pewnego zdarzenia lub sytuacji. Treści publikowane w internecie w formie tekstu niosą ze sobą tylko reprezentację tekstową, dlatego odczytanie nastroju wyrażanego w taki sposób jest trudnym zadaniem. Tym bardziej złożonym procesem wymagającym dokładnego przeanalizowania każdego słowa jest zadanie odczytania wydźwięku wypowiedzi przez napisany system.

Dziedziną zajmującą się analizą wypowiedzi jest przetwarzanie języka naturalnego (ang. *NLP - natural language processing*), gdzie językiem naturalnym określa się język stosowany przez ludzi do komunikacji interpersonalnej. Zadaniem systemów rozumiejących język naturalnych jest przekształcenie go na formę bardziej przyjazną dla komputerów. Natomiast podzbiór tych systemów służący do określania sentymentu zwraca ogólną informację o wydźwięku w ustalonej skali. Przykładowo analiza sentymentu w uważanej za punkt odniesienia dla takich

algorytmów bibliotece NLP, napisanej przez pracowników i studentów amerykańskiego Uniwersytetu Stanforda, zwraca informację o sentymencie w pięciostopniowej skali: bardzo negatywny, negatywny, neutralny, pozytywny, bardzo pozytywny. Jest to ogólna informacja, ale nawet taka w postaci statystyk jest w stanie posłużyć jako cenne źródło informacji.

Algorytmy analizujące sentyment muszą poradzić sobie z następującymi wymaganiami [14, 15]:

- złożoność języka naturalnego,
- niejednoznaczność wypowiedzi np. wieloznaczność słowa zamek: *akcja Zemsty Fredry dzieje się na zamku w Odrzykoniu* lub *zamek w moich drzwiach nie chce się otworzyć* lub syntaktyczna niejednoznaczność gdy w jednej części zdania znajduje się pozytywny wydźwięk, a w kolejnej negatywny: *pogoda była okropna, ale obiad bardzo nam smakował*,
- specyfika stosowanego języka np. w internecie z powodu niejednokrotnie nie zachowanych zasad gramatycznych lub wiadomości nie niosących ze sobą żadnej treści,
- neologizmy np. retweet,
- idiomy np. "urwanie głowy"
- problemy z rozpoznawaniem nazw np. *byliśmy wczoraj na K2* - czy chodzi o film czy o szczyt górski,
- problem wyrwania wypowiedzi z kontekstu,
- tekst o charakterze sarkastycznym.

Techniki badania sentymentu wypowiedzi dzielimy na dwa rodzaje: korzystające ze słowników, które korzystają ze zbudowanych wcześniej list słów kluczowych z określonym sentymentem oraz oparte na klasyfikatorach, które wykorzystują techniki maszynowego uczenia się (ang. *machine learning*). Podstawowe techniki badania sentymentu to [16]:

- **słownik** (ang. *lexicon based approach*) - klasyfikator ten bada tekst na podstawie wystąpień słów przy wykorzystaniu słownika składającego się ze słów z przypisanym pozytywnym lub negatywnym wydźwiękiem; sentyment pojedynczego słowa określa się jako iloraz prawdopodobieństwa wystąpienia z pozytywnym sentymentem do prawdopodobieństwa wystąpienia z negatywnym; wadą takiego podejścia jest brak analizy powiązania między słowami.
- **naiwny klasyfikator Bayesa** (ang. *naive Bayes classifier*) - probabilistyczny klasyfikator wykorzystujący techniki maszynowego uczenia się, opierający się o zasadę niezależności słów oraz wykorzystujący założenie, że teksty o charakterze pozytywnym charakteryzują się określonym słownictwem, a te o charakterze negatywnym charakteryzują się innym; podejście to zakłada także, że tekst, w którym występuje więcej słów o charakterze z kategorii pozytywnej lub negatywnej powinien zostać zaklasyfikowany do tej kategorii;
- **technika maksymalnej entropii** (ang. *maximum entropy technique*) - podejście szacujące rozkład prawdopodobieństwa opierające się na założeniu, że rozkład ma maksymalną entropię, jeśli dane nie są dobrze znane; entropia zwana także miarą niepewności rozkładu jest kolejną metodą wykorzystującą techniki maszynowego uczenia się.

– **metoda wektorów nośnych** (ang. *support vector machines*) - technika *machine learning* opierająca się o ideę hiperpłaszczyzny dzielącej teksty na pozytywne oraz negatywne z jak najmniejszym marginesem; celem jest znalezienie funkcji, dla której błąd sklasyfikowania tekstu będzie najmniejszy; programy wykorzystujące tą metodę dobrze radzą sobie z dużą ilością słów, ale oznaczenie części z nich jako nieistotne może powodować utratę cześci informacji.

Jak wynika z badań przeprowadzonych na grupie ponad 2000 dorosłych Amerykanów 81% internautów przynajmniej raz szukało opinii o produkcie w internecie, a 20% Amerykanów robi to na co dzień [17]. Około 80% użytkowników internetu spośród wspomnianej grupy badawczej przyznało, że opinie przeczytane w internecie miały znaczny wpływ na dokonane przez nich zakupy. Wyniki tych badań pokazują jak bardzo wydźwięk informacji, opinii, recenzji oraz poglądów w internecie ma wpływ na zachowania ludzi. Najczęściej spotykane zastosowania analizy sentymentu wypowiedzi zamieszczanych w internecie to [18]:

– analizowanie opinii wyrażanych na temat produktów np. krótko po premierze nowego produktu firmy chcą wiedzieć jak są one odbierane tak aby móc w porę zaplanować wprowadzenie poprawek lub aby prognozować wyniki sprzedaży oraz cenę akcji, firmy próbują także docierać do osób krytykujących aby przekonać ich do zmiany zdania lub żeby krytyków konkurencji zachęcić do zakupu swoich produktów,

– badanie opinii na temat firm np. jako pracodawców lub ich odbiór na rynku,

– analiza opinii na temat ugrupowań politycznych i polityków np. dotycząca obecnej sytuacji lub przyszłych decyzji oraz nastawienia badanych grup społecznych,

– badanie nastrojów społecznych podczas wydarzeń sportowych może posłużyć np. do uzyskania wiedzy jak odbierane są decyzje właścicieli klubów piłkarskich.

Rozdział 4

Wymagania funkcjonalne i niefunkcjonalne

Jak już zostało wspomniane w rozdziale dotyczącym serwisu Twitter przy okazji omówienia dostępnych narzędzi, nie ma obecnie na rynku aplikacji, która umożliwiałaby śledzenie występowania dowolnego słowa w wiadomościach zamieszczanych w czasie rzeczywistym w tym serwisie, rysowałaby wykres zależności pomiędzy użytkownikami, analizowałaby nastroje społeczne użytkowników z wyświetleniem informacji o nastroju wyrażanym w poszczególnych wiadomościach oraz pozwalałaby na analizowanie danych historycznych i zamieszczanych w czasie rzeczywistym. Głównym celem tej pracy dyplomowej jest stworzenie aplikacji, która posiadałaby wspomniane funkcjonalności.

Główną funkcjonalnością, którą powinna zapewnić budowana aplikacja jest dostęp do usystematyzowanych danych pochodzących z sewisu Twitter, które będą nieść ze sobą informację o sentymencie. Dane te powinny być także przechowywane w taki sposób, aby móc zapewnić do nich dostęp w dowolnym momencie oraz spoza zaimplementowanego narzędzia.

