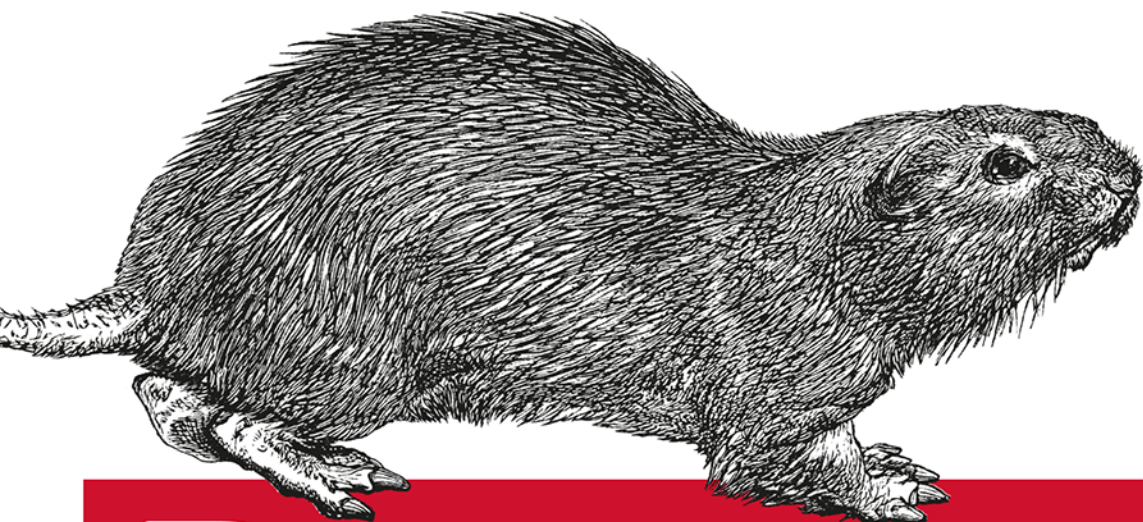


O'REILLY®

Wydanie III



Data Mining

EKSPLORACJA DANYCH W SIECIACH SPOŁECZNOŚCIOWYCH

Helion 

Matthew A. Russell
Mikhail Klassen

Tytuł oryginału: Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More, 3rd Edition

Tłumaczenie: Radosław Meryk

ISBN: 978-83-283-5554-5

© 2019 Helion S.A.

Authorized Polish translation of the English edition of Mining the Social Web, 3E ISBN 9781491985045 © 2019 Matthew A. Russell and Mikhail Klassen

This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Helion SA dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Helion SA nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:

<ftp://ftp.helion.pl/przyklady/datam3.zip>

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/datam3>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Przedmowa	11
-----------------	----

Część I. Przewodnik po sieciach społecznościowych

Wstęp	25
1. Eksploracja Twittera: odkrywanie trendów, dowiadywanie się, o czym się rozmawia, i trochę więcej	27
1.1. Przegląd	27
1.2. Dlaczego Twitter to jest „to”?	28
1.3. Odkrywanie API Twittera	30
1.3.1. Podstawowa terminologia związana z Twitterem	30
1.3.2. Tworzenie połączenia z API Twittera	33
1.3.3. Odkrywanie trendów	36
1.3.4. Wyszukiwanie tweetów	40
1.4. Analiza 140 (lub więcej) znaków	46
1.4.1. Wyodrębnianie podmiotów z tweetów	47
1.4.2. Analizowanie tweetów i występujących w nich podmiotów z wykorzystaniem analizy częstości	49
1.4.3. Obliczanie różnorodności leksykalnej tweetów	51
1.4.4. Badanie wzorców w retweetach	53
1.4.5. Wizualizacja danych częstości za pomocą histogramów	55
1.5. Uwagi końcowe	59
1.6. Zalecane ćwiczenia	60
1.7. Zasoby online	61

2. Eksploracja Facebooka: analizowanie fanpage'y, znajomości i więcej	63
2.1. Przegląd	64
2.2. Interfejs API Graph Facebooka	64
2.2.1. Wprowadzenie do API Graph	66
2.2.2. Protokół Open Graph	70
2.3. Analiza połączeń grafu społecznościowego	75
2.3.1. Analizowanie stron Facebooka	78
2.3.2. Manipulowanie danymi z wykorzystaniem pakietu pandas	88
2.4. Uwagi końcowe	95
2.5. Zalecane ćwiczenia	96
2.6. Zasoby online	96
3. Eksploracja Instagrama: komputerowy wzrok, sieci neuronowe, rozpoznawanie obiektów i wykrywanie twarzy	99
3.1. Przegląd	100
3.2. Poznawanie API Instagrama	101
3.2.1. Tworzenie żądań do API Instagrama	101
3.2.2. Odczytywanie własnego kanału na Instagramie	103
3.2.3. Pobieranie medium według hashtagu	105
3.3. Anatomia posta na Instagramie	105
3.4. Szybki kurs na temat sztucznych sieci neuronowych	108
3.4.1. Trening sieci neuronowej pod kątem „oglądania” zdjęć	109
3.4.2. Rozpoznawanie cyfr pisanych odręcznie	111
3.4.3. Rozpoznawanie obiektów na zdjęciach przy użyciu wstępnie przeszkolonych sieci neuronowych	116
3.5. Wykorzystanie sieci neuronowych do postów na Instagramie	119
3.5.1. Oznaczanie zawartości obrazu	119
3.5.2. Wykrywanie twarzy na zdjęciach	121
3.6. Uwagi końcowe	122
3.7. Zalecane ćwiczenia	123
3.8. Zasoby online	124
4. Eksploracja sieci LinkedIn: stanowiska, współpracownicy i nie tylko	127
4.1. Przegląd	128
4.2. Poznawanie API LinkedIna	128
4.2.1. Tworzenie żądań do API LinkedIn	129
4.2.2. Pobieranie połączeń LinkedIn w pliku CSV	132

4.3. Krótki kurs grupowania danych	132
4.3.1. Normalizacja danych w celu umożliwienia analizy	135
4.3.2. Mierzenie podobieństwa	145
4.3.3. Algorytmy klasteryzacji	147
4.4. Uwagi końcowe	161
4.5. Zalecane ćwiczenia	161
4.6. Zasoby online	162
5. Eksploracja danych z plików tekstowych: obliczanie podobieństwa dokumentów, wyodrębnianie kolokacji i inne	163
5.1. Przegląd	164
5.2. Pliki tekstowe	164
5.3. Wprowadzenie do TF-IDF	166
5.3.1. Częstość terminu	166
5.3.2. Odwrotna częstość dokumentu	168
5.3.3. TF-IDF	169
5.4. Odpytywanie danych w języku naturalnym za pomocą TF-IDF	172
5.4.1. Natural Language Toolkit — wprowadzenie	172
5.4.2. Zastosowanie współczynnika TF-IDF do języka naturalnego	176
5.4.3. Wyszukiwanie podobnych dokumentów	177
5.4.4. Analiza bigramów w języku naturalnym	184
5.4.5. Refleksje na temat analizy danych języka naturalnego	193
5.5. Uwagi końcowe	194
5.6. Zalecane ćwiczenia	195
5.7. Zasoby online	195
6. Eksploracja stron internetowych: przetwarzanie języka naturalnego w celu zrozumienia języka ludzkiego, tworzenie podsumowań postów na blogu i inne	197
6.1. Przegląd	198
6.2. Scraping, parsowanie i crawling stron internetowych	199
6.2.1. Przeszukiwanie wszcz w crawlingu stron internetowych	202
6.3. Odkrywanie semantyki przez dekodowanie składni	205
6.3.1. Przetwarzanie języka naturalnego krok po kroku	207
6.3.2. Wykrywanie zdań w danych w języku naturalnym	210
6.3.3. Tworzenie streszczeń dokumentów	214
6.4. Zmiana paradygmatu. Analiza obiektów	222
6.4.1. Podsumowania danych w języku naturalnym	226

6.5. Jakość analiz do przetwarzania danych w języku naturalnym	230
6.6. Uwagi końcowe	234
6.7. Zalecane ćwiczenia	234
6.8. Zasoby online	235
7. Eksploracja skrzynek pocztowych: analiza, kto rozmawia z kim, o czym, jak często i nie tylko	237
7.1. Przegląd	238
7.2. Uzyskiwanie i przetwarzanie korpusu danych pocztowych	239
7.2.1. Uniksowe skrzynki pocztowe	239
7.2.2. Pobieranie danych Enron	243
7.2.3. Konwersja korpusu poczty na uniksowy format mbox	245
7.2.4. Konwertowanie uniksowych skrzynek pocztowych na obiekty DataFrame biblioteki pandas	247
7.3. Analiza korpusu Enron	249
7.3.1. Zapytania według zakresu dat (godzin)	250
7.3.2. Analiza wzorców w komunikacji nadawca-odbiorca	253
7.3.3. Wyszukiwanie wiadomości e-mail według słów kluczowych	257
7.4. Analiza własnych danych pocztowych	258
7.4.1. Dostęp do Twojej skrzynki Gmail za pomocą OAuth	260
7.4.2. Pobieranie i parsowanie wiadomości e-mail	262
7.4.3. Wizualizacja wzorców w e-mailu za pomocą frameworka Immersion	264
7.5. Uwagi końcowe	265
7.6. Zalecane ćwiczenia	265
7.7. Zasoby online	266
8. Eksploracja serwisu GitHub: badanie nawyków podczas współtworzenia oprogramowania, tworzenie grafów zainteresowań i nie tylko	269
8.1. Przegląd	270
8.2. Odkrywanie API GitHuba	270
8.2.1. Tworzenie połączenia do API serwisu GitHub	272
8.2.2. Tworzenie żądań do API GitHuba	275
8.3. Modelowanie danych za pomocą grafów właściwości	277
8.4. Analiza grafów zainteresowań serwisu GitHub	280
8.4.1. „Wysiewanie” grafu zainteresowań	281
8.4.2. Obliczanie miar centralności grafu	284
8.4.3. Rozszerzanie grafu zainteresowań z wykorzystaniem krawędzi „śledzi” dla użytkowników	287

8.4.4. Używanie węzłów jako punktów przestawnych w celu tworzenia bardziej wydajnych zapytań	296
8.4.5. Wizualizacja grafów zainteresowań	301
8.5. Uwagi końcowe	303
8.6. Zalecane ćwiczenia	304
8.7. Zasoby online	305

Część II. Twitter. Receptury

9. Twitter. Receptury	309
9.1. Dostęp do interfejsu API Twittera dla celów programistycznych	310
9.2. Wykorzystanie OAuth w celu uzyskania dostępu do interfejsu API Twittera dla aplikacji produkcyjnych	311
9.3. Odkrywanie trendów	315
9.4. Wyszukiwanie tweetów	316
9.5. Konstruowanie wygodnych wywołań funkcji	318
9.6. Zapisywanie i przywracanie danych JSON z wykorzystaniem plików tekstowych	319
9.7. Zapisywanie danych JSON i uzyskiwanie dostępu do nich za pomocą MongoDB	320
9.8. Pobieranie próbek z mechanizmu firehose Twittera za pomocą API Streaming	323
9.9. Pobieranie danych szeregów czasowych	324
9.10. Wyodrębnianie podmiotów z tweetów	326
9.11. Znajdowanie najpopularniejszych tweetów w kolekcji	327
9.12. Znajdowanie najpopularniejszych obiektów w kolekcji tweetów	329
9.13. Tabularyzacja analizy częstości	330
9.14. Znajdowanie użytkowników, którzy retweetowali status	331
9.15. Wyodrębnianie przypisania retweeta	333
9.16. Wykonywanie odpornych na błędy żądań do Twittera	334
9.17. Pobieranie informacji o profilu użytkownika	337
9.18. Wyodrębnianie podmiotów tweeta z dowolnego tekstu	338
9.19. Pobieranie wszystkich znajomych lub obserwatorów użytkownika	339
9.20. Analiza znajomych i obserwatorów użytkownika	341
9.21. Zbieranie tweetów użytkownika	342
9.22. Crawling grafu znajomości	344
9.23. Analiza treści tweetów	346
9.24. Tworzenie streszczeń celów łączy	347
9.25. Analizowanie ulubionych tweetów użytkownika	350
9.26. Uwagi końcowe	352
9.27. Zalecane ćwiczenia	352
9.28. Zasoby online	353

Część III. Załączniki

A	Informacje o maszynie wirtualnej przeznaczonej dla tej książki	357
B	Elementarz OAuth	359
C	Porady i wskazówki na temat Pythona i środowiska Jupyter Notebook	363
	Skorowidz	365

Eksploracja Facebooka: analizowanie fanpage'y, znajomości i więcej

W tym rozdziale zagłębimy się w tematykę platformy Facebook za pośrednictwem jej interfejsu API (Social) Graph i zbadamy niektóre z ogromnych możliwości tego API. Facebook jest bez wątpienia sercem sieci społecznościowych. To, że ponad połowa z dwóch miliardów użytkowników¹ codziennie aktualizuje statusy, publikuje zdjęcia, wymienia się wiadomościami, czatuje w czasie rzeczywistym, sprawdza fizyczne lokalizacje, gra w gry, robi zakupy i wykonuje różne inne czynności, jakie tylko można sobie wyobrazić — jest czymś w rodzaju wielkiego cudu. Z punktu widzenia eksploracji danych sieci społecznościowych bogactwo danych na temat osób, grup i produktów przechowywanych na Facebooku jest ekscytujące. Czysty interfejs API Facebooka oferuje niesamowite możliwości dokonywania syntezy wydobywanych danych w formie informacji (najcenniejszy towar na świecie) i gromadzenia cennych wiadomości. Z drugiej strony z tymi wielkimi możliwościami wiąże się olbrzymia odpowiedzialność. Aby wzmocnić ochronę użytkowników przed nadużywaniem należących do nich danych, Facebook opracował najbardziej zaawansowane mechanizmy kontroli prywatności online (<http://on.fb.me/1a1llg9>).

Warto zauważyć, że chociaż Facebook określa swoją sieć jako graf społecznościowy, to stale przekształca się ona w graf zainteresowań, ponieważ za pośrednictwem stron na Facebooku i zdolności reagowania (na przykład przez kliknięcia linków *Lubię to*) utrzymywane są relacje między ludźmi oraz tym, co ich interesuje. Dlatego coraz częściej Facebook jest określany mianem „grafu społecznych zainteresowań”. W większości przypadków można zaryzykować stwierdzenie, że grafy zainteresowań istnieją niejawnie i można je wywnioskować na podstawie większości źródeł danych społecznościowych. Na przykład w rozdziale 1. podano, że Twitter jest w rzeczywistości grafem zainteresowań ze względu na asymetryczne „śledzenie” (lub, inaczej mówiąc, „zainteresowanie”) relacji pomiędzy ludźmi, miejscami lub przedmiotami. Pojęcie Facebooka jako grafu zainteresowań będzie się pojawiało w tym rozdziale. Do koncepcji jawnego tworzenia grafu zainteresowań z danych społecznościowych powrócimy w rozdziale 8.

¹ Statystyki korzystania z internetu (<http://bit.ly/1a1ljF8>) pokazują, że populacja świata w 2017 r. została oszacowana na około 7,5 miliarda, natomiast szacowana liczba użytkowników internetu wynosi prawie 3,9 miliarda.

W pozostałej części tego rozdziału założono, że masz aktywne konto na Facebooku (<http://on.fb.me/1a1lkcd>). Jest ono niezbędne do uzyskania dostępu do interfejsów API Facebooka. W 2015 r. Facebook wprowadził w swoich interfejsach API zmiany (<http://tcrn.ch/2zFetfo>), które ograniczyły rodzaj danych dostępnych dla podmiotów z zewnątrz. Na przykład za pośrednictwem API nie można już uzyskać dostępu do aktualizacji statusu znajomego ani jego zainteresowań. Ta zmiana była spowodowana obawami o prywatność. Ze względu na wspomniane zmiany w tym rozdziale skupimy się bardziej na tym, jak można wykorzystać API Facebooka do pomiaru zaangażowania użytkowników na stronach o dostępie publicznym, takich jak te, które są tworzone przez firmy komercyjne lub celebrytów. Nawet te dane stają się w coraz większym stopniu poufne, a uzyskanie dostępu do niektórych z tych własności może wymagać zgody za pośrednictwem platformy programistycznej Facebooka.



Należy pamiętać, by pobrać najnowszy, poprawiony kod źródłowy przykładów zamieszczonych w tym rozdziale (oraz we wszystkich pozostałych rozdziałach), który jest dostępny w serwisie GitHub (<http://bit.ly/Mining-the-Social-Web-3E>). Aby w maksymalny sposób skorzystać z przykładowego kodu, pamiętaj również o użyciu wirtualnej maszyny przeznaczonej dla tej książki. Została ona opisana w dodatku A.

2.1. Przegląd

Ponieważ jest to drugi rozdział, pojęcia, które omówimy, będą nieco bardziej złożone niż te z rozdziału 1. Nadal jednak powinny być zrozumiałe dla szerokiego kręgu czytelników. W tym rozdziale omówimy następujące zagadnienia:

- API Graph Facebooka i sposób tworzenia żądań API.
- Protokół Open Graph i jego związek z grafem społecznościowym Facebooka.
- Programowy dostęp do stron publicznych, na przykład tych, których posiadaczami są właściciele popularnych marek i celebryci.
- Wyodrębnianie kluczowych miar społecznościowych, takich jak liczba polubień, komentarzy i udostępnień jako wskaźników zainteresowania odbiorców.
- Manipulowanie danymi za pomocą obiektów DataFrames z pakietu pandas oraz wizualizacja wyników.

2.2. Interfejs API Graph Facebooka

Platforma Facebook jest dojrzałą, solidną i dobrze udokumentowaną bramą prowadzącą do tego, co być może jest najbardziej wszechstronnym i dobrze zorganizowanym magazynem informacji, jaki kiedykolwiek powstał, tak pod względem zakresu, jak i głębokości. Jest obszerna, ponieważ baza jej użytkowników stanowi około jednej czwartej całej populacji świata, i jest głęboka w odniesieniu do ilości informacji, jakich dostarcza na temat każdego z jej użytkowników. Podczas gdy Twitter charakteryzuje się asymetrycznym modelem znajomości, który jest otwarty i bazuje na śledzeniu innych użytkowników bez ich jawnej zgody, model znajomości Facebooka jest symetryczny — aby uzyskać wgląd we wzajemne interakcje i działania, potrzebne jest porozumienie pomiędzy użytkownikami.

Poza tym, o ile praktycznie wszystkie interakcje, z wyjątkiem prywatnych wiadomości pomiędzy użytkownikami na Twitterze, mają status publiczny, o tyle Facebook pozwala na znacznie dokładniejszą kontrolę prywatności. Znajomości na Facebooku mogą być zorganizowane i utrzymywane w postaci list o różnym poziomie widoczności, a poszczególne aktywności mogą być udostępniane wybranym znajomym. Na przykład link lub zdjęcie można udostępnić tylko znajomym z konkretnej listy, a nie wszystkim użytkownikom sieci społecznościowej.

Jedynym sposobem na eksplorację danych z Facebooka jest zarejestrowanie aplikacji i użycie jej jako punktu wejścia do platformy programistycznej Facebooka. Co więcej, aplikacja ma dostęp wyłącznie do tych danych, które użytkownik jawnie udostępnił. Na przykład programista piszący aplikację Facebooka jest użytkownikiem, który loguje się do aplikacji, a aplikacja może uzyskać dostęp do tych danych, które wyraźnie jej wskażesz. W związku z tym jako użytkownik Facebooka możesz traktować aplikację jako jednego ze swoich znajomych na Facebooku. Zarządzasz tym, do czego aplikacja ma dostęp, i możesz w każdej chwili ten dostęp ograniczyć. Dokument *Facebook Platform Policy* (<http://bit.ly/1a1lm3C>) to lektura obowiązkowa dla każdego programisty Facebooka. Zestawiono w nim kompletny zbiór praw i obowiązków dla wszystkich użytkowników Facebooka, a także zaprezentowano ducha i literę prawa obowiązujące programistów Facebooka. Jeśli jeszcze tego nie zrobiłeś, poświęć chwilę na zapoznanie się z zasadami obowiązującymi programistów Facebooka i dodaj do zakładek stronę główną jego programistów (<http://bit.ly/1a1lm3Q>). Jest to bowiem punkt wejścia do platformy Facebook i jej dokumentacji. Pamiętaj również, że interfejsy API mogą się zmieniać. Z powodu obaw dotyczących bezpieczeństwa i prywatności uprawnienia dostępne dla użytkownika podczas eksperymentowania z platformą Facebook są ograniczone. Aby uzyskać dostęp do niektórych funkcji i punktów końcowych API, może być konieczne przesłanie aplikacji w celu jej przejrzania i zatwierdzenia (<http://bit.ly/2vDb2B1>). Jeśli Twoja aplikacja spełnia warunki korzystania z usługi (<http://on.fb.me/1a1lMXM>), nie powinieneś mieć problemów z uzyskaniem zgody na jej wykorzystywanie.



Osoby próbujące eksplorować dane z własnego konta nie powinny mieć problemów z umożliwieniem aplikacji dostępu do wszystkich danych konta. Jednak w przypadku próby stworzenia aplikacji żądającej dostępu do obszerniejszych danych niż te, które są konieczne do wykonania przez nią zadania, istnieje duże prawdopodobieństwo, że użytkownicy nie zaufają jej w takim stopniu (co zresztą jest słuszne).

Sposób uzyskania programowego dostępu do platformy Facebook pokażemy w dalszej części tego rozdziału. Przedtem jednak omówimy szereg przydatnych narzędzi programistycznych (<http://bit.ly/1a1lnVf>), w tym aplikację Graph API Explorer (<http://bit.ly/2jd5Xdq>). Wykorzystamy ją do wstępnego zapoznania się z grafem społecznościowym. Aplikacja ta zapewnia intuicyjny i kompleksowy sposób odpytywania grafu społecznościowego. Po zapoznaniu się ze sposobem działania takiego grafu przetłumaczenie zapytań na kod Pythona w celu automatyzacji i przetwarzania danych przychodzi zupełnie naturalnie. Chociaż w tym rozdziale będziemy sukcesywnie omawiać interfejs API Graph, zachęcamy do wstępnego zapoznania się z nim poprzez lekturę dokumentu *Graph API Overview* (<http://bit.ly/1a1lobU>) — obszernego wstępu do tego API.



Należy zwrócić uwagę, że Facebook zakończył obsługę swojego języka zapytań FQL (ang. *Facebook Query Language*) (<http://bit.ly/1a1lmRd>). Począwszy od 8 sierpnia 2016 r. nie można już używać tych zapytań. Zamiast tego programiści muszą wykonywać wywołania API Graph. Dla tych, którzy wcześniej opracowali aplikację bazującą na FQL-u, Facebook stworzył narzędzie API Upgrade Tool (<http://bit.ly/2MGU7Z9>), którego możesz użyć do aktualizacji.

2.2.1. Wprowadzenie do API Graph

Jak wskazuje jego nazwa, społeczny graf Facebooka jest ogromną strukturą danych (<http://bit.ly/1a1loIX>), reprezentującą interakcje społeczne i składającą się z węzłów i połączeń pomiędzy nimi. API Graph zapewnia podstawowy sposób interakcji z grafem społecznościowym. Najlepszym sposobem na zapoznanie się z tym API jest poświęcenie kilku minut, by poeksperymentować z narzędziem Graph API Explorer (<http://bit.ly/2jd5Xdq>).

Graph API Explorer nie jest zbyt skomplikowanym narzędziem. Posiada możliwość wstępnego wypełniania i debugowania tokena dostępu, ale poza tym jest to zwykła aplikacja Facebooka, która korzysta z tych samych interfejsów API co dowolne inne aplikacje. Graph API Explorer jest przydatny w przypadku, gdy dysponujemy określonym tokenem OAuth powiązanim z konkretnym zestawem autoryzacji dla tworzonej aplikacji i chcemy uruchomić pewne zapytania w ramach eksperymentów lub podczas cyklu debugowania. Powrócimy do tej ogólnej koncepcji wkrótce, przy okazji opisywania programowego dostępu do wywołań API Graph. Na rysunkach od 2.1 do 2.3 zilustrowano serię zapytań API Graph, które wynikają z klikania symbolu plus (+) i dodawania połączeń oraz pól. Na tych rysunkach warto zwrócić uwagę na kilka elementów:

Token dostępu

Token dostępu, który pojawia się w aplikacji, to token OAuth (<http://bit.ly/1a1kZWN>), zapewniający wygodę zalogowanemu użytkownikowi. Jest to ten sam token OAuth, którego aplikacja potrzebowałaby, aby uzyskać dostęp do danych. Z tego tokena dostępu będziemy korzystać w dalszej części tego rozdziału. Warto jednak zapoznać się z dodatkiem B, aby uzyskać zwięzłe informacje o OAuth — w tym szczegóły dotyczące sposobu implementacji przepływu OAuth dla Facebooka w celu uzyskania tokena dostępu. Zgodnie z tym, o czym wspomniano w rozdziale 1., jeśli jest to Twoje pierwsze spotkanie z OAuth, prawdopodobnie wystarczy, żebyś na razie wiedział, że protokół ten jest standardem społecznościowym, a jego nazwa pochodzi od Open Authorization. W skrócie mówiąc, OAuth jest mechanizmem pozwalającym użytkownikom na udzielanie aplikacjom firm zewnętrznych autoryzacji dostępu do danych konta bez konieczności udostępniania poufnych informacji, takich jak hasło.



Szczegółowe informacje na temat implementacji przepływu OAuth 2.0 można znaleźć w dodatku B. Implementacja tego przepływu jest potrzebna do zbudowania aplikacji, która wymaga od dowolnego użytkownika autoryzacji w celu uzyskania dostępu do danych konta.