Wymagania funkcjonalne, które dotyczą budowanego rozwiązania to:

- **przetwarzanie danych z serwisu Twitter** - aplikacja powinna przetwarzać tweety użytkowników serwisu Twitter, do których dostęp można uzyskać przez Streaming API tego serwisu, które zostało szczegółowo omówione w rozdziale 2.;
- **filtrowanie napływających danych po słowie kluczowym** - dane napływające w czasie rzeczywistym powinny być filtrowane pod względem zawartości w treści wiadomości słowa kluczowego, które zostało określone przez użytkownika za pomocą graficznego interfejsu aplikacji;
- **zapis napływających danych** - budowane narzędzie powinno zapisywać przetworzone, uporządkowane i dotyczące wybranego słowa kluczowego dane w lokalnej bazie danych, która umożliwiałaby dostęp do nich przez swój wbudowany pulpit w dowolnym momencie oraz spoza zaimplementowanego narzędzia;
- **duża częstotliwość pobierania danych** - aplikacja powinna pobierać informacje w krótkich odstępach czasu, ponieważ serwis Twitter charakteryzuje duża ilość informacji przesyłanych w każdej sekundzie;
- **prezentowanie danych historycznych** - narzędzie powinno prezentować dane historyczne zgromadzone podczas przetwarzania danych napływających wówczas w czasie

rzeczywistym;

- **prezentowanie podstawowych informacji o danych napływających w czasie rzeczywistym** - aplikacja powinna umożliwiać analizowanie podstawowych informacji o wiadomościach i użytkownikach, które będą napływać w czasie rzeczywistym;
- **dostęp do szczegółowej informacji o użytkowniku** - implementowane narzędzie powinno umożliwiać dostęp do informacji o każdym użytkowniku zainteresowanym wybranym słowem kluczowym, którego wiadomość udało się zarejestrować podczas gromadzenia danych z wykorzystaniem Streaming API serwisu Twitter;
- **dostęp do szczegółowej informacji o wiadomości** - aplikacja powinna wyświetlać szczegółową informację, o każdej wiadomości zawierającej wybrane słowo kluczowe i zapisanej podczas gromadzenia danych z wykorzystaniem Streaming API;
- **prezentowanie informacji statystycznej o sentymencie na dany temat** - narzędzie powinno prezentować statystyki sentymentu użytkowników serwisu Twitter, którzy w swoich wiadomościach zawarli wybrane słowo kluczowe i których wiadomości udało się zapisać podczas analizy danych napływających w czasie rzeczywistym;
- **możliwość jednoczesnej analizy danych historycznych i napływających w czasie rzeczywistym** - aplikacja powinna umożliwiać jednoczesną analizę danych historycznych i napływających w czasie rzeczywistym bez konieczności ponownego uruchamiania aplikacji lub zatrzymywania jednej z analiz;
- **przyjazny interfejs graficzny** - narzędzie powinno posiadać wygodny, łatwy do nauczenia oraz prosty interfejs graficzny.

Wymagania niefunkcjonalne, które powinna spełniać przygotowywana aplikacja to:

Spełnienie głównych założeń systemu czasu rzeczywistego

Głównym przypadkiem biznesowym, dla którego tworzona jest wspomniana aplikacja jest sytuacja dużego i globalnego zainteresowania pewnym tematem, które objawia się odnoszeniem się do niego w wiadomościach zamieszczanych przez użytkowników serwisu Twitter. Można stwierdzić, że przygotowywane narzędzie będzie przykładem systemem czasu rzeczywistego jeśli system ten będzie "*urządzeniem technicznym, którego wynik i efekt działania będzie zależny od chwili wypracowania tego wyniku*"[19]. Wspólną cechą definicji takiego systemu jest "*zwrócenie uwagi na równoległość w czasie zmian w środowisku oraz obliczeń realizowanych na podstawie stanu środowiska*"[19].

Płynna obsługa danych

Budowany system powinien przetwarzać dane w czasie nie większym niż tempo napływania nowych informacji. Koniecznością jest zatem skorzystanie z narzędzi umożliwiających sprawne przetwarzanie danych, ale także ich zapis do bazy w czasie nie większym niż czas trwania określonego okna czasowego.

Możliwość działania aplikacji i analizowania danych historycznych w trybie offline

Dane zapisane podczas sesji korzystania ze strumienia danych serwisu Twitter będą zapisywane w lokalnej bazie danych, dlatego stworzone narzędzie powinno umożliwiać analizowanie danych historycznych bez połączenia z internetem.

Jednorazowe przetwarzanie informacji ze strumienia danych

Aplikacja powinna tylko jeden raz przetwarzać i zapisywać do bazy danych informacje pozyskane ze strumienia. Jeśli w bazie istnieje już informacja o użytkowniku to nowe wiadomości powinny być z nim powiązane.

Niezawodność

System powinien charakteryzować się niezawodnością podczas pracy z dużą ilością danych napływających w krótkich odstępach czasu oraz jak najbliższym prawdy określaniem sentymentu wypowiedzi zawartej w wielu wiadomościach.

System napisany na potrzeby tej pracy dyplomowej powinien spełniać wszystkie z wymienionych wymagań niefunkcjonalnych, ponieważ nie spełnienie nawet jednej z nich może spowodować, że aplikacja będzie nieużyteczna. Postawione wymagania funkcjonalne i niefunkcjonalne, łącząc się z opisem serwisu Twitter oraz analizą sentymentu wypowiedzi, definiują potrzeby jakie powinny umożliwiać narzędzia wybrane do jej budowy oraz samo narzędzie. Zostanie to omówione w następnych rozdziałach.

Rozdział 5

Wybór narzędzi

Przy pisaniu aplikacji tworzonej wraz z niniejszą pracą dyplomową będzie trzeba się zmierzyć z wymaganiami funkcjonalnymi i niefunkcjonalnymi określonymi w poprzednim rozdziale. Aby tego dokonać niezbędne jest wybranie odpowiednich narzędzi, które pomogą w ich realizacji. W dalszej części rozdziału zaprezentowano uzasadnienie wyboru konkretnych składowych tworzonego systemu.

Językiem programowania wybranym do napisania tej części aplikacji jest Java. Jest to język, który charakteryzuje niezależność od systemu operacyjnego i procesora. Uniwersalny kod kompilowany jest do kodu bajtowego i następnie wykonywanego przez maszynę wirtualną Javy.

W wersji 8 tego języka, która w 2018 roku była najczęściej wybierana jako wersja do tworzenia dużych systemów komercyjnych [20], dodano wiele udogodnień pozwalających na częściowe pisanie kodu w myśl paradygmatu programowania funkcyjnego dzięki wprowadzonym operacjom na strumieniach i wyrażeniom lambda. Java swoją popularność zawdzięcza także wykorzystaniu do tworzenia aplikacji webowych, której przykładem jest aplikacja tworzona na potrzeby tej pracy. Język ten cały czas się rozwija. Przygotowywana jest obecnie 12. wersja Javy, która ma być dostępna w marcu 2019 roku. Przykład zastosowania operacji na strumieniach oraz wyrażenie lambda przedstawiono na rysunku 5.1..

```
private Set<Link> computeInterestedInLinks(Set<TwitterUser> interestedInUsers, String keywordName) {
    return interestedInUsers
            .stream()
            .map(user -> new Link(keywordName, user.getScreenName(), RelationType.INTERESTED_IN))
            .collect(Collectors.toSet());
}
```

Rysunek 5.1: Przykład zastosowania operacji na strumieniach oraz wyrażenia lambda w kodzie napisanym w języku *Java* w wersji 8 [materiały własne].

Jedną z bibliotek użytych w aplikacji jest biblioteka Lombok, która jest bardzo pomocna przy pisaniu prostych i powtarzalnych części kodu takich jak np. konstruktory, akcesory, mutatory, czy metody equals i hashCode. Przykładowo żeby napisać konstruktor posiadający jako argumenty wszystkie pola klasy wystarczy dodać na klasie adnotację *@AllArgsConstructor* co zaprezentowano na rysunku 5.2.. Lombok jest rozwijaną bibliteką typu open-source. Jest bardzo przydatna w podstawowych potrzebach kodu pisanego w Javie.

Językiem programowania wybranym do napisania części prezentacji przygotowywanej aplikacji jest JavaScript. Język ten znajduje swoje zastosowanie w tworzeniu stron internetowych.

```
@AllArgsConstructor
@Getter
public class Link {

    private final String keyword;
    private final String source;
    private final String target;
    private final RelationType type;

}
```

Rysunek 5.2: Przykład zastosowania adnotacji biblioteki *Lombok* [materiały własne].

Umożliwia reagowanie na zdarzenia wywołane w przeglądarce oraz wyświetlanie efektownych efektów wizualnych.

Dodatkowo w części front-end zostanie użyta biblioteka *React.js*, która jest określana jako *silnik do budowy interfejsów użytkownika* [21]. React.js charakteryzuje się wykorzystaniem reaktywnego renderowania, które polega na przechowywaniu w pamięci reprezentacji struktury DOM i ponowne renderowanie interfejsu użytkownika tylko w tych miejscach, które zależą od zmienionego stanu. Jest to biblioteka bardzo wydajna i szybka co będzie bardzo pomocne przy budowie aplikacji.

Frameworkiem, na którym opierać się będzie tworzony system jest *Spring Boot*. *Spring Framework* jest określany jako główna platforma programistyczna, która sprawia, że programowanie w języku Java staje się łatwiejsze i bardziej wydajne. Spring Boot natomiast opiera się na zasadzie działania Spring Framework, ale rozwiązuje wiele problemów wynikających z potrzeby konfiguracji projektu przed rozpoczęciem implementowania nowych funkcji. Posiada zbiór wszystkich wymaganych bibliotek oraz konfiguracji zwanych *starterami* oraz wbudowany serwer aplikacyjny Tomcat. Spring jest bardzo dobrze udokumentowanym frameworkiem używanym przez programistów w wielu bardzo zróżnicowanych systemach.