Identyfikatory węzłów

Podstawą tego zapytania jest odpowiadający osobie o nazwisku „Matthew A. Russell” węzeł z identyfikatorem „644382747”, który reprezentuje aktualnie zalogowanego użytkownika aplikacji Graph Explorer. Wartości *id* i *name* dla węzła są nazywane *polami*. Podstawą zapytania równie dobrze może być dowolny inny węzeł, a jak wkrótce zobaczymy, „chodzenie” lub przeglądanie wykresu i wysyłanie zapytań do innych węzłów (mogą to być ludzie lub przedmioty takie jak książki czy programy telewizyjne) jest bardzo naturalne.

Ograniczenia połączenia

Jak widać na rysunku 2.2, pierwotne zapytanie można zmodyfikować za pomocą połączenia *friends*. W tym celu należy kliknąć +, a następnie przewinąć do pozycji *friends* w rozwijanym menu *connections*. Połączenia *friends*, które wyświetlają się w konsoli, reprezentują węzły połączone z pierwotnym węzłem zapytania. W tym momencie możesz kliknąć dowolne niebieskie pole *id* w tych węzłach i zainicjować zapytanie do tego konkretnego węzła. W terminologii nauki o sieciach występuje tzw. *wykres ego*, ponieważ ma on aktora (lub *ego*) jako punkt skupienia lub logiczne centrum połączone z innymi węzłami znajdującymi się wokół niego. Gdybyśmy spróbowali narysować wykres ego, przypominałby on piastę i szprychy.

Ograniczenia polubień

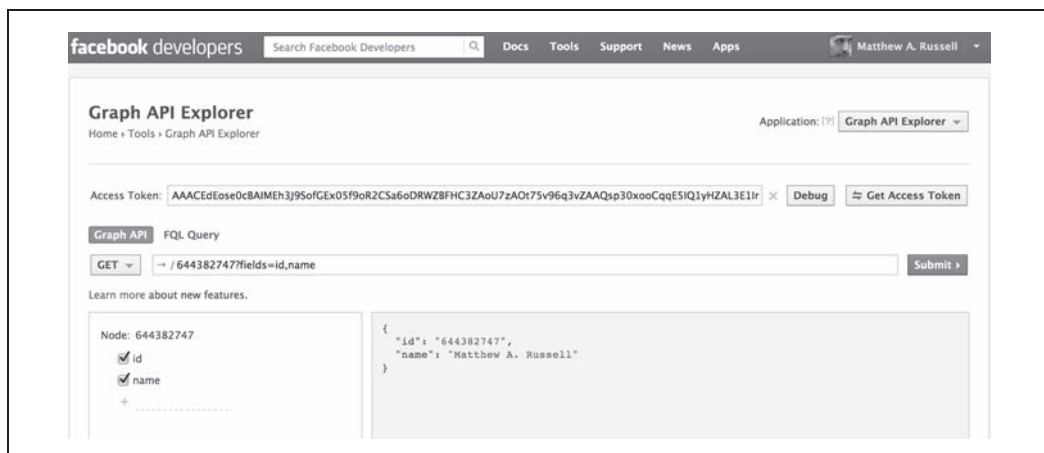
Kolejna modyfikacja oryginalnego zapytania polega na dodaniu połączeń „polubień” dla każdego z Twoich znajomych, co pokazano na rysunku 2.3. Te dane są jednak teraz zastrzeżone, a aplikacje nie mogą już uzyskać dostępu do polubień znajomych. Facebook wprowadził zmiany w interfejsie API, ograniczając zakres informacji, do których aplikacje mogą uzyskać dostęp.

Debugowanie

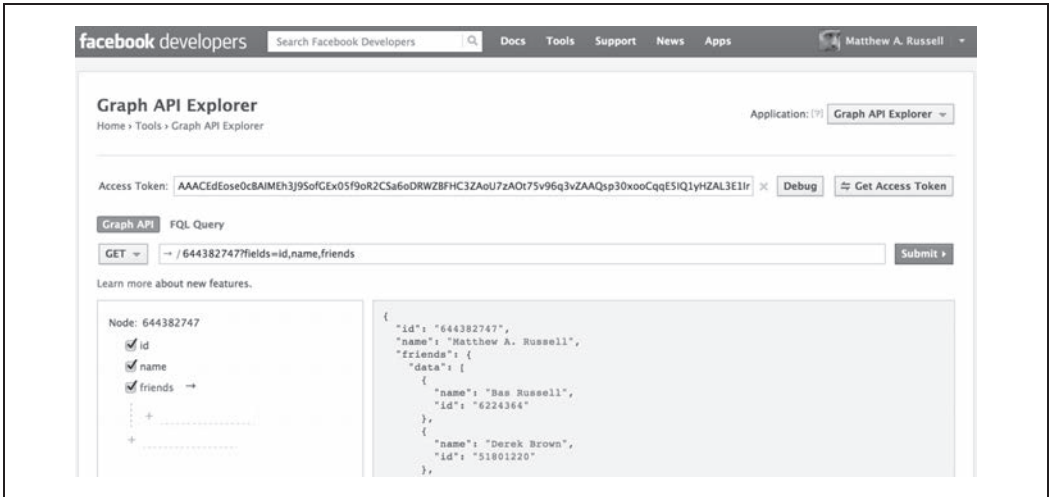
Przycisk *Debug* może się przydać do rozwiązywania problemów z zapytaniami, które na podstawie autoryzacji związanych z tokenem dostępu powinny zwracać dane, ale tego nie robią.

Format odpowiedzi JSON

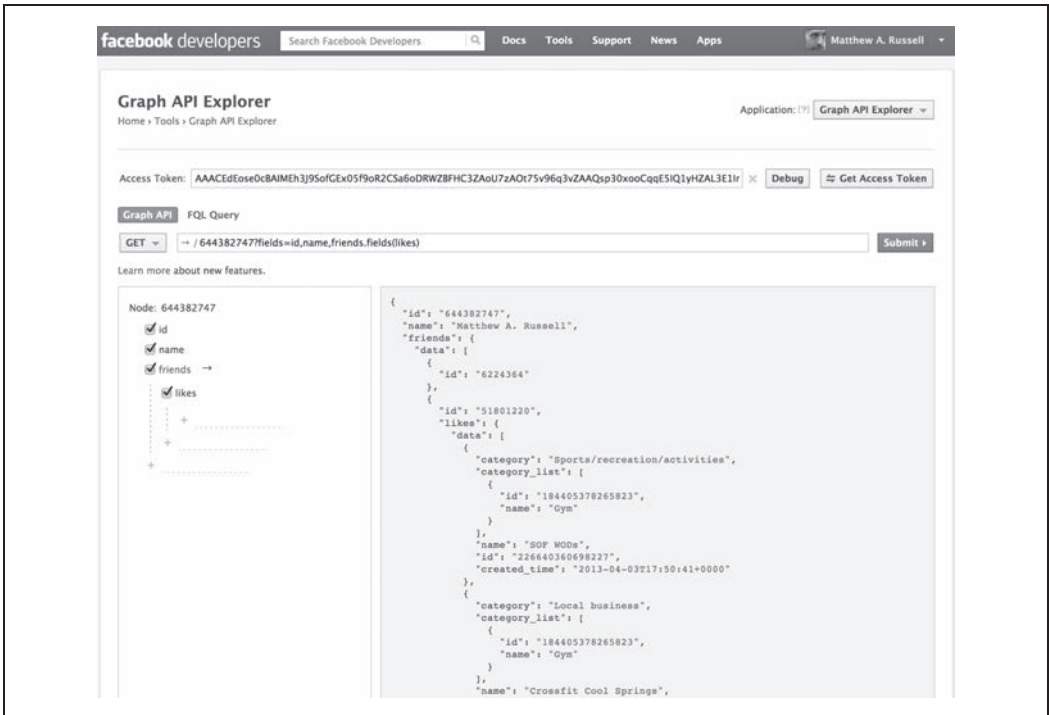
Wyniki zapytania API Graph są zwracane w wygodnym formacie JSON, który pozwala na łatwe wykonywanie operacji i przetwarzanie tych wyników.



Rysunek 2.1. Wykorzystanie programu Graph API Explorer do zapytań o węzeł grafu społecznościowego



Rysunek 2.2. Wykorzystanie aplikacji Graph API Explorer do stopniowego tworzenia zapytania o węzeł i połączenia ze znajomymi. Należy pamiętać, że niektóre dane mogą wymagać uprzywilejowanego dostępu, a aplikacje działające w piaskownicy mają dostęp tylko do bardzo ograniczonego zbioru danych. Na przestrzeni lat Facebook wprowadził wiele zmian w zasadach dostępu do danych



Rysunek 2.3. Używanie aplikacji Graph API Explorer do stopniowego tworzenia zapytania o zainteresowania znajomych: zapytanie o węzeł, połączenia ze znajomymi i polubienia tych znajomych. Ten przykład ma być opisowy. Zasady korzystania z danych Facebooka stały się bardziej rygorystyczne. Kładzie się w nich większy nacisk na prywatność użytkowników

W dalszej części tego rozdziału opiszemy programowy dostęp do API Graph za pośrednictwem pakietu Pythona. Zapytania do API Graph można jednak tworzyć w sposób bardziej bezpośredni — poprzez HTTP. W tym celu wystarczy podejrzeć żądanie, które jest widoczne w programie Graph API Explorer, i wysłać je przez HTTP. Na przykład w kodzie z listingu 2.1, w celu uproszczenia procesu tworzenia żądania HTTP do pobierania znajomych i ich upodobań, użyto pakietu `request` (<http://bit.ly/1a1lrEt>) (zamiast znacznie bardziej uciążliwego pakietu ze standardowej biblioteki Pythona, takiego jak `urllib`). Aby zainstalować ten pakiet, należy wpisać w terminalu prawdopodobne polecenie `pip install requests`. Zapytaniem sterują wartości przekazane w parametrze `fields`. Ma ono taką samą postać jak zapytanie budowane interaktywnie w programie Graph API Explorer. Szczególnie interesujące jest to, że w instrukcji `likes.limit(10){about}` wykorzystano własność API Graph zwaną ekspansją pola (<http://bit.ly/1a1lsIE>), która umożliwi tworzenie w pojedynczym wywołaniu API zagnieżdżonych zapytań do grafu.

Listing 2.1. Wykonywanie żądań do API Graph przez HTTP

```
import requests # pip install requests
import json

base_url = 'https://graph.facebook.com/me'

# Wskaż pola do pobrania
fields = 'id,name,likes.limit(10){about}'

url = '{0}?fields={1}&access_token={2}'.format(base_url, fields, ACCESS_TOKEN)

# To API bazuje na HTTP. Żądania mogą być przesyłane w przeglądarce,
# za pomocą narzędzia wiersza polecenia takiego jak curl lub za pomocą niemal
# każdego języka programowania, poprzez wysłanie żądania do adresu URL.
# Aby przekonać się samemu, kliknij hiperłącze, które wyświetli się w wynikach notatnika Jupyter Notebook,
# kiedy uruchomisz tę komórkę kodu...
print(url)

# Zinterpretuj odpowiedź jako JSON i przekonwertuj ją z powrotem
# na struktury danych Pythona
content = requests.get(url).json()

# Wyświetl estetycznie sformatowany ciąg JSON
print(json.dumps(content, indent=1))
```

Dzięki składni rozwijania pola Facebooka można ustawić limity i przesunięcia dla zapytania API. Celem pierwszego przykładu jest pokazanie, że API Facebooka bazuje na HTTP. Poniżej przedstawiono kilka przykładów ograniczeń dla pól (przesunięcia), które to przykłady ilustrują możliwości selektorów pola:

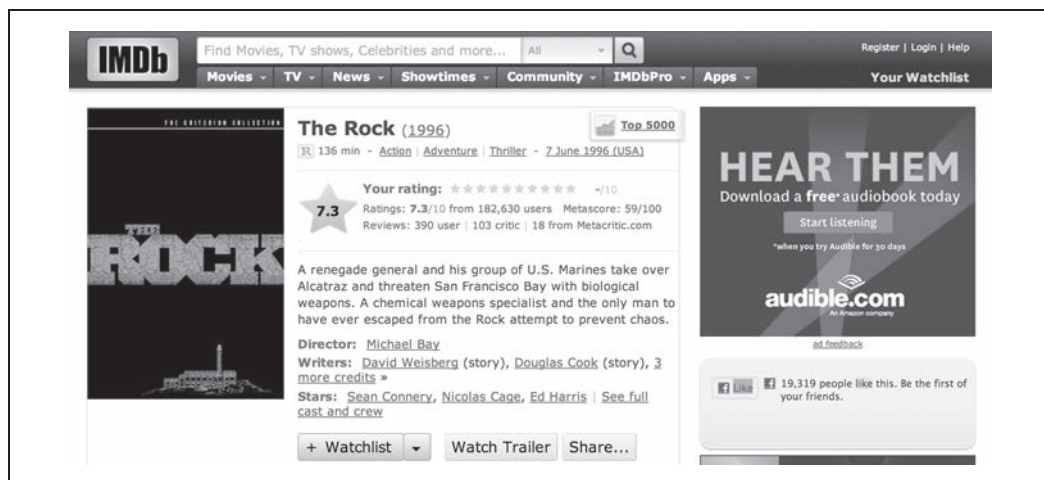
```
# Pobranie 10 moich polubień
fields = 'id,name,likes.limit(10)'
# Pobranie kolejnych 10 moich polubień
fields = 'id,name,likes.offset(10).limit(10)'
```

Interfejs API Facebooka automatycznie dzieli na strony wyniki, które zwraca, co oznacza, że jeśli zapytanie zwraca mnóstwo wyników, to interfejs API nie przekaże ich wszystkich naraz. Zamiast tego podzieli je na porcje (*strony*) i zwróci kursor wskazujący na następną stronę wyników. Więcej informacji na temat dzielenia wyników na strony można znaleźć w dokumentacji mechanizmu podziału na strony (<http://bit.ly/1a1ltMP>).