Narzędziem wybranym do przetwarzania danych strumieniowych będzie *Apache Spark Streaming*, które charakteryzuje się łatwością integracji oraz operowaniem na kolekcjach informacji zamiast pojedynczych wiadomościach. Rozwiązanie to opiera się o *RDDs - Resilient Distributed Datasets* reprezentujące kolekcje rekordów, na których można wykonywać operacje znane z paradygmatu programowania funkcyjnego np. *map*, *filter*, *groupByKey* i *join* [22]. Spark Streaming z definicji operuje na każdej porcji danych dokładnie raz. W bardzo rzadkim przypadku wystąpienia błędu może się zdarzyć, że dane nie zostaną przetworzone ani razu. Wówczas zostaną utracone dane z jednego okna czasowego trwającego kilka sekund co w kontekście budowanego rozwiązania wydaje się być dopuszczalne. Spark Streaming dobrze integruje się z językiem Java i frameworkiem Spring. Schemat zasady działania Apache Spark znajduje się na rysunku 5.3..

Systemem bazy danych wybranym do budowanego narzędzia jest *neo4j*. Jest to nierelacyjny system bazodanowy, które w odróżnieniu do relacyjnych charakteryzują się brakiem narzuconego schematu modelu informacji. Umożliwia to przechowywanie różnych danych bez potrzeby zapisu ich w postaci relacji, które w przypadku większej ilości danych są bardzo kosz-

Rysunek 5.3: Zasada działania narzędzia *Apache Spark Streaming* [21].

towne, ponieważ wymagają łączenia informacji z kilku tabel.

Dane w neo4j przechowywane są w postaci grafów, w których węzły odpowiadają wierszom z relacyjnych baz danych, a krawędzie pomiędzy węzłami są odpowiednikiem kluczów obcych. Dzięki takim rozwiązaniom wyszukiwanie zależności między węzłami polega na poruszaniu się po grafie, a nie na kosztownym łączeniu danych. Grafowe bazy danych powstały wraz z rozwojem sieci społecznościowych, dlatego jest naturalne, że tego typu rozwiązanie zostało wybrane do budowy przygotowywanej aplikacji. neo4j dobrze integruje się z wybranymi już narzędziami. Przykład zapytania napisanego w języku Cypher wykorzystywanego przez grafową bazę neo4j zaprezentowano na rysunku 5.5..

```
@Query("MATCH (keyword:Keyword)<-[relation:INTERESTED_IN]-(user:TwitterUser) " +
        "WHERE keyword.name = {keyword} " +
        "RETURN DISTINCT user")
Stream<TwitterUser> findAllInterestedIn(@Param("keyword") String keyword);
```

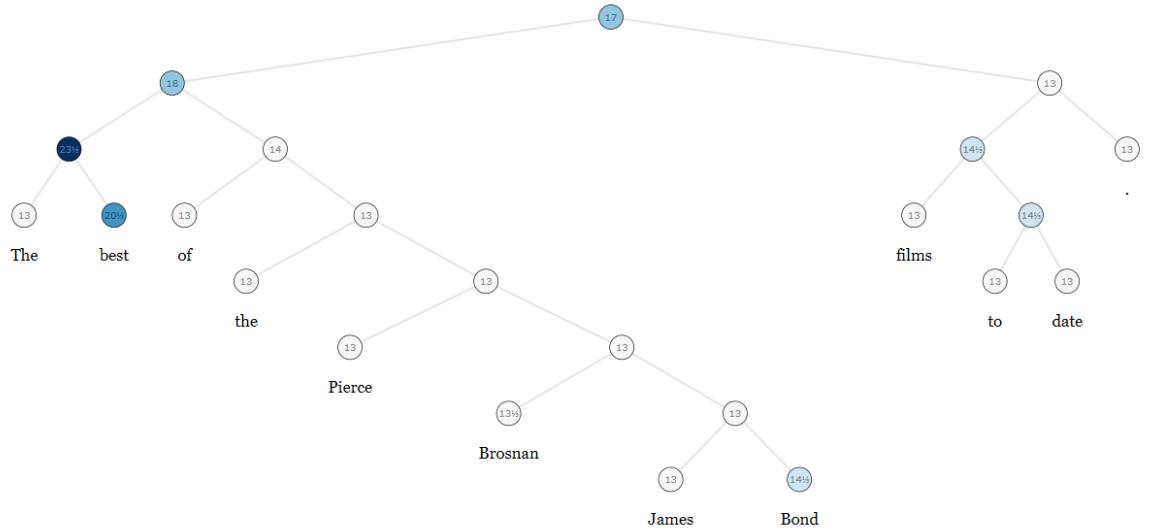
Rysunek 5.4: Przykład zapytania napisanego w języku *Cypher* wykorzystującego *Spring Data* [materiały własne].

Biblioteką wybraną do analizy sentymentu jest rozwiązanie *Stanford NLP Sentiment Analysis* tworzone i cały czas rozwijane na Uniwersytecie Stanforda. Większość dostępnych rozwiązań określa sentyment tekstu biorąc pod uwagę wydźwięk pojedynczych słów i następnie podając wynik dla całego tekstu. W ten sposób umyka informacja o kolejności wyrazów oraz zostaje zaburzony sentyment. Stanford NLP analizuje każde zdanie tekstu w całości i oblicza sentyment na podstawie tego w jaki sposób słowa składają się one na znaczenie w dłuższe zwroty [23].

Narzędzie to korzysta z sieci neuronowej *Recursive Neural Network*, która opiera się na zasadach gramatycznych. Sieć ta została wytrenowana z wykorzystaniem zbioru recenzji filmowych zawierających prawie 12 tysięcy zdań i około 215 tysięcy słów.

Jednym z pierwszych kroków w analizie wydźwięku tekstu za pomocą tej biblioteki jest podział tekstu na zdania. Stanford NLP w swojej pełnej wersji udostępnia także parser tekstów, który może zostać użyty przy tej okazji. Następnie każde zdanie jest analizowane oddzielnie, a sentyment całego tekstu jest obliczany jako średnia arytmetyczna jego składowych. Wydźwięk podawany jest w pięciostopniowej skali: bardzo negatywny, negatywny, neutralny, pozytywny, bardzo pozytywny. Co ciekawe twórcy tego narzędzia umożliwiają na swojej stronie internetowej zwrócenie im uwagi, że pewne zdanie ze zbioru treningowego źle określa sentyment. Wówczas zostanie to przeanalizowane i poprawione. Przykład rozkładu zdania za pomocą omawionej biblioteki zaprezentowano na rysunku 5.6..

Stanford NLP wspiera obecnie badanie sentymentu tylko w tekstach angielskojęzycznych. Jego twórcy podają, że skuteczność stosowanego przez nich algorytmu wynosi 85.4% oraz że jest to obecnie jedyne rozwiązanie, które wspiera negacje na różnych poziomach budowanych



Rysunek 5.5: Przykład rozkładu opinii na temat filmu z serii *James Bond* [24].

drzew w zwrotach pozytywnych i negatywnych. Narzędzie to dobrze integruje się z wybranymi już wcześniej składowymi budowanego systemu.

Rozdział 6

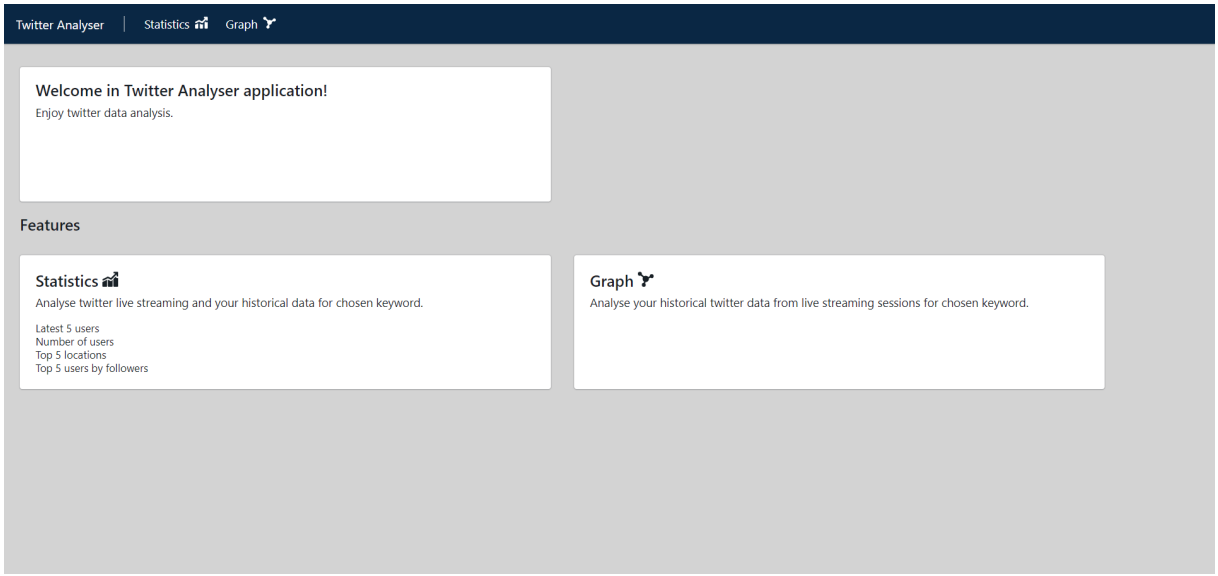
Aplikacja Twitter Analyser

Narzędziem stworzonym w ramach pracy dyplomowej jest aplikacja nazwana *Twitter Analyser*. Rozwiązanie to zostało zaimplementowane z użyciem narzędzi wymienionych w poprzednim rozdziale. Jest to webowa aplikacja działająca lokalnie, która po odpowiedniej konfiguracji, może zostać zainstalowana na komputerze.

System ten posiada dwie główne funkcjonalności:

1. Przetwarzanie interesujących danych ze strumienia udostępnionego przez portal społecznościowy Twitter i zapis ich do bazy danych;
2. Analizowanie zapisanych danych pod względem wydźwięku zamieszczanych opinii oraz powiązań między użytkownikami.

Główny ekran narzędzia, widoczny na rysunku 6.1, prezentuje podstawowe informacje o aplikacji oraz jej podstronach, a także umożliwia przejście na nie.

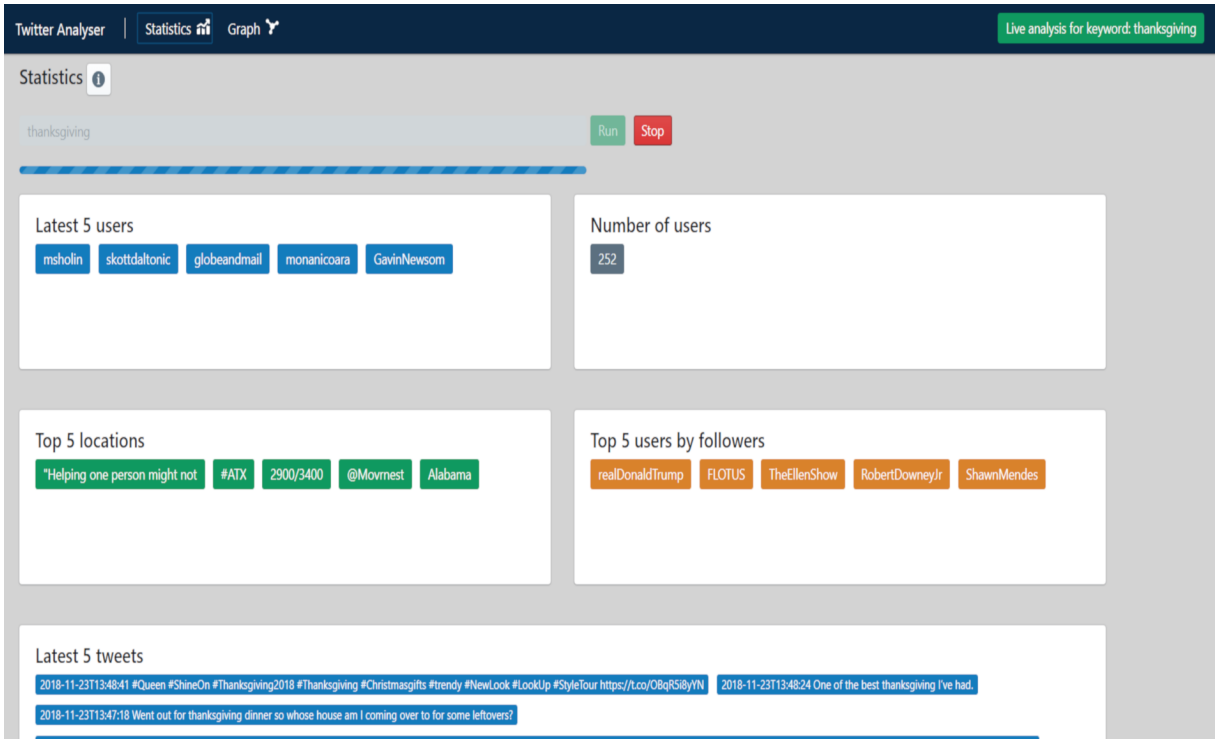


Rysunek 6.1: Ekran startowy aplikacji *Twitter Analyser* [materiały własne].

Strona Statistics

Pierwsza ze wspomnianych funkcjonalności jest dostępna na stronie *Statistics*, której zrzut ekranu jest widoczny na rysunku 6.2.. Interfejs użytkownika na tej stronie prezentuje podstawowe statystyki z przeprowadzanej właśnie sesji analizy danych. Są to:

- nazwy ostatnich pięciu użytkowników,
- liczba wszystkich użytkowników,
- nazwy pięciu najczęściej pojawiających się lokalizacji,
- nazwy pięciu użytkowników z największą liczbą osób śledzących ich profil tzw. followersów,
- treść pięciu ostatnich wiadomości tzw. tweetów,
- treść pięciu ostatnich wiadomości przekazywanych dalej tzw. retweetów.



Rysunek 6.2: Interfejs graficzny strony *Statistics* podczas przetwarzania postów zawierających w treści wyrażenie *thanksgiving* związane ze Świętem Dziękczynienia obchodzonym w Stanach Zjednoczonych [materiały własne].

Rozpoczęcie przetwarzania danych następuje w momencie kiedy po wpisaniu przez użytkownika interesującego go słowa kluczowego uruchomi on wspomniany proces przyciskiem Run. Konsekwencją tego jest skonfigurowanie oraz uruchomienie Apache Spark podłączonego do strumienia danych Twitter Streaming API. Aby uzyskać dostęp do tej porcji danych należy zarejestrować wcześniej aplikację i dostać od serwisu Twitter poświadczenia to umożliwiające. Są to unikatowe: *consumer key*, *consumer secret*, *access token* i *access token secret*. Fragment konfiguracji Apache Spark zamieszczono na rysunku 6.3..

Na podstawie przeprowadzonych testów stwierdzono, że czas trwania pojedynczego okna czasowego powinien wynieść 10 sekund. Jest to czas, który pozwala przetworzyć odebrane wiadomości i nie powoduje opóźnień.

```
private Configuration createConfiguration(TwitterCredentials credentials) {
    return new ConfigurationBuilder()
            .setDebugEnabled(false)
            .setOAuthConsumerKey(credentials.getConsumerKey())
            .setOAuthConsumerSecret(credentials.getConsumerSecret())
            .setOAuthAccessToken(credentials.getAccessToken())
            .setOAuthAccessTokenSecret(credentials.getAccessTokenSecret())
            .build();
}
```

Rysunek 6.3: Fragment konfiguracji narzędzia *Apache Spark* [materiały własne].

Mechanizm przetwarzania danych, którego kod jest widoczny na rysunkach 6.4 oraz 6.5, polega w pierwszej kolejności na sprawdzeniu czy były już przeprowadzane sesje analizy danych z użyciem wybranego słowa kluczowego. Jeśli tak było to wyniki obecnie uruchomionej sesji będą połączone z wynikami z historycznych danych powiązanych z zapisaną encją Keyword. Kolejnym krokiem jest filtracja strumienia danych pod kątem zawartości słowa kluczowego w treści postów. Wielkość liter słowa kluczowego wybranego przez użytkownika aplikacji Twitter Analyser nie ma znaczenia. Potem przefiltrowany strumień jest zamieniany na strumień par *<User, Status>*, gdzie *User* i *Status* odpowiadają klasom implementującym interfejsy opisane w rozdziale 2.. Końcowym etapem jest zapis informacji o użytkowniku oraz jego wiadomości na podstawie wspomnianego strumienia. Jeśli użytkownik pojawił się wcześniej podczas uruchamianych sesji to właśnie przetwarzana wiadomość zostanie dodana do wcześniejszych informacji o nim zapisanej w encji TwitterUser. Status umożliwia m.in. odczytanie informacji czy wiadomość jest wiadomością własną użytkownika czyli tzw. tweet czy podaną dalej tzw. retweet. Stąd biorą się dwa typy relacji występujące w obowiązującym w tworzonej aplikacji modelu danych. Przygotowanie relacji między węzłami wykorzystuje informacje o słowie kluczowym, użytkowniku którego post jest obecnie przetwarzany oraz jego statusie. Metoda odpowiedzialna za przygotowanie relacji typu RetweetedToRelation posługuje się rekurencją, czyli wywoływaniem samej siebie, z powodu możliwych przypadków przekazywania dalej przekazanej już przez innego użytkownika wiadomości.