2.2.2. Protokół Open Graph

Oprócz poznania interfejsu API Graph, który umożliwia poruszanie się po grafie społecznościowym i odpytywanie o znane obiekty Facebooka, warto również wiedzieć, że w kwietniu 2010 r., na tej samej konferencji F8, na której wprowadzono graf społecznościowy, Facebook udostępnił mechanizm znany jako protokół Open Graph (<http://bit.ly/1a1lu3m>) (OGP). Krótko mówiąc, OGP jest mechanizmem, który umożliwia programistom tworzenie dowolnych stron internetowych na grafie społecznościowym Facebooka poprzez wstrzykiwanie do strony metadanych RDFa (<http://bit.ly/1a1lvEr>). Oprócz możliwości dostępu z „ogrodzonego obszaru” wewnątrz Facebooka do dziesiątek obiektów opisanych w podręczniku API Graph (<http://bit.ly/1a1lvEr>) (użytkownicy, zdjęcia, filmy, zalogowania, linki, komunikaty statusu itp.) możesz także spotkać w grafie społecznościowym strony z internetu reprezentujące istotne pojęcia. Innymi słowy, OGP jest sposobem „otwarcia” grafu społecznościowego. Opis tych pojęć znajdziesz w dokumentacji programistów Facebooka pod hasłem „graf otwarty” (ang. *open graph*)².

Istnieją praktycznie nieograniczone możliwości wykorzystywania OGP do „wszczepiania” stron internetowych do grafu społecznościowego. Jest bardzo prawdopodobne, że spotkałeś już wiele z nich, nawet nie zdając sobie z tego sprawy. Dla przykładu rozważmy rysunek 2.4, który ilustruje stronę filmu *The Rock* z serwisu IMDb.com (<http://imdb.com/>). Na pasku bocznym po prawej stronie widzisz dość znajomy przycisk *Lubię to* (*Like*) z komunikatem „19,319 people like this. Be the first of your friends” (dosł. „19 319 osób to lubi. Bądź pierwszym spośród Twoich znajomych”). Serwis IMDb włączył tę funkcjonalność poprzez implementację OGP dla każdego adresu URL odpowiadającego obiektom istotnym do włączenia do grafu społecznościowego. Dzięki odpowiednim metadansom RDFa na stronie Facebook może jednoznacznie łączyć się z tymi obiektami i uwzględnić je w strumieniach aktywności oraz innych kluczowych elementach interfejsu użytkownika.



Rysunek 2.4. Strona IMDb zawierająca implementację OGP dla filmu *The Rock*

² W tym punkcie, opisującym implementację OGP, termin *graf społecznościowy*, jeśli nie zaznaczono tego wyraźnie inaczej, jest używany zarówno w odniesieniu do grafu społecznościowego, jak i grafu otwartego.

Implementacja OGP za pośrednictwem przycisków typu *Lubię to* na stronach internetowych może wydawać się oczywista, jeśli przyzwyczaiłeś się do oglądania ich przez ostatnich kilka lat. Jednak fakt, że Facebook odniósł sukces, otwierając swoją platformę programistyczną w sposób umożliwiający włączanie dowolnych obiektów z internetu, jest dość ważny i ma potencjalnie istotne konsekwencje.

W 2013 r. Facebook wdrożył semantyczną wyszukiwarkę o nazwie Facebook Graph Search. Był to mechanizm pozwalający użytkownikom wykonywać zapytania w języku naturalnym, bezpośrednio z paska wyszukiwania. Na przykład w pasku wyszukiwania można było napisać „Moi znajomi, którzy mieszkają w Londynie i lubią koty”, a Facebook znajdował część wspólną pomiędzy znajomymi mieszkającymi w Londynie i tymi, którzy jednocześnie są miłośnikami kotów. Ten nowy sposób odpytywania Facebooka jednak nie przetrwał długo. Pod koniec 2014 r. Facebook zastąpił tę funkcję wyszukiwania semantycznego podejściem bazującym na słowach kluczowych. Niemniej jednak graf łączący obiekty na Facebooku z innymi artefaktami wykorzystywany jest nadal. W listopadzie 2017 r. Facebook uruchomił samodzielną aplikację mobilną o nazwie Facebook Local, koncentrującą się na łączeniu fizycznych miejsc i wydarzeń oraz tworzeniu społeczności. Aplikacja ta może Ci na przykład powiedzieć, które restauracje mają największą popularność wśród Twoich znajomych na Facebooku. Technologia, na której bazuje ta aplikacja, jest niemal na pewno grafem społecznościowym Facebooka.

Zanim przejdziemy do zapytań API Graph, przyjrzyjmy się pokrótce istocie implementacji OGP. Kanoniczny przykład z dokumentacji OGP, który pokazuje, jak zamienić stronę IMDb filmu *The Rock* w obiekt protokołu Open Graph w ramach dokumentu XHTML korzystającego z przestrzeni nazw, jest mniej więcej taki:

```
<html xmlns:og="http://ogp.me/ns#">
<head>
<title>The Rock (1996)</title>
<meta property="og:title" content="The Rock" />
<meta property="og:type" content="movie" />
<meta property="og:url" content="http://www.imdb.com/title/tt0117500/" />
<meta property="og:image" content="http://ia.media-imdb.com/images/rock.jpg" />
...
</head>
...
</html>
```

Takie fragmenty metadanych mają ogromny potencjał, jeśli zostaną użyte na wielką skalę, ponieważ umożliwiają korzystanie z adresów URI takich jak <http://www.imdb.com/title/tt0117500> w celu jednoznacznego reprezentowania dowolnej strony internetowej — osoby, firmy, produktu itd. — w sposób możliwy do odczytania maszynowo, ale także wspierający wizję sieci semantycznej. Oprócz możliwości „polubienia” filmu *The Rock* użytkownicy za pośrednictwem niestandardowych akcji mogą potencjalnie wchodzić z nim w inne interakcje. Na przykład mogą wskazać, że oglądali film *The Rock* (<http://bit.ly/2Qx0Kfo>). Protokół OGP udostępnia obszerny i elastyczny zbiór działań pomiędzy użytkownikami a obiektami w ramach grafu społecznościowego.



Jeśli jeszcze tego nie zrobiłeś, zobacz kod źródłowy HTML <http://www.imdb.com/title/tt0117500> i przekonaj się, jak wygląda zastosowanie formatu RDFa.

W gruncie rzeczy odpytywanie API Graph o obiekty Open Graph jest niezwykle proste: aby pobrać szczegóły dotyczące obiektu, dołącz adres URL strony internetowej lub identyfikator obiektu do adresu `http(s)://graph.facebook.com/`. Na przykład próba pobrania adresu URL `http://graph.facebook.com/http://www.imdb.com/title/tt0117500` w przeglądarce internetowej zwróci następującą odpowiedź:

```
{
  "share": {
    "comment_count": 0,
    "share_count": 1779
  },
  "og_object": {
    "id": "10150461355237868",
    "description": "Directed by Michael Bay. With Sean Connery, ...",
    "title": "The Rock (1996)",
    "type": "video.movie",
    "updated_time": "2018-09-18T08:39:39+0000"
  },
  "metadata": {
    "fields": [{
      "name": "id",
      "description": "The URL being queried",
      "type": "string"
    }, {
      "name": "app_links",
      "description": "AppLinks data associated with the URL",
      "type": "aplinks"
    }, {
      "name": "development_instant_article",
      "description": "Instant Article object for the URL, in developmen...",
      "type": "instantarticle "
    }, {
      "name": "instant_article ",
      "description": " Instant Article object for the URL ",
      "type": "instantarticle "
    }, {
      "name": "og_object ",
      "description": " Open Graph Object for the URL ",
      "type": "opengraphobject: generic "
    }, {
      "name": "ownership_permissions ",
      "description": " Permissions based on ownership of the URL ",
      "type": "urlownershippermissions "
    }
  ],
  "type": "url "
},
  "id": "http://www.imdb.com/title/tt0117500"
}
```

Jeśli sprawdzisz źródło adresu URL `http://www.imdb.com/title/tt0117500`, przekonasz się, że pola w odpowiedzi odpowiadają danym w tagach meta strony. Nie jest to przypadek. Dostarczanie obszernych metadanych w odpowiedzi na proste zapytanie to podstawa sposobu zaprojektowanego działania OGP. Za pomocą narzędzia Graph API Explorer można uzyskać dostęp do jeszcze większej ilości metadanych dotyczących obiektów należących do grafu. Spróbuj wkleić do programu Graph API Explorer `380728101301?metadata=1`. Skorzystaliśmy tu z identyfikatora filmu *The Rock* i poprosiliśmy o dodatkowe metadane. Oto przykładowa odpowiedź:

```

{
  "created_time": "2007-11-18T20:32:10+0000",
  "title": "The Rock (1996)",
  "type": "video.movie",
  "metadata": {
    "fields": [
      {
        "name": "id",
        "description": "The Open Graph object ID",
        "type": "numeric string"
      },
      {
        "name": "admins",
        "description": "A list of admins",
        "type": "list<opengraphobjectprofile>"
      },
      {
        "name": "application",
        "description": "The application that created this object",
        "type": "opengraphobjectprofile"
      },
      {
        "name": "audio",
        "description": "A list of audio URLs",
        "type": "list<opengraphobjectaudio>"
      },
      {
        "name": "context",
        "description": "Context",
        "type": "opengraphcontext"
      },
      {
        "name": "created_time",
        "description": "The time the object was created",
        "type": "datetime"
      },
      {
        "name": "data",
        "description": "Custom properties of the object",
        "type": "opengraphstruct:video.movie"
      },
      {
        "name": "description",
        "description": "A short description of the object",
        "type": "string"
      },
      {
        "name": "determiner",
        "description": "The word that appears before the object's title",
        "type": "string"
      },
      {
        "name": "engagement",
        "description": "The social sentence and like count for this object and its associated share. This is the same info used for the like button",
        "type": "engagement"
      },
      {
        "name": "image",
        "description": "A list of image URLs",
        "type": "list<opengraphobjectimagevideo>"
      },
      {
        "name": "is_scraped",
        "description": "Whether the object has been scraped",
        "type": "bool"
      },
      {
        "name": "locale",
        "description": "The locale the object is in",
        "type": "opengraphobjectlocale"
      }
    ]
  }
}

```

```

    }, {
      "name": "location",
      "description": "The location inherited from Place",
      "type": "location"
    }, {
      "name": "post_action_id",
      "description": "The action ID that created this object",
      "type": "id"
    }, {
      "name": "profile_id",
      "description": "The Facebook ID of a user that can be followed",
      "type": "opengraphobjectprofile"
    }, {
      "name": "restrictions",
      "description": "Any restrictions that are placed on this object",
      "type": "opengraphobjectrestrictions"
    }, {
      "name": "see_also",
      "description": "An array of URLs of related resources",
      "type": "list<string>"
    }, {
      "name": "site_name",
      "description": "The name of the web site upon which the object resides",
      "type": "string"
    }, {
      "name": "title",
      "description": "The title of the object as it should appear in the graph",
      "type": "string"
    }, {
      "name": "type",
      "description": "The type of the object",
      "type": "string"
    }, {
      "name": "updated_time",
      "description": "The last time the object was updated",
      "type": "datetime"
    }, {
      "name": "video",
      "description": "A list of video URLs",
      "type": "list<opengraphobjectimagevideo>"
    }
  ],
  "type": "opengraphobject:video.movie",
  "connections": {
    "comments": "https://graph.facebook.com/v3.1/380728101301/comments?access_token=EAAyvPRk4YUEBAHvVKqnhZBxMDAwBKEpWrsM6J8ZCxHkLu...&pretty=0",
    "likes": "https://graph.facebook.com/v3.1/380728101301...&pretty=0",
    "picture": "https://graph.facebook.com/v3.1/380728101...&pretty=0",
    "reactions": "https://graph.facebook.com/v3.1/380728101...reactions?access_token=EAAyvPRk4YUEBAHvVKqnhZBxMDAwBKEpWrsM6J8ZC...&pretty=0"
  }
},
"id": "380728101301"
}

```

Elementy w `metadata.connections` są wskaźnikami do innych węzłów grafu, które można przeszukiwać. W ten sposób można dostać się do innych intrygujących fragmentów danych, chociaż to, co jest się w stanie znaleźć, może być mocno ograniczone przez ustawienia prywatności Facebooka.