Wspomniane typy relacji posiadają odpowiadające im encje InterestedInRelation oraz RetweetedToRelation widoczne na rysunkach 6.6 i 6.7.. Odpowiadają one połączeniom między węzłami. W przypadku pierwszej z nich jest to powiązanie węzłów typu TwitterUser i Keyword. Jest ono skierowane ku temu drugiemu. Drugi typ relacji odpowiada połączeniu pomiędzy węzłami typu TwitterUser i jest skierowane ku użytkownikowi, którego wiadomość została przesłana dalej. Graficzną reprezentację modelu danych zaprezentowano na rysunku 6.10..

Informacje o przetwarzanych wiadomościach i użytkownikach pobierane są z grafowej bazy danych, a następnie przesyłane z serwera do przeglądarki internetowej co 10 s za pomocą dwukierunkowego kanału komunikacji o nazwie *WebSocket*, który wykorzystuje protokół TCP [25]. Rozpoczęcie przesyłania danych jest inicjowane przez przeglądarkę internetową. Następnie pomiędzy klientem a serwerem następuje nawiązanie komunikacji. Serwer wysyła statystyki z przeprowadzanej właśnie sesji analizy danych co ustalony czas bez ciągłego wymuszania tej akcji ze strony klienta. Konfigurację po stronie serwera aplikacyjnego zaprezentowano na rysunkach 6.6 oraz 6.7..

Strona Graph

Druga z głównych funkcjonalności programu czyli analiza zapisanych danych pod względem wydźwięku zamieszczanych opinii oraz powiązań między użytkownikami jest dostępna na stronie *Graph*, której zrzut ekranu widoczny jest na rysunku 6.13.. Interfejs użytkownika na tej

```
public void processData(JavaReceiverInputDStream<Status> inputStream, String keywordName) {
    String finalKeywordName = keywordName.toLowerCase();
    Keyword keyword = keywordRepository.findByName(finalKeywordName)
            .orElseGet(() -> {
                Keyword k = new Keyword(finalKeywordName);
                keywordRepository.save(k);
                return k;
            });
    JavaDStream<Status> filteredDStream = inputStream.filter(status ->
            StringUtils.containsIgnoreCase(status.getText(), finalKeywordName)
    );
    JavaPairDStream<User, Status> userStatusStream = filteredDStream.mapToPair(
            status -> new Tuple2<>(status.getUser(), status)
    );
    userStatusStream.foreachRDD((VoidFunction<JavaPairRDD<User, Status>>) pairRDD ->
            pairRDD.foreach((VoidFunction<Tuple2<User, Status>>) t -> {
                User user = t._1();
                Status status = t._2();
                TwitterUser twitterUser = twitterUserRepository.findByUserId(user.getId());
                if (twitterUser == null) {
                    twitterUser = new TwitterUser(user);
                }

                if (status.getRetweetedStatus() != null) {
                    prepareRetweetedToRelation(keyword, twitterUser, status);
                } else {
                    prepareInterestedInRelation(keyword, twitterUser, status);
                }
            })
    );
}
```

Rysunek 6.4: Przetwarzanie danych ze strumienia udostępnianego przez serwis społecznościowy *Twitter* [materiały własne].

stronie prezentuje szczegółowy wykres węzłów połączonych relacjami.

Rozpoczęcie analizowania danych następuje w momencie wyboru słowa kluczowego jakie zostało podane podczas sesji na stronie Statistics. Po kliknięciu przycisku Search następuje przesłanie polecenia do serwera o wyszukanie w bazie danych wszystkich danych będących w korelacji z tym słowem kluczowym. Kolejnym krokiem jest przetworzenie danych i zapisanie ich do modelu, z którego będzie mogła skorzystać przeglądarka. Tak przygotowane dane są przesyłane do klienta, a następnie wyświetlone w postaci wykresu węzłów powiązanych relacjami. Kolor każdego z węzłów jest określany na podstawie sentymentu obliczonego z postów, które zamieścił użytkownik. Sentyment bardzo pozytywny jest określany kolorem ciemnozielonym, pozytywny zielonym, neutralny szarym, negatywny pomarańczowym, a bardzo negatywny czerwonym. Jeśli przynajmniej jeden z postów użytkownika nie jest napisany w języku angielskim to dla takiego profilu nie jest obliczany wydźwięk wypowiedzi i kolor węzła jest biały. Ponad wykresem widnieją statystyki pokazujące ilu użytkowników jest dostępnych w bieżącej analizie oraz ich podział procentowy i liczbowy ze względu na sentyment. Istnieje także możliwość uzyskania szczegółowych informacji o użytkowniku poprzez kliknięcie w węzeł na wykresie. Zrzut ekranu okienka typu pop-up zawierającego te dane zaprezentowano na rysunku 6.14.. Informacje dotyczące relacji łączącej węzły są dostępne poprzez kliknięcie w nie. Zrzut ekranu okienka znajduje się na rysunku 6.15.. W obu przypadkach dane są pobierane z bazy dopiero w momencie kliknięcia. Optymalizacja ta została wykonana aby nie przesyłać niepotrzebnych informacji i nie obciążać mechanizmy odpowiadające za wyświetlenie wykresu.

```
private void prepareInterestedInRelation(Keyword keyword, TwitterUser twitterUser, Status status) {
    InterestedInRelation interestedInRelation = new InterestedInRelation(keyword, twitterUser, status);
    twitterUser.addInterestedInRelation(interestedInRelation);
    twitterUserRepository.save(twitterUser);
}

private void prepareRetweetedToRelation(Keyword keyword, TwitterUser retweeter, Status status) {
    Status retweetedStatus = status.getRetweetedStatus();
    TwitterUser twitterUser = twitterUserRepository.findByUserId(retweetedStatus.getUser().getId());
    if (twitterUser == null) {
        twitterUser = new TwitterUser(retweetedStatus.getUser());
    }
    RetweetedToRelation retweetedToRelation = new RetweetedToRelation(retweeter, twitterUser, status);
    retweeter.addRetweetedToRelation(retweetedToRelation);
    twitterUserRepository.save(retweeter);
    if (retweetedStatus.getRetweetedStatus() != null) {
        prepareRetweetedToRelation(keyword, retweeter, retweetedStatus);
    }
    prepareInterestedInRelation(keyword, twitterUser, retweetedStatus);
}
```

Rysunek 6.5: Przygotowywanie i zapis relacji pomiędzy użytkownikami [materiały własne].

Mechanizm przetwarzania danych, którego część kodu jest widoczna na rysunku 6.16., polega w pierwszej kolejności na pobraniu z grafowej bazy informacji powiązanych z wybranym słowem kluczowym. W następnym kroku kolekcja obiektów TwitterUser jest mapowana na kolekcję obiektów typu Node. Podczas wykonywania operacji mapowania następuje wspomniane sprawdzenie czy user napisał wszystkie posty w języku angielskim. Taka informacja jest dostępna dla każdego posta podczas przetwarzania danych ze strumienia danych serwisu Twitter i wówczas zapisywana do bazy. Sentyment każdego z postów jest obliczany niezależnie, a następnie aplikacja oblicza średnią arytmetyczną i przyporządkowuje wydźwięk z pięciostopniowej skali.

Napisany algorytm obliczania sentymentu rozpoczyna swoje działanie od usunięcia z treści posta słów z prefixem '@' oraz adresów internetowych, ponieważ w innym wypadku zaburzałoby to działanie tego algorytmu. Tak przygotowany tweet przekazywany jest do mechanizmów biblioteki Stanford NLP, która dzieli go na zdania. Dla każdego ze zdań obliczany jest sentyment z wykorzystaniem wytrenowej sieci neuronowej dostarczanej przez tą bibliotekę. Wydźwięk tekstu obliczany jest jako średnia arytmetyczna sentymentu każdego ze zdań.