Spróbuj użyć identyfikatora Facebooka książki „MiningTheSocialWeb”, aby uzyskać szczegółowe informacje na temat jej oficjalnego fanpage’a (<http://on.fb.me/1a1lAI8>) za pomocą narzędzia Graph API Explorer. Możesz także zmodyfikować listing 2.1 w celu programowego zapytania o stronę <https://graph.facebook.com/MiningTheSocialWeb>, aby pobrać podstawowe informacje na jej temat, w tym opublikowane na niej treści. Na przykład dołączenie do tego adresu URL ciągu zapytania z kwalifikatorem, takim jak `?fields=posts` spowoduje zwrócenie listy opublikowanych treści.

I ostatnia uwaga przed przejściem do omawiania programowego dostępu do API Graph. Podczas analizy możliwości OGP trzeba myśleć przyszłościowo i być kreatywnym. Trzeba jednak pamiętać, że technologie te stale ewoluują. Ponieważ odnosi się to do sieci semantycznej i ogólnie do standardów w internecie, użycie słowa „open” (otwarty) (<http://tcrn.ch/1a1lAYF>), co zrozumiałe, wywołało pewną konsternację. Po drodze wprowadzono różne wyjątki w specyfikacji (<http://bit.ly/1a1lAbd>), a niektóre prawdopodobnie są opracowywane nadal. Można również stwierdzić, że OGP to w istocie wysiłek jednego dostawcy i ma niewiele większe możliwości od elementów meta (<http://bit.ly/1a1lBMA>) dostępnych znacznie wcześniej, chociaż te dwie technologie wydają się różnie odbierane.

Jednak to, czy OGP lub jakiś następcza API Graph Search będzie kiedyś dominował w sieci, jest kwestią dyskusyjną. Z pewnością technologia ta ma duży potencjał; wskaźniki sukcesu pokazują, że rozwój idzie w dobrym kierunku, a w przyszłości, w miarę wprowadzania innowacji, może się wydarzyć wiele ekscytujących rzeczy. Teraz, gdy poznaliśmy szerszy kontekst grafu społecznościowego, wróćmy do opisu korzystania z API Graph.

2.3. Analiza połączeń grafu społecznościowego

Oficjalny pakiet Pythona — SDK dla API Graph (<http://bit.ly/2kpej52>) — jest utrzymywany przez społeczność użytkowników rozwidleniem, czyli kopią (ang. *fork*) repozytorium wcześniej obsługiwanego przez Facebook. Można go zainstalować zgodnie ze standardowym protokołem za pomocą menedżera pip, korzystając z polecenia `pip install facebook-sdk`. Pakiet ten zawiera kilka przydatnych metod pomocniczych, które umożliwiają różne sposoby interakcji z Facebookiem. Jednak aby użyć API Graph do pobierania danych, trzeba znać tylko kilka kluczowych metod klasy GraphAPI (zdefiniowanych w pliku źródłowym *facebook.py*). Równie dobrze — jeśli takie są Twoje preferencje — możesz kierować zapytania bezpośrednio przez HTTP (tak jak pokazano w listingu 2.1). Oto te metody:

```
get_object(self, id, **args)
```

Przykładowe użycie: `get_object("me", metadata=1)`

```
get_objects(self, id, **args)
```

Przykładowe użycie: `get_objects(["me", "some_other_id"], metadata=1)`

```
get_connections(self, id, connection_name, **args)
```

Przykładowe użycie: `get_connections("me", "friends", metadata=1)`

```
request(self, path, args=None, post_args=None)
```

Przykładowe użycie: `request("search", {"q": "social web", "type": "page"})`



Chociaż Facebook stosuje dla swojego API ograniczenia liczby żądań (<http://bit.ly/2iXSeKs>), limity są ustalane indywidualnie dla każdego użytkownika. Im więcej użytkowników ma Twoja aplikacja, tym wyższy jest jej limit żądań w czasie. Niemniej jednak należy starannie zaprojektować aplikację, tak aby korzystać z API w jak najmniejszym stopniu. Zalecaną najlepszą praktyką powinno być także obsługiwanie wszystkich występujących błędów.

Najpopularniejszym (a często jedynym) argumentem reprezentowanym przez słowo kluczowe jest `metadata = 1`. Używa się go do pobrania — oprócz szczegółowych danych na temat obiektu — danych o połączeniach związanych z obiektem. Przyjrzyjmy się listingowi 2.2. Wprowadzono na nim klasę `GraphAPI` i użyto jej metod do wyszukiwania informacji o Tobie, Twoich połączeniach lub wyszukiwanym hasle, takim jak *serwis społecznościowy*. W tym przykładzie wprowadzono również funkcję pomocniczą o nazwie `pp`, która jest używana w pozostałej części tego rozdziału do estetycznego drukowania wyników w formacie JSON.

Listing 2.2. Odpytywanie API Graph za pomocą Pythona

```
import facebook # pip install facebook-sdk
import json
# Funkcja pomocnicza do estetycznego drukowania obiektów Pythona w formacie JSON
def pp(o):
    print(json.dumps(o, indent=1))
# Utworzenie połączenia z interfejsem API Graph z wykorzystaniem tokena dostępu
g = facebook.GraphAPI(ACCESS_TOKEN, version='2.8')
# Uruchomienie kilku przykładowych zapytań:
# Pobierz mój identyfikator
pp(g.get_object('me'))
# Uzyskaj połączenia do identyfikatora
# Przykładowe nazwy połączeń: 'feed', 'likes', 'groups', 'posts'
pp(g.get_connections(id='me', connection_name='likes'))
# Wyszukaj lokalizację. Może wymagać zatwierdzonej aplikacji
pp(g.request("search", {'type': 'place', 'center': '40.749444, -73.968056',
                        'fields': 'name, location'}))
```



API Facebooka uległo pewnym zmianom, a uzyskanie uprawnień do programowego pobierania publicznych treści ze stron Facebooka wymaga przesłania aplikacji w celu sprawdzenia i zatwierdzenia (<http://bit.ly/2vDb2B1>).

Zapytanie dotyczące wyszukiwania lokalizacji jest interesujące, ponieważ zwraca elementy z grafu, które odpowiadają miejscom geograficznie bliskim podanej szerokości i długości geograficznej. Oto kilka przykładowych wyników z tego zapytania:

```
{
  "data": [{
    "name": "United Nations",
    "location": {
      "city": "New York",
      "country": "United States",
      "latitude": 40.748801288774,
      "longitude": -73.968307971954,
      "state": "NY"
    }
  }],
}
```

```

    "id": "10155092544275820"
  }, {
    "name": "United Nations Security Council",
    "location": {
      "city": "New York",
      "country": "United States",
      "latitude": 40.749474401558,
      "longitude": -73.967919463251,
      "state": "NY",
      "street": "760 United Nations Plaza",
      "zip": "10017"
    },
    "id": "113874685435095"
  }, {
    "name": "United Nations Secretariat Building",
    "location": {
      "city": "New York",
      "country": "United States",
      "latitude": 40.749,
      "longitude": -73.968,
      "state": "NY",
      "street": "United Nations Plz",
      "zip": "10017"
    },
    "id": "846565615355416"
  }, {
    "name": "Manhattan, New York",
    "location": {
      "city": "Manhattan",
      "country": "United States",
      "latitude": 40.7779,
      "longitude": -73.9675,
      "state": "NY",
      "zip": "03501"
    },
    "id": "214957115200718"
  },
  ...
],
"paging": {
  "cursors": {
    "after": "MjQZD"
  },
  "next": "https://graph.facebook.com/v3.2/search?access_token=..."
}
}

```



Gdybyśmy skorzystali z narzędzia Graph API Explorer, wyniki byłyby identyczne. Podczas pracy nad rozwojem aplikacji często przydaje się zamienne używanie narzędzia Graph API Explorer i notatnika Jupyter Notebook, w zależności od konkretnego celu. Zaletą narzędzia Graph API Explorer jest możliwość łatwego klikania wartości ID i tworzenia nowych zapytań podczas eksperymentowania.

W tym momencie masz do dyspozycji zarówno narzędzie Graph API Explorer, jak i konsolę Pythona (i wszystko, co mają one do zaoferowania). Teraz, kiedy przeskalowaliśmy otoczony murem ogród Facebooka, spróbujmy przeanalizować niektóre z jego danych.

2.3.1. Analizowanie stron Facebooka

Facebook zaczął jako portal czysto społecznościowy, bez grafu społecznościowego lub dobrego sposobu na obecność firm i innych podmiotów, lecz szybko dostosował się do potrzeb rynku. Obecnie swoje strony na Facebooku wraz z bazą swoich fanów mają firmy, kluby, książki i wiele innych rodzajów podmiotów nieożywionych (<http://on.fb.me/1a1lCzQ>). Strony na Facebooku są potężnym narzędziem wykorzystywanym przez firmy do angażowania swoich klientów. Facebook dołożył wszelkich starań, aby umożliwić administratorom stron zrozumienie swoich fanów dzięki niewielkiemu przyborkowi, który został trafnie nazwany *Insights* (<http://2ox6w7j/>).

Jeśli już jesteś użytkownikiem Facebooka, istnieje duże prawdopodobieństwo, że polubiłeś jedną lub więcej stron na Facebooku, które odzwierciedlają coś, co akceptujesz, lub coś, co jest według Ciebie interesujące. Pod tym względem strony Facebooka znacznie rozszerzyły możliwości grafu społecznościowego jako platformy. Wyraźne miejsce na Facebooku dla stron podmiotów nieożywionych, przycisk *Lubię to* oraz charakter grafu społecznościowego łącznie stanowią potężny arsenał interesującej platformy graficznej, która niesie ze sobą ogrom możliwości (opis grafów zainteresowań i ich możliwości zamieszczono w podrozdziale „Dlaczego Twitter to jest »to«?” w rozdziale 1.).

Analiza strony Facebooka tej książki

Biorąc pod uwagę, że strona tej książki na Facebooku okazała się jednym z najpopularniejszych wyników wyszukiwania hasła „social web”, użycie jej jako punktu wyjścia do zobrazowania niektórych pouczających analiz zamieszczonych w niniejszym rozdziale wydaje się dość naturalne³.

Oto kilka pytań, które warto rozważyć w odniesieniu do facebookowej strony tej książki lub prawie każdej innej strony na Facebooku:

- Jak popularna jest strona?
- Jak bardzo zaangażowani są fani strony?
- Czy któryś z fanów tej strony jest szczególnie szczerzy i aktywny?
- Jakie są najczęściej poruszane tematy na stronie?

Jedynym ograniczeniem w kwestii tego, o co możesz poprosić API Graph podczas wyszukiwania treści na Facebooku, jest Twoja wyobraźnia. Powyższe pytania powinny Cię skłonić do wybrania właściwego kierunku. Będziemy je również wykorzystywać jako podstawę do pewnych porównań stron.

Przypomnijmy, że punktem wyjścia do naszej analizy było poszukiwanie hasła „social web”, w wyniku którego uzyskaliśmy dane o książce pod tytułem *Mining the Social Web*. Oto uzyskany wynik wyszukiwania:

```
{
  "data": [{
    "name": "Mining the Social Web",
```

³ Podczas lektury tego punktu pamiętaj, że Facebook ogranicza publiczny dostęp do treści dla aplikacji, które nie zostały przesłane i zatwierdzone. Kod w tym punkcie i wynik jego działania zamieszczono w celach poglądowych. Warto zapoznać się z dokumentacją programisty dostępną tutaj: <https://developers.facebook.com/docs/apps/review>.


```

    "id": "146803958708175"
  }, {
    "name": "R: Mining spatial, text, web, and social media",
    "id": "321086594970335"
  }
],
"paging": {
  "cursors": {
    "before": "MAZDZD",
    "after": "MQZDZD"
  }
}
}
}

```

Dla każdego elementu w wynikach wyszukiwania moglibyśmy użyć ID jako podstawy zapytania wykonywanego za pomocą metody `get_object` egzemplarza obiektu `facebook.GraphAPI`. Jeśli nie masz do dyspozycji liczbowego identyfikatora, możesz wykonać żądanie wyszukiwania po nazwie i sprawdzić uzyskany wynik. Zgodnie z tym, co pokazano w listingu 2.3, za pomocą metody `get_object` możemy uzyskać więcej informacji — na przykład liczbę fanów na Facebooku.

*Listing 2.3. Odpytywanie interfejsu API Graph w celu wyszukania informacji o książce *Mining the Social Web* i liczbie jej fanów*

```

# Wyszukaj identyfikator strony według nazwy
pp(g.request("search", {'q': 'Mining the Social Web', 'type': 'page'}))

# Pobierz identyfikator książki i sprawdź liczbę fanów
mtsw_id = '146803958708175'
pp(g.get_object(id=mtsw_id, fields=['fan_count']))

```

Oto wynik działania tego kodu:

```

{
  "data": [{
    "name": "Mining the Social Web",
    "id": "146803958708175"
  }, {
    "name": "R: Mining spatial, text, web, and social media",
    "id": "321086594970335"
  }
],
  "paging": {
    "cursors": {
      "before": "MAZDZD",
      "after": "MQZDZD"
    }
  }
}
}

```

Zliczanie liczby fanów, których ma strona, i porównywanie jej z innymi stronami z podobnej kategorii może być sposobem pomiaru siły „marki” na Facebooku. *Mining the Social Web* jest dość niszową książką techniczną, więc warto porównać ją z innymi książkami wydawanymi przez O’Reilly Media, które mają strony na Facebooku.

Dla każdego rodzaju analizy popularności niezbędne do zrozumienia szerszego kontekstu są porównywane elementy. Istnieje wiele sposobów wykonywania porównań. Od razu rzuca się w oczy kilka uderzających danych. W czasie pisania tej książki jej wydawca, wydawnictwo O’Reilly Media

(<http://on.fb.me/1a1lD6F>), miał około 126 000 polubień, natomiast język programowania Python (<http://on.fb.me/1a1lD6V>) — 121 000 polubień. Tak więc popularność książki *Mining the Social Web* wynosiła około 2% całej grupy fanów wydawcy oraz grupy fanów języka programowania. Jest oczywiste, że popularność tej książki będzie wzrastać, mimo że jej temat jest niszowy.

Z pewnością lepiej byłoby porównać inną niszową książkę, podobną do *Mining the Social Web*, lecz podczas przeglądania danych na Facebooku nie jest łatwo znaleźć dobre porównanie w rodzaju „jabłko z jabłkiem”. Na przykład aby znaleźć dobrą stronę do porównywania, nie można wyszukiwać dowolnych stron i filtrować wyniki, tak by zostały same strony o książkach. Zamiast tego trzeba szukać stron o książkach, a następnie filtrować zestaw wyników według kategorii, tak aby pobrać wyłącznie strony w kategorii „książki”. Do rozważenia jest jednak kilka opcji.

Jedną z nich jest wyszukiwanie podobnego tytułu wydawnictwa O'Reilly. Na przykład w momencie pisania drugiego wydania tej książki wyniki wyszukiwania API Graph dla zapytania *Programming Collective Intelligence* (podobna niszowa książka O'Reilly) zawierały stronę społeczności z około 925 polubieniami.

Inna możliwość polega na wykorzystaniu do porównania pojęć z protokołu Open Graph Facebooka. Na przykład katalog online O'Reilly wykorzystywany do implementacji OGP dla wszystkich jego tytułów zawiera przyciski *Lubię to* zarówno dla książki *Mining the Social Web*, wydanie drugie (<http://oreil.ly/1cMLoug>), jak i dla książki *Programming Collective Intelligence* (<http://oreil.ly/1a1lGzw>). Możemy bez trudu wysłać zapytania do API Graph, aby zobaczyć, jakie są dostępne dane, po prostu żądając w przeglądarce następujących adresów URL:

Zapytanie API Graph dla książki *Mining the Social Web*

```
https://graph.facebook.com/http://shop.oreilly.com/product/0636920030195.do
```

Zapytanie API Graph dla książki *Programming Collective Intelligence*

```
https://graph.facebook.com/http://shop.oreilly.com/product/9780596529321.do
```

W przypadku zapytań programowych w Pythonie adresy URL są obiektami, które odpytujemy (podobnie jak adres URL wpisu *IMDb* dla *The Rock* był tym, o co pytaliśmy wcześniej), zatem w kodzie możemy zapytać o te obiekty w sposób pokazany w listingu 2.4.

Listing 2.4. Odpytywanie interfejsu API Graph o obiekty Open Graph na podstawie ich adresów URL

```
# Link do katalogu MTSW
pp(g.get_object('http://shop.oreilly.com/product/0636920030195.do'))
# Link do katalogu PCI
pp(g.get_object('http://shop.oreilly.com/product/9780596529321.do'))
```

Należy pamiętać o subtelnej, ale bardzo ważnej różnicy. Chociaż zarówno strona katalogu O'Reilly, jak i fanpage książki *Mining the Social Web* na Facebooku logicznie reprezentują tę samą książkę, węzły (i towarzyszące im metadane, takie jak liczba polubień) odpowiadające stronie na Facebooku i stronie katalogu O'Reilly są całkowicie niezależne. Łączy je tylko to, że każda przedstawia ten sam podmiot z rzeczywistego świata.



Całkowicie odrębnym rodzajem analizy, znanej jako *rozdzielczość encji* (ang. *entity resolution*) (lub *ujednoznacznienie encji* — ang. *entity disambiguation* — w zależności od sposobu ujęcia problemu), jest proces agregacji wzmianek o podmiotach w pojedynczy byt platoński. Na przykład w tym przypadku w procesie analizy encji moglibyśmy zaobserwować, że w grafie Open Graph istnieje wiele węzłów, które faktycznie odwołują się do tego samego platońskiego bytu *Mining the Social Web*, i stworzyć pomiędzy nimi połączenia wskazujące, że w rzeczywistości dotyczą one tego samego podmiotu w świecie rzeczywistym. Rozdzielczość encji to ekscytująca dziedzina badań, która w przyszłości nadal będzie wywierała głęboki wpływ na sposób, w jaki wykorzystujemy dane.

Eksplorację API Graph z poziomu notatnika Jupyter Notebook zaprezentowano na rysunku 2.5.

```
jupyter Chapter 2 - Mining Facebook (autosaved)
File Edit View Insert Cell Kernel Widgets Help
+ -> [Navigation icons] [Markdown] [Cell toolbar] [id]

# Create a connection to the Graph API with your access token
g = facebook.GraphAPI(ACCESS_TOKEN, version='2.7')

# Execute a few example queries:

# Get my ID
pp(g.get_object('me'))

{
  "name": "Mikhail Klassen",
  "id": "10103107761133572"
}

In [13]: # Get the connections to an ID
# Example connection names: 'feed', 'likes', 'groups', 'posts'
pp(g.get_connections(id='me', connection_name='likes'))

{
  "name": "Work Nicer Coworking",
  "id": "1667787896826240",
  "created_time": "2018-03-19T18:46:53+0000"
},
{
  "name": "Amazon Web Services",
  "id": "153063591397681",
  "created_time": "2018-03-18T17:45:00+0000"
},
{
  "name": "Fission Ventures",
  "id": "859758314179015",
  "created_time": "2018-03-13T11:32:41+0000"
},
{
  "name": "The Points Guy",
  "id": "123185754376465",

```

Rysunek 2.5. Eksploracja API Graph jest bardzo prosta dzięki interaktywnemu środowisku programistycznemu, takiemu jak Jupyter Notebook

Chociaż podczas eksploracji danych często nie jest możliwe dokonywanie zapewniających miarodajny wynik porównań typu „jabłko do jabłka”, wciąż jest wiele do nauczenia się. Eksploracja zbioru danych wystarczająco obszernego, aby można było zgromadzić silne trendy w danych, często dostarcza wszystkich niezbędnych informacji, które są potrzebne, gdy napotka się problem po raz pierwszy.

Angażowanie fanów i mierzenie siły marki mediów społecznościowych

Podczas gdy niektóre gwiazdy są bardzo aktywne w mediach społecznościowych, inne pozostawiają zadanie zaangażowania publiczności online specjalnemu zespołowi ds. marketingu w sieciach społecznościowych lub specjalistom w dziedzinie *public relations*. Strony na Facebooku to świetny sposób

na kontakt z fanami. To prawda niezależnie od tego, czy jesteś popularnym celebrytą, gwiazdką YouTube'a, czy po prostu uruchamiasz stronę na Facebooku dla organizacji non profit, w której jesteś wolontariuszem.

W tym punkcie będziemy eksplorować dane stron Facebooka trzech niezwykle popularnych muzyków, aby dowiedzieć się, jak bardzo fani na Facebooku reagują na informacje publikowane na tychże stronach. Na tej podstawie można zrozumieć, dlaczego dla tych artystów (i ich menedżerów ds. reklamy) te informacje są tak ważne. Dbanie o to, by fani byli zaangażowani i by była z nimi łączność, ma kluczowe znaczenie dla tego, żeby wprowadzenie produktu na rynek, sprzedaż biletów na imprezę czy zmobilizowanie fanów zakończyło się sukcesem.

Skuteczna komunikacja z publicznością ma zatem duże znaczenie. Zła komunikacja może odstraszyć nawet najbardziej lojalnych fanów. Z tego powodu dla artysty lub publicysty zdobycie twardych danych dotyczących zaangażowania swojej publiczności na Facebooku może mieć duże znaczenie. Właśnie o to chodzi w tym punkcie.

Artyści, których wybraliśmy do porównania, to Taylor Swift, Drake i Beyoncé. Każdy z nich jest niezwykle doświadczony i utalentowany. Wszyscy mają strony na Facebooku z dużą liczbą fanów.



Aby znaleźć identyfikatory stron każdego z artystów, możesz użyć narzędzi wyszukiwania, które omówiliśmy wcześniej. Oczywiście choć w przedstawionych przykładach wykorzystano dane ze stron na Facebooku, to prezentowane koncepcje mają bardziej ogólny charakter. Podobne statystyki możesz spróbować pobierać z innych platform — na przykład z Twittera. Możesz także napisać kod przetwarzający portale informacyjne w poszukiwaniu wzmianek o konkretnych celebrytach lub markach, które chcesz śledzić. Jeśli kiedykolwiek korzystałeś z Google Alerts, powinieneś mieć pogląd na temat tego, jak to może wyglądać.

W listingu 2.5 zidentyfikujemy każdego artystę według identyfikatora strony, który można pobrać za pomocą operacji wyszukiwania, tak jak w listingu 2.3. Następnie zdefiniujemy funkcję pomocniczą do pobierania liczby fanów stron jako liczby całkowitej. Zapiszemy te liczby w trzech różnych zmiennych, a następnie wyświetlimy wyniki.

Listing 2.5. Zliczanie całkowitej liczby fanów strony

```
# Poniższy kod może wymagać przesłania aplikacji do przeglądu
# i zatwierdzenia. Zobacz https://developers.facebook.com/docs/apps/review

# Weźmy na przykład trzech popularnych muzyków i ich identyfikatory stron
taylor_swift_id = '19614945368'
drake_id = '83711079303'
beyonce_id = '28940545600'

# Deklaracja funkcji pomocniczej do pobierania całkowitej liczby fanów („polubień”) strony
def get_total_fans(page_id):
    return int(g.get_object(id=page_id, fields=['fan_count'])['fan_count'])

tswift_fans = get_total_fans(taylor_swift_id)
drake_fans = get_total_fans(drake_id)
beyonce_fans = get_total_fans(beyonce_id)
```

```
print('Taylor Swift: {0} fanów na Facebooku'.format(tswift_fans))
print('Drake: {0} fanów na Facebooku'.format(drake_fans))
print('Beyoncé: {0} fanów na Facebooku'.format(beyonce_fans))
```



W czasie pisania tej książki aplikacja programisty, w celu pobrania zawartości strony publicznej z Facebooka, musiała być przesłana do sprawdzenia i zatwierdzenia za pośrednictwem platformy programisty (<http://bit.ly/2vDb2B1>). Jest to skutek wysiłków Facebooka mających zapewnić tej platformie większe bezpieczeństwo i zapobiegać nadużyciom.

Oto wynik działania tego kodu:

```
Taylor Swift: 73896104 fanów na Facebooku
Drake: 35821534 fanów na Facebooku
Beyoncé: 63974894 fanów na Facebooku
```

Liczba fanów na Facebooku jest pierwszą i najbardziej podstawową miarą popularności strony. Nie wiemy jednak nic o tym, jak aktywni są ci fani, jakie jest prawdopodobieństwo, że zareagują na wiadomość, czy zwrócą na nią uwagę i czy skomentują ją lub udostępnią. Te różne sposoby zaangażowania się w post są ważne dla właściciela strony, ponieważ każdy z fanów prawdopodobnie ma wielu przyjaciół na Facebooku, a algorytm publikacji informacji Facebooka często ujawnia działania określonego użytkownika jego znajomym. Oznacza to, że dzięki aktywnym fanom posty będą widzialne przez większą liczbę osób — a większy rozgłos przekłada się na większą uwagę, więcej fanów, więcej kliknięć, większą sprzedaż lub większą wartość innego wskaźnika, który staramy się zmaksymalizować.

Aby rozpocząć proces pomiaru zaangażowania, trzeba pobrać kanał strony. Obejmuje to nawiązanie połączenia z interfejsem API Graph z wykorzystaniem identyfikatora strony, a następnie wybranie postów. Każdy post będzie zwrócony jako obiekt JSON z dużą ilością metadanych. Python traktuje JSON jako słownik zawierający klucze i wartości.