Ogólne informacje o aplikacji

Użytkownik może korzystać ze stron aplikacji w niezależny sposób. Podczas uruchomienia przetwarzania danych na stronie Statistics można korzystać ze strony Graph i na odwrót. Umożliwia to m.in. analizowanie zgromadzonych danych na stronie Graph tuż po ich zapisaniu przez część serwera powiązaną ze stroną Statistics.

Zarówno strona Graph jak i Statistics posiadają pomoc, która ułatwia korzystanie z aplikacji nowym użytkownikom. Zrzut ekranu okienka typu pop-up z opisanym korzystaniem z funkcjonalności na stronie Graph zamieszczono na rysunku 6.17..

Aplikacja korzysta z ogólnodostępnych danych serwisu Twitter. Pozwala zaobserwować korelacje między użytkownikami oraz wydźwięk ich wypowiedzi, które to informacje byłyby trudne do przeanalizowania przez człowieka z powodu dużego wolumenu danych, ich złożoności oraz zmienności w czasie.

Odporność na błędy aplikacji Twitter Analyser zależy w głównej mierze od mechanizmów

narzędzi z których ona korzysta. Nawet nieprawidłowe zachowanie jednego z nich, które może doprowadzić do zatrzymania działania aplikacji, nie wpłynie jednak na już przetworzone i zapisane dane. Ponadto strona Graph może działać bez połączenia z internetem.

System napisany na potrzeby tej pracy dyplomowej spełnia wszystkie wymagania funkcjonalne i niefunkcjonalne postawione w rozdziale 4..

```java
@NodeEntity
@NoArgsConstructor
@Getter
public class TwitterUser implements Serializable {

    private LocalDate createdAt;
    private String description;
    private int favouritesCount;
    private int followersCount;
    private int friendsCount;
    @Id
    @GeneratedValue
    private Long id;
    private long userId;
    private String language;
    private String location;
    private String screenName;
    private String timeZone;

    @Relationship(type = "RETWEETED_TO")
    private List<RetweetedToRelation> retweetedToRelations = new ArrayList<>();

    @Relationship(type = "INTERESTED_IN")
    private List<InterestedInRelation> interestedInRelations = new ArrayList<>();

    public TwitterUser(User user) {
        this.createdAt = new Date(user.getCreatedAt().getTime()).toLocalDate();
        this.description = user.getDescription();
        this.favouritesCount = user.getFavouritesCount();
        this.followersCount = user.getFollowersCount();
        this.friendsCount = user.getFollowersCount();
        this.userId = user.getId();
        this.language = user.getLang();
        this.location = user.getLocation();
        this.screenName = user.getScreenName();
        this.timeZone = user.getTimeZone();
    }

    public void addRetweetedToRelation(RetweetedToRelation relation) {
        this.retweetedToRelations.add(relation);
    }

    public void addInterestedInRelation(InterestedInRelation relation) {
        this.interestedInRelations.add(relation);
    }

}
```

Rysunek 6.6: Kod encji *TwitterUser* [materiały własne].

```java
@NodeEntity
@NoArgsConstructor
@Getter
public class Keyword implements Serializable {

    @Id
    @GeneratedValue
    private Long id;
    private String name;

    @Relationship(type = "INTERESTED_IN", direction = Relationship.INCOMING)
    private List<TwitterUser> twitterUsers = new ArrayList<>();

    public Keyword(String name) { this.name = name; }

}
```

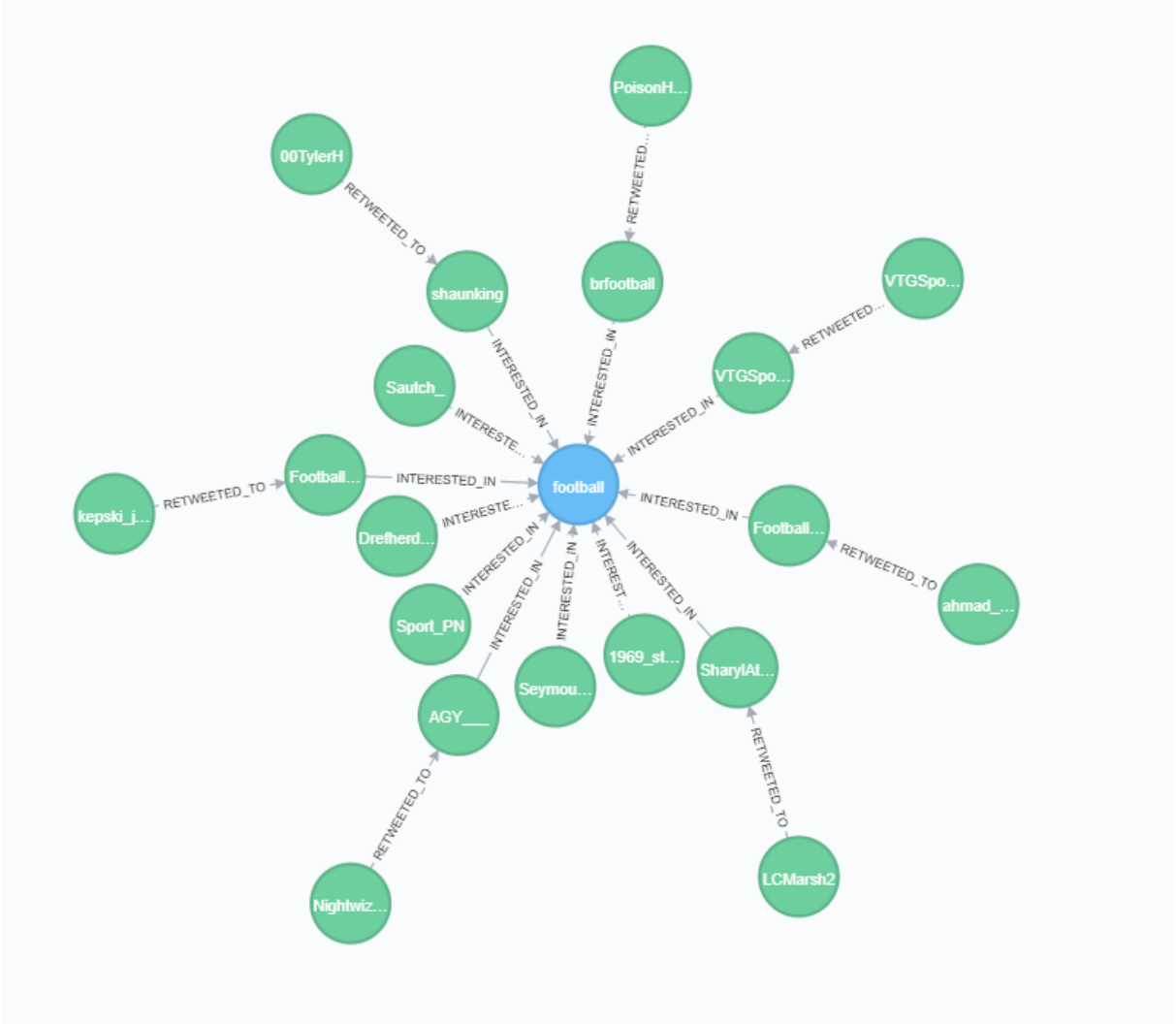
Rysunek 6.7: Kod encji *Keyword* [materiały własne].

```java
@RelationshipEntity(type = "INTERESTED_IN")
@NoArgsConstructor
@Getter
public class InterestedInRelation implements Serializable {

    private LocalDateTime createdAt;
    private GeoLocation geoLocation;
    private Long id;
    @EndNode
    private Keyword keyword;
    private String language;
    private String location;
    private Place place;
    private String text;
    @StartNode
    private TwitterUser twitterUser;

    public InterestedInRelation(Keyword keyword, TwitterUser twitterUser, Status status) {
        this.createdAt = new Timestamp(status.getCreatedAt().getTime()).toLocalDateTime();
        this.geoLocation = status.getGeoLocation();
        this.keyword = keyword;
        this.language = status.getLang();
        this.location = status.getUser().getLocation();
        this.place = status.getPlace();
        this.text = status.getText();
        this.twitterUser = twitterUser;
    }

}
```

Rysunek 6.8: Kod encji relacji *InterestedInRelation* [materiały własne].

```java
@RelationshipEntity(type = "RETWEETED_TO")
@NoArgsConstructor
@Getter
public class RetweetedToRelation implements Serializable {

    private LocalDateTime createdAt;
    private GeoLocation geoLocation;
    private Long id;
    private String language;
    private String location;
    private Place place;
    @StartNode
    private TwitterUser retweeter;
    private String text;
    @EndNode
    private TwitterUser twitterUser;

    public RetweetedToRelation(TwitterUser retweeter, TwitterUser twitterUser, Status status) {
        this.createdAt = new Timestamp(status.getCreatedAt().getTime()).toLocalDateTime();
        this.geoLocation = status.getGeoLocation();
        this.language = status.getLang();
        this.location = status.getUser().getLocation();
        this.place = status.getPlace();
        this.text = status.getText();
        this.twitterUser = twitterUser;
        this.retweeter = retweeter;
    }

}
```

Rysunek 6.9: Kod encji relacji *RetweetedToRelation* [materiały własne].