W listingu 2.6 najpierw definiujemy funkcję, która pobiera kanał strony i scala wszystkie dane kanału w listę zawierającą określoną liczbę postów. Ponieważ interfejs API automatycznie stronicuje wyniki, funkcja kontynuuje stronicowanie do momentu osiągnięcia określonej liczby postów, a następnie zwraca tę liczbę. Teraz, gdy możemy utworzyć listę danych postów za pośrednictwem kanału strony, chcemy mieć możliwość wyodrębnienia z tych postów informacji. Możemy być zainteresowani treścią posta — na przykład wiadomością dla fanów — więc następną funkcją pomocniczą wyodrębnią wiadomość z posta i ją zwraca.

Listing 2.6. Pobieranie kanału strony

```
# Zadeklaruj funkcję pomocniczą do pobierania oficjalnego kanału danej strony
def retrieve_page_feed(page_id, n_posts):
    """Pobierz pierwsze n_posts postów z kanału strony w odwrotnej kolejności
    chronologicznej."""
    feed = g.get_connections(page_id, 'posts')
    posts = []
    posts.extend(feed['data'])

    while len(posts) < n_posts:
        try:
            feed = requests.get(feed['paging']['next']).json()
            posts.extend(feed['data'])
```

```

except KeyError:
    # Przerwij, jeśli nie ma więcej postów w kanale
    print('Osiągnięto koniec postów.')
    break

if len(posts) > n_posts:
    posts = posts[:n_posts]

print('z kanału pobrano {} pozycji'.format(len(posts)))
return posts

# Deklaracja funkcji pomocniczej w celu zwrócenia treści komunikatu posta
def get_post_message(post):
    try:
        message = post['story']
    except KeyError:
        # Post może zawierać pole 'message' zamiast 'story'
        pass
    try:
        message = post['message']
    except KeyError:
        # Post nie zawiera żadnego z tych pól
        message = ''
    return message.replace('\n', ' ')

# Pobranie ostatnich 5 elementów z kanałów
for artist in [taylor_swift_id, drake_id, beyonce_id]:
    print()
    feed = retrieve_page_feed(artist, 5)
    for i, post in enumerate(feed):
        message = get_post_message(post)[:50]
        print('{0} - {1}...'.format(i+1, message))

```

W ostatnim bloku kodu przetwarzamy w pętli posty trzech przykładowych artystów i pobieramy ostatnie 5 postów na ich kanałach, drukując pierwszych 50 znaków każdego z nich. Oto wynik:

```

z kanału pobrano 5 pozycji
1 - Check out a key moment in Taylor writing "This Is ...
2 - ...
3 - ...
4 - The Swift Life is available for free worldwide in ...
5 - #TheSwiftLife App is available NOW for free in the...
z kanału pobrano 5 pozycji
1 - ...
2 - http://www.hollywoodreporter.com/features/drakes-h...
3 - ...
4 - Tickets On Sale Friday, September 15....
5 - https://www.youcaring.com/jjwatt...
z kanału pobrano 5 pozycji
1 - ...
2 - ...
4 - New Shop Beyoncé 2017 Holiday Capsule: shop.beyonc...
5 - Happy Thicccsgiving. www.beyonce.com...

```

Elementy oznaczone samym wielokropkiem (...) nie zawierały tekstu (pamiętaj, że Facebook pozwala także na publikowanie w kanale filmów i zdjęć).

Facebook pozwala swoim użytkownikom na wiele różnych sposobów zaangażować się w post. Przycisk *Lubię to* pojawił się po raz pierwszy w 2009 r. i od tego czasu zmieniał swoją postać we wszystkich mediach społecznościowych. Ponieważ Facebook i wiele innych platform mediów społecznościowych są wykorzystywane przez reklamodawców, przycisk *Lubię to* lub podobny wysyła silny sygnał, że wiadomość współgra z jej docelowymi odbiorcami. Przyniosło to również pewne niezamierzone konsekwencje. Wiele osób zaczęło dostosowywać publikowaną treść w taki sposób, aby generować jak najwięcej „polubień”.

W 2016 r. Facebook rozszerzył zakres możliwych reakcji, dodając reakcje „Super”, „Ha ha”, „Wow”, „Przykro mi” i „Wrr”. Od maja 2017 r. użytkownicy mogą również sygnalizować podobne reakcje na komentarze.

Poza okazywaniem emocji użytkownicy mogą angażować się w posty poprzez komentowanie lub udostępnianie. Każde z tych działań zwiększa prawdopodobieństwo pojawienia się oryginalnego posta w kanałach informacyjnych innych użytkowników. Chcielibyśmy mieć narzędzia umożliwiające zmierzenie liczby użytkowników odpowiadających na posty na dowolnej z tych stron. Dla uproszczenia ograniczyliśmy reakcje emocjonalne do przycisku *Lubię to*, chociaż kod można zmodyfikować w taki sposób, aby zbadać odpowiedzi z większą szczegółowością.

Liczba odpowiedzi wygenerowanych przez post jest w przybliżeniu proporcjonalna do rozmiaru „publiczności” odbiorców posta. Liczba fanów, którą ma strona, jest przybliżonym wskaźnikiem liczby odbiorców posta, chociaż nie wszyscy fani koniecznie muszą widzieć każdy post. To, kto zobaczy post, określa zastrzeżony algorytm dystrybucji informacji Facebooka, a autorzy postów mogą także „zwiększyć” widoczność posta, płacąc Facebookowi opłatę reklamową.

Dla potrzeb tego opisu założymy, że w naszym przykładzie cała baza fanów każdego artysty zawsze zobaczy każdy post. Chcielibyśmy zmierzyć, jaka część bazy fanów artysty w jakiś sposób angażuje się w publikację — klika przycisk *Lubię to*, komentuje lub udostępnia post. Dzięki temu możemy lepiej porównywać bazy fanów każdego artysty: kto ma najbardziej aktywnych fanów? kto najczęściej udostępnia informacje?

Jeśli publikujesz posty związane z jednym z tych artystów i starasz się, by zyskały one jak największy odzew, być może zainteresuje Cię, które posty przynoszą najlepsze wyniki. Czy fani Twojego klienta reagują bardziej na treści wideo, czy na zdjęcia lub tekst? Czy fani są bardziej aktywni w określone dni tygodnia, czy w określonych porach dnia? Ostatnie pytanie dotyczące czasu publikacji posta nie ma większego znaczenia dla Facebooka, ponieważ algorytm publikacji informacji kontroluje, co kto widzi. Jednak pomimo tego być może warto się temu przyjrzeć.

W listingu 2.7 połączyliśmy ze sobą różne elementy. Funkcja pomocnicza `measure_response` zlicza dla każdego posta polubienia, udostępnienia i komentarze. Inna funkcja pomocnicza o nazwie `measure_engagement` porównuje te liczby z całkowitą liczbą fanów strony.

Listing 2.7. Pomiar zaangażowania

```
# Pomiar odpowiedzi na post pod względem liczby polubień, udostępnień i komentarzy
def measure_response(post_id):
    """Zwraca liczbę polubień, udostępnień i komentarzy dotyczących
    danego posta jako miarę zaangażowania użytkowników."""
```

```

likes = g.get_object(id=post_id,
                    fields=['likes.limit(0).summary(true)']\
                    ['likes']['summary']['total_count'])
shares = g.get_object(id=post_id,
                    fields=['shares.limit(0).summary(true)']\
                    ['shares']['count'])
comments = g.get_object(id=post_id,
                    fields=['comments.limit(0).summary(true)']\
                    ['comments']['summary']['total_count'])
return likes, shares, comments

```

Zmierz względny odsetek fanów strony zaangażowanych w post

```

def measure_engagement(post_id, total_fans):
    """Zwraca liczbę polubień, udostępnień i komentarzy dotyczących
    danego posta jako miarę zaangażowania użytkowników."""
    likes = g.get_object(id=post_id,
                        fields=['likes.limit(0).summary(true)']\
                        ['likes']['summary']['total_count'])
    shares = g.get_object(id=post_id,
                        fields=['shares.limit(0).summary(true)']\
                        ['shares']['count'])
    comments = g.get_object(id=post_id,
                        fields=['comments.limit(0).summary(true)']\
                        ['comments']['summary']['total_count'])
    likes_pct = likes / total_fans * 100.0
    shares_pct = shares / total_fans * 100.0
    comments_pct = comments / total_fans * 100.0
    return likes_pct, shares_pct, comments_pct

```

Pobierz ostatnie 5 elementów z kanałów artysty, wyświetl

reakcję i stopień zaangażowania

```

artist_dict = {'Taylor Swift': taylor_swift_id,
              'Drake': drake_id,
              'Beyoncé': beyonce_id}
for name, page_id in artist_dict.items():
    print()
    print(name)
    print('-----')
    feed = retrieve_page_feed(page_id, 5)
    total_fans = get_total_fans(page_id)

    for i, post in enumerate(feed):
        message = get_post_message(post)[:30]
        post_id = post['id']
        likes, shares, comments = measure_response(post_id)
        likes_pct, shares_pct, comments_pct = measure_engagement(post_id, total_fans)
        print('{0} - {1}...'.format(i+1, message))
        print('   Polubień {0} ({1:7.5f}%)'.format(likes, likes_pct))
        print('   Udostępnień {0} ({1:7.5f}%)'.format(shares, shares_pct))
        print('   Komentarzy {0} ({1:7.5f}%)'.format(comments, comments_pct))

```

Przetwarzanie w pętli danych każdego z trzech artystów z naszego przykładu i pomiar zaangażowania fanów zwróciły następujący wynik:

```

Taylor Swift
-----
z kanału pobrano 5 pozycji

```


- 1 - Check out a key moment in Tayl...
 - Polubień 33134 (0.04486%)
 - Udostępnień 1993 (0.00270%)
 - Komentarzy 1373 (0.00186%)
- 2 - ...
 - Polubień 8282 (0,01121%)
 - Udostępnień 19 (0,00003%)
 - Komentarzy 353 (0,00048%)
- 3 - ...
 - Polubień 11083 (0,01500%)
 - Udostępnień 8 (0,00001%)
 - Komentarzy 383 (0,00052%)
- 4 - The Swift Life is available fo...
 - Polubień 39237 (0,05312%)
 - Udostępnień 926 (0,00125%)
 - Komentarzy 1012 (0,00137%)
- 5 - #TheSwiftLife App is available...
 - Polubień 60721 (0,08221%)
 - Udostępnień 1895 (0,00257%)
 - Komentarzy 2105 (0,00285%)

Drake

z kanału pobrano 5 pozycji

- 1 - ...
 - Polubień 23938 (0,06685%)
 - Udostępnień 2907 (0,00812%)
 - Komentarzy 3785 (0,01057%)
- 2 - <http://www.hollywoodreporter.c...>
 - Polubień 4474 (0,01250%)
 - Udostępnień 166 (0,00046%)
 - Komentarzy 310 (0,00087%)
- 3 - ...
 - Polubień 44887 (0,12536%)
 - Udostępnień 8 (0,00002%)
 - Komentarzy 1895 (0,00529%)
- 4 - Tickets On Sale Friday, Septem...
 - Polubień 19003 (0,05307%)
 - Udostępnień 1343 (0,00375%)
 - Komentarzy 6459 (0,01804%)
- 5 - <https://www.youcaring.com/jjwa...>
 - Polubień 17109 (0,04778%)
 - Udostępnień 1777 (0,00496%)
 - Komentarzy 859 (0,00240%)

z kanału pobrano 5 pozycji

- 1 - ...
 - Polubień 8328 (0,01303%)
 - Udostępnień 134 (0,00021%)
 - Komentarzy 296 (0,00046%)
- 2 - ...
 - Polubień 18545 (0,02901%)
 - Udostępnień 250 (0,00039%)
 - Komentarzy 819 (0,00128%)
- 3 - ...
 - Polubień 21589 (0,03377%)
 - Udostępnień 460 (0,00072%)
 - Komentarzy 453 (0,00071%)

- 4 - New Shop Beyoncé 2017 Holiday ...
 - Polubień 10717 (0,01676%)
 - Udostępnień 246 (0,00038%)
 - Komentarzy 376 (0,00059%)
- 5 - Happy Thiccssgiving. www.beyonc...
 - Polubień 25497 (0,03988%)
 - Udostępnień 653 (0,00102%)
 - Komentarzy 610 (0,00095%)

Może się wydawać, że zaangażowanie w post zaledwie 0,04% bazy fanów jest niewielkie. Należy jednak pamiętać, że większość osób w żaden sposób nie reaguje na większość postów, które widzi. A kiedy mamy dziesiątki milionów fanów, mobilizacja nawet niewielkiej ich części może wywierać duży wpływ.

2.3.2. Manipulowanie danymi z wykorzystaniem pakietu pandas

Pythonowa biblioteka pandas stała się niezbędnym narzędziem w zestawie narzędzi każdego inżyniera danych. Będziemy z niej korzystać w różnych częściach tej książki. Zapewnia wydajne struktury danych do przechowywania danych tabelarycznych oraz potężne narzędzia do analizy danych napisane w Pythonie, a niektóre, najbardziej intensywne obliczeniowo części — zoptymalizowane z wykorzystaniem języka C lub technologii Cython. Projekt został rozpoczęty w 2008 r. przez Wesa McKinneya jako narzędzie do analizy danych finansowych.