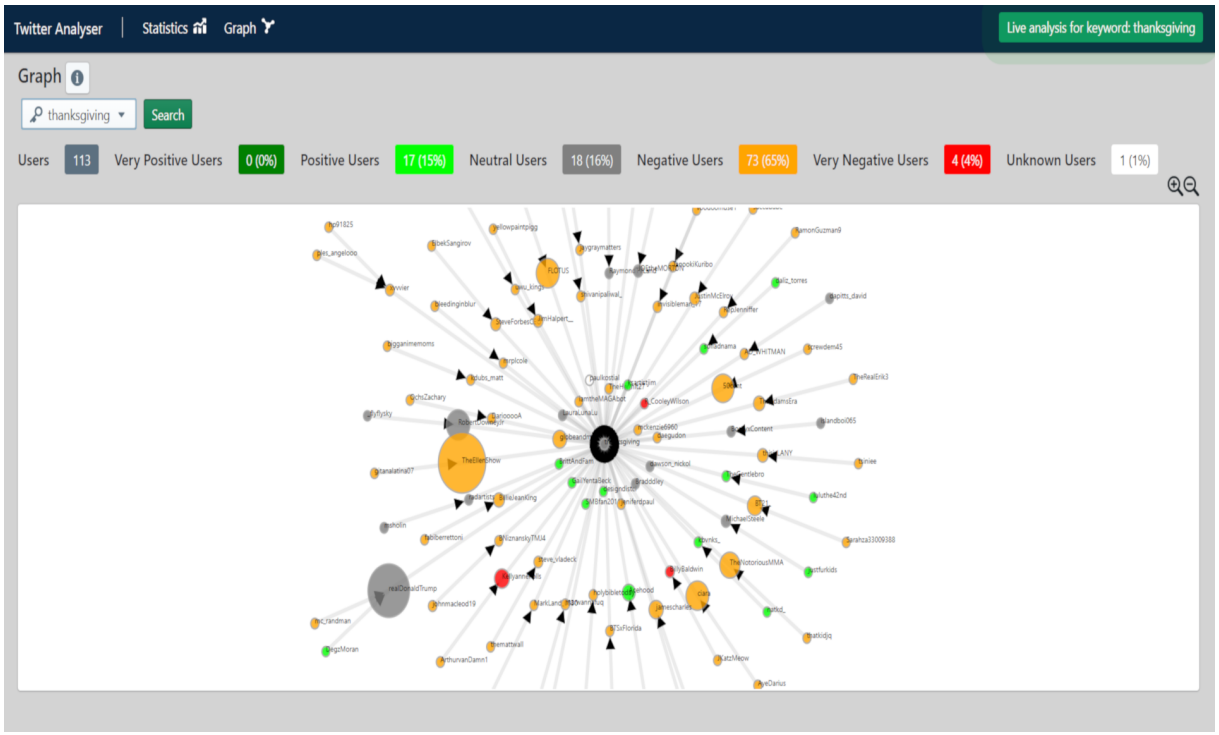
Rysunek 6.10: Graficzna reprezentacja modelu danych. Niebieski węzeł jest typu *Keyword*, a pozostałe zielone typu *TwitterUser*. Pomiędzy węzłami występują relacje typu *InterestedInRelation* oraz *RetweetedToRelation* [materiały własne].

```java
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {

    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker( ...destinationPrefixes: "/statistics");
        config.setApplicationDestinationPrefixes("/app");
    }

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry
                .addEndpoint( ...strings: "/twitter-analyser")
                .setAllowedOrigins("*")
                .withSockJS();
    }

}
```

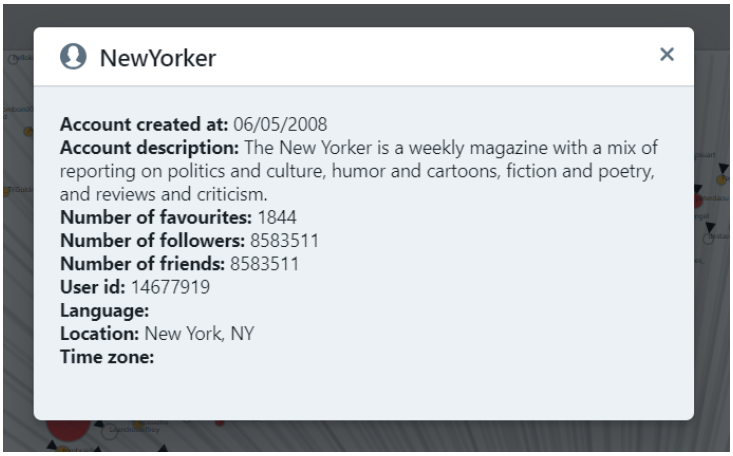
Rysunek 6.11: Konfiguracja dwukierunkowego kanału komunikacji *WebSocket* [materiały własne].

```java
@Scheduled(fixedRate = 10000)
public void getStatisticsData() {
    if (contextInitialized && apacheSparkStarted) {
        log.info("Sending statistics data...");
        simpMessagingTemplate.convertAndSend(
                destination: "/statistics/statisticsData",
                statisticsService.getStatistics(keywordName)
        );
    }
}
```

Rysunek 6.12: Konfiguracja dwukierunkowego kanału komunikacji *WebSocket* [materiały własne].



Rysunek 6.13: Interfejs graficzny strony *Graph* podczas analizy danych związanych ze Świętem Dziękczynienia obchodzonym w Stanach Zjednoczonych [materiały własne].

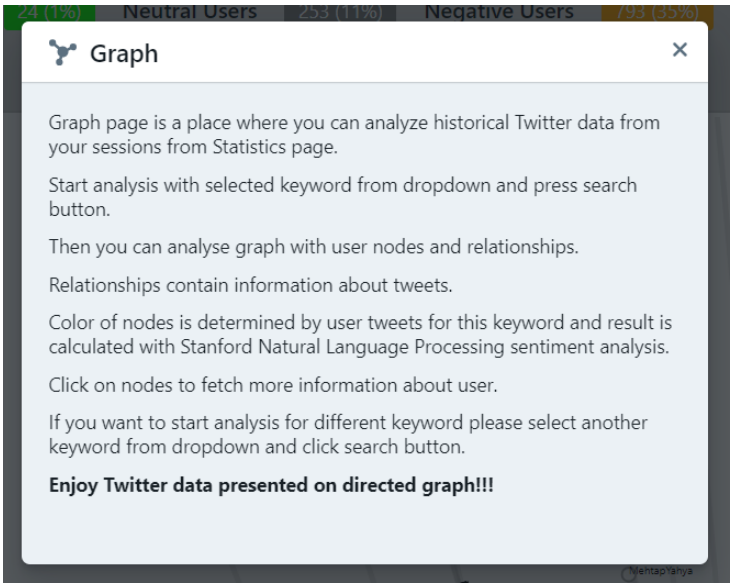


Rysunek 6.14: Okienko typu pop-up służące do wyświetlenia informacji o użytkowniku [materiały własne].