Jedną z głównych nowych struktur danych dostarczanych przez bibliotekę pandas jest DataFrame — konstrukcja, która przypomina bazę danych lub tabelę. Ma oznakowane kolumny i indeks. Obsługuje „z wdziękiem” brakujące dane, wspiera dane szeregów czasowych i indeksy czasowe, umożliwia łatwe scalanie i tworzenie wycinków danych oraz zawiera wiele przydatnych narzędzi do odczytu i zapisu danych.

Więcej informacji o bibliotece pandas można znaleźć w jej repozytorium GitHub (<http://bit.ly/2C2k4gt>) oraz w oficjalnej dokumentacji (<http://bit.ly/2BRE3vC>). W celu krótkiego wprowadzenia zapoznaj się z samouczkiem zatytułowanym *10 Minutes to pandas* (<http://bit.ly/2Dyd20w>).

Wizualizacja zaangażowania odbiorców z wykorzystaniem matplotlib

Mamy narzędzia potrzebne do pomiaru zaangażowania w posty na Facebooku. Chcemy jednak mieć możliwość łatwego porównywania ze sobą różnych stron (zamiast śledzić strony muzyków, możesz przyglądać się temu, jak dobrze strona Twojej firmy angażuje swoich zwolenników, i porównywać uzyskane wyniki z konkurentami z Twojej branży).

Szukamy sposobu na to, by móc bez problemu zebrać dane z wielu źródeł w jednej tabeli, tak aby było można łatwo manipulować zgromadzonymi w niej danymi i — być może — wygenerować z tych danych kilka przydatnych wykresów. Do tego celu świetnie nadaje się biblioteka Pythona pandas. Zapewnia ona zbiór rozbudowanych struktur danych i narzędzi analitycznych, które znacznie ułatwiają życie inżyniera danych lub analityka.

W poniższym przykładzie będziemy agregować dane ze stron tych samych trzech muzyków, o których wspominaliśmy już w tej części rozdziału. Będziemy przechowywać te dane w obiekcie DataFrame biblioteki pandas — tabelarycznej strukturze danych dostarczanej przez tę bibliotekę.

Instalacja biblioteki pandas sprowadza się do wpisania w wierszu poleceń komendy `pip install -U pandas`. Zastosowanie flagi `-U` daje pewność, że zainstalowano najnowszą wersję.

Zacniemy tak jak w listingu 2.8 od zdefiniowania kolumn pustego obiektu `DataFrame`, który będzie nam służyć do przechowywania danych.

Listing 2.8. Definiowanie pustej struktury `DataFrame` z biblioteki `pandas`

```
import pandas as pd # pip install pandas

# Utwórz obiekt DataFrame przeznaczony na informacje
# pochodzące ze strony artysty
columns = ['Nazwisko',
           'Całk. liczba fanów',
           'Numer posta',
           'Data posta',
           'Nagłówek',
           'Polubień',
           'Udostępnień',
           'Komentarzy',
           'Wzgl. liczba polubień',
           'Wzgl. liczba udostępnień',
           'Wzgl. liczba komentarzy']
musicians = pd.DataFrame(columns=columns)
```

W powyższym kodzie wymieniliśmy interesujące nas kolumny. Podczas przeglądania w pętli postów z kanału każdego z muzyków otrzymamy dla każdego posta liczbę jego polubień, udostępnień i komentarzy, a także wartości względne każdego z tych wskaźników (tzn. ułamek użytkowników, którzy zareagowali, w stosunku do całkowitej liczby fanów). Sposób, w jaki należy to zrobić, pokazano w listingu 2.9.

Listing 2.9. Przechowywanie danych w strukturze `DataFrame` biblioteki `pandas`

```
# Zbuduj obiekt DataFrame przez dodanie danych 10 ostatnich postów i reakcji odbiorców
# dla każdego artysty
for page_id in [taylor_swift_id, drake_id, beyonce_id]:
    name = g.get_object(id=page_id)['name']
    fans = get_total_fans(page_id)
    feed = retrieve_page_feed(page_id, 10)
    for i, post in enumerate(feed):
        likes, shares, comments = measure_response(post['id'])
        likes_pct, shares_pct, comments_pct = measure_engagement(post['id'], fans)
        musicians = musicians.append({'Nazwisko': name,
                                     'Całk. liczba fanów': fans,
                                     'Numer posta': i+1,
                                     'Data posta': post['created_time'],
                                     'Nagłówek': get_post_message(post),
                                     'Polubień': likes,
                                     'Udostępnień': shares,
                                     'Komentarzy': comments,
                                     'Wzgl. liczba polubień': likes_pct,
                                     'Wzgl. liczba udostępnień': shares_pct,
                                     'Wzgl. liczba komentarzy': comments_pct,
                                     }, ignore_index=True)

# Napraw typ kilku kolumn
for col in ['Numer posta', 'Całk. liczba fanów', 'Polubień', 'Udostępnień', 'Komentarzy']:
    musicians[col] = musicians[col].astype(int)
```

Przeładowujemy w pętli strony poszczególnych artystów według identyfikatora strony, pobierając nazwę, liczbę fanów i ostatnie 10 postów z kanału strony. Następnie w wewnętrznej pętli for iterujemy po każdym z 10 postów w kanale i pobieramy takie dane jak całkowita liczba polubień, udostępnień i komentarzy oraz obliczamy, jaki procent całkowitej liczby fanów reprezentują te wartości. Wszystkie te informacje są zapisywane jako wiersz struktury DataFrame. Ustawiając słowo kluczowe ignore_index na True, wskazujemy, że żaden wiersz nie ma z góry określonego indeksu, a biblioteka pandas wyszczególnia wiersze w miarę ich dodawania.

W ostatniej pętli w przykładzie modyfikujemy typ danych (dtype) kilku kolumn danych liczbowych tak, aby poinformować bibliotekę pandas, że są to liczby całkowite, a nie liczby zmiennoprzecinkowe lub inne.

Uruchomienie tego kodu może zająć trochę czasu, ponieważ dane są agregowane z Facebooka przez API. Ostatecznie jednak uzyskamy estetyczną tabelę do przetwarzania. Struktury DataFrame biblioteki pandas oferują zdefiniowaną na nich wygodną metodę .head(), która umożliwia podgląd pierwszych pięciu wierszy tabeli (patrz rysunek 2.6). Zachęcam do przeprowadzania dowolnych eksploracyjnych analiz danych w środowisku Jupyter Notebook (<http://bit.ly/2om1qdG>). Przykłady notatników Jupyter Notebook tego rodzaju dostarczone w repozytorium GitHub⁴ dla tej książki (<http://bit.ly/Mining-the-Social-Web-3E>).

	Name	Total Fans	Post Number	Post Date	Headline	Likes	Shares	Comments	Rel. Likes	Rel. Shares	Rel. Comments
0	Taylor Swift	73862332	1	2017-12-19T17:07:33+0000	Check out a key moment in Taylor writing "This...	33134	1994	1373	0.044859	0.002700	0.001859
1	Taylor Swift	73862332	2	2017-12-17T16:42:38+0000		8282	19	353	0.011213	0.000026	0.000478
2	Taylor Swift	73862332	3	2017-12-17T03:51:04+0000		11083	8	383	0.015004	0.000011	0.000519
3	Taylor Swift	73862332	4	2017-12-16T20:19:52+0000	The Swift Life is available for free worldwide...	39237	925	1012	0.053122	0.001252	0.001370
4	Taylor Swift	73862332	5	2017-12-15T13:18:45+0000	#TheSwiftLife App is available NOW for free in...	60721	1895	2105	0.082210	0.002566	0.002850

Rysunek 2.6. Pierwszych pięć wierszy struktury DataFrame utworzonej ze stron muzyków

Obiekty DataFrame biblioteki pandas zawierają metody wykreślania ułatwiające szybką wizualizację danych. Funkcje te wywołują „pod spodem” bibliotekę matplotlib, należy zatem zadbać o jej zainstalowanie.

Jedną z interesujących własności indeksowania zapewnianych przez bibliotekę pandas zaprezentowano w listingu 2.10. Bierzemy obiekt DataFrame stron muzyków i indeksujemy według zbioru wierszy, w których kolumna Nazwisko pasuje do Drake. Pozwala nam to wykonywać dalsze operacje tylko na danych związanych z Drake’em, a nie na wierszach odpowiadających Taylor Swift lub Beyoncé.

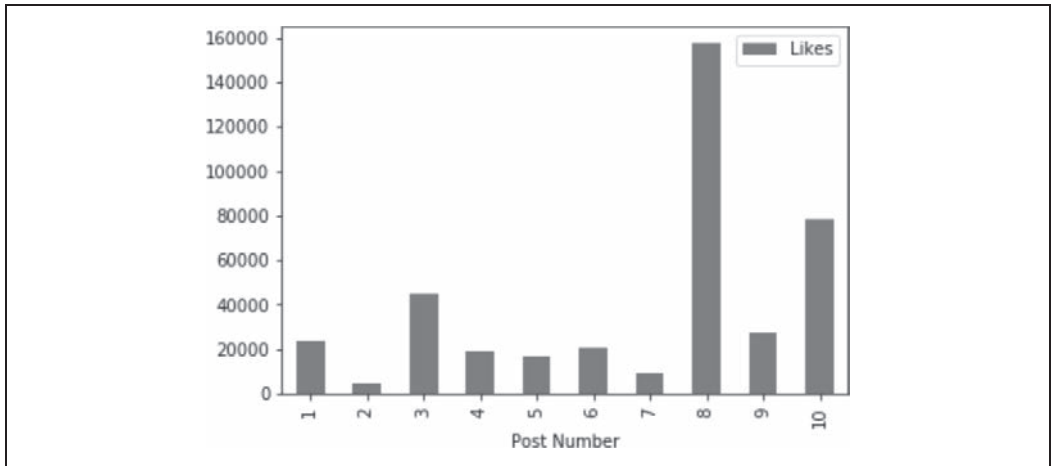
⁴ Ze względu na ciągle zmiany w polityce bezpieczeństwa firmy Facebook, niektóre z przykładów w tym rozdziale mogą nie działać zgodnie z oczekiwaniami — *przyp. red.*

Listing 2.10. Rysowanie wykresu słupkowego na podstawie struktury DataFrame z biblioteki pandas

```
import matplotlib # pip install matplotlib

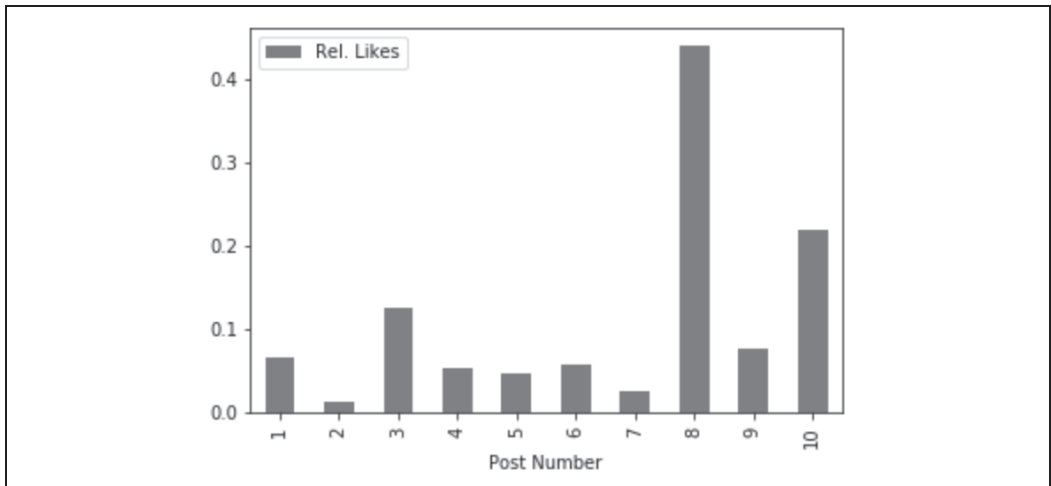
musicians[musicians['Name'] == 'Drake'].plot(x='Post Number', y='Likes', kind='bar')
musicians[musicians['Name'] == 'Drake'].plot(x='Post Number', y='Shares', kind='bar')
musicians[musicians['Name'] == 'Drake'].plot(x='Post Number',
y='Comments', kind='bar')
```

Powyższy kod tworzy wykres słupkowy (patrz rysunek 2.7) liczby polubień wygenerowanych przez ostatnich 10 postów na stronie Drake'a na Facebooku.



Rysunek 2.7. Wykres słupkowy liczby polubień otrzymanych dla ostatnich 10 postów

Wystarczy spojrzeć na rysunek 2.7, aby zobaczyć, że bardzo dobry efekt przyniósł post nr 8, co może nas zainteresować. Równie szybko możemy zobaczyć, jaką część całkowitej liczby fanów zaangażował ten post (rysunek 2.8).



Rysunek 2.8. Wykres słupkowy liczby polubień otrzymanych dla ostatnich 10 postów, podzielonej przez liczbę fanów strony

Post nr 8 zaangażował