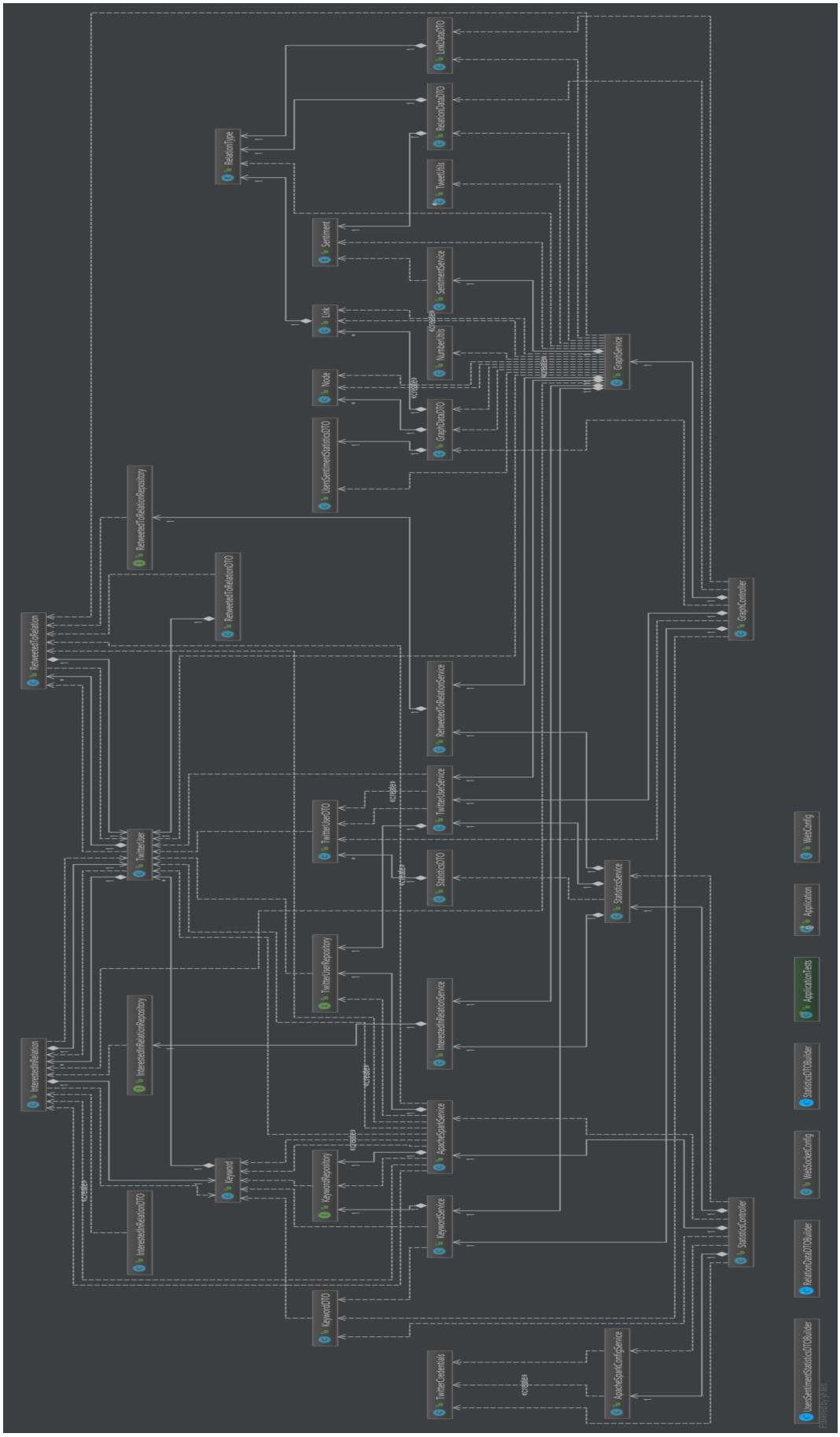
Rysunek 6.15: Okienko typu pop-up służące do wyświetlenia informacji o relacji łączącej węzły [materiały własne].

```
private Set<Node> computeRetweetersNodes(Set<TwitterUser> retweeters, String keyword) {
    return retweeters.stream()
            .map(retweeter -> new Node(
                    retweeter.getScreenName(),
                    getRetweetedToNodeColor(retweeter.getRetweetedToRelations(), keyword),
                    retweeter.getFollowersCount()
            )
            )
            .collect(Collectors.toSet());
}

private String getRetweetedToNodeColor(List<RetweetedToRelation> retweetedToRelations, String keyword) {
    if (!allEnglishRetweetedToRelations(retweetedToRelations, keyword)) {
        return Sentiment.UNKNOWN.getColor();
    }
    List<Integer> sentiments = retweetedToRelations
            .stream()
            .filter(retweetedToRelation -> StringUtils.containsIgnoreCase(retweetedToRelation.getText(), keyword))
            .map(retweetedToRelation -> {
                String text = retweetedToRelation.getText();
                return sentimentService.computeSentiment(TweetUtils.cleanTweet(text));
            })
            .collect(Collectors.toList());
    if (sentiments.isEmpty()) {
        return Sentiment.NEUTRAL.getColor();
    }
    return sentimentService.getSentiment(NumberUtils.calculateAverage(sentiments)).getColor();
}

private boolean allEnglishRetweetedToRelations(List<RetweetedToRelation> retweetedToRelations, String keyword) {
    return retweetedToRelations
            .stream()
            .filter(retweetedToRelation -> StringUtils.containsIgnoreCase(retweetedToRelation.getText(), keyword))
            .allMatch(retweetedToRelation -> ENGLISH_LANGUAGE_ABBREVIATION.equals(retweetedToRelation.getLanguage()));
}
```

Rysunek 6.16: Przygotowanie danych dotyczących użytkowników publikujących wiadomości innych autorów. [materiały własne]



Rysunek 6.17: Okienko typu pop-up służące do wyświetlenia krótkiej informacji o funkcjonalności strony *Graph* [materiały własne].



Rysunek 6.18: Diagram klas aplikacji *Twitter Analyser* [materiały własne].

Rozdział 7

Badania i wnioski

Rozdział 8

Podsumowanie

Bibliografia

[1] https://en.wikipedia.org/wiki/Twitter [Dostęp 4 listopada 2018]

[2] https://medium.com/@ssola/playing-with-twitter-streaming-api-b1f8912e50b0 [Dostęp 11 listopada 2018]

[3] https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html [Dostęp 11 listopada 2018]

[4] http://140dev.com/twitter-api-programming-tutorials/aggregating-tweets-search-api-vs-stre [Dostęp 11 listopada 2018]

[5] https://www.talkwalker.com/blog/5-free-twitter-analytics-tools-with-views-from-experts [Dostęp 19 listopada 2018]

[6] http://twitter4j.org/javadoc/twitter4j/User.html [Dostęp 12 listopada 2018]

[7] http://twitter4j.org/javadoc/twitter4j/Status.html [Dostęp 12 listopada 2018]

[8] https://pl.wikipedia.org/wiki/Bieg_maratoński [Dostęp 24 listopada 2018]

[9] A. Mykowiecka, *"Inżynieria lingwistyczna: komputerowe przetwarzanie tekstów w języku naturalnym"*, Wydawnictwo Polsko-Japońskiej Wyższej Szkoły Technik Komputerowych, 2007, s. 127 -170.

[10] https://en.wikipedia.org/wiki/Bag-of-words_model [Dostęp 2 grudnia 2018]

[11] https://en.wikipedia.org/wiki/Vector_space_model [Dostęp 11 grudnia 2018]

[12] V. Dixit, A. Saroliya, *"A semantic Vector Space Model approach for sentiment analysis"*, International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 8, August 2013.

[13] F. Zhou, F. Zhang, G. Yang, *"Graph-based text representation model and its realization"*, Natural Language Processing and Knowledge Engineering, 2010.

[14] https://blog.brand24.pl/co-to-jest-analiza-sentymentu-oraz-jak-mozesz-ja-wykorzystac/ [Dostęp 24 listopada 2018]

[15] https://pl.wikipedia.org/wiki/Przetwarzanie_języka_naturalnego [Dostęp 24 listopada 2018]

[16] M. Desai, M. A. Mehta, *"Techniques for Sentiment Analysis of Twitter Data: A Comprehensive Survey"*, International Conference on Computing, Communication and Automation (ICCCA2016), April 2016.

[17] B. Pang, L. Lee, *"Opinion mining and sentiment analysis"* Foundations and Trends in Information Retrieval, 2008, s. 1-2.

[18] Jak to działa? Era Big Data, TVP VOD.

[19] https://pl.wikipedia.org/wiki/System_czasu_rzeczywistego [Dostęp 10 grudnia 2018]

[20] https://www.baeldung.com/java-in-2018 [Dostęp 6 stycznia 2019]

[21] C. de Sousa Antonio, *"React dla zaawansowanych."*, Helion, 2017.

[22] https://spark.apache.org/docs/2.1.1/streaming-programming-guide.html [Dostęp 11 grudnia 2018]

[23] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, *"Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank"*, Conference on Empirical Methods in Natural Language Processing, Stanford University, 2013.

[24] https://nlp.stanford.edu/sentiment/treebank.html [Dostęp 11 grudnia 2018]

[25] https://pl.wikipedia.org/wiki/WebSocket [Dostęp 6 stycznia 2019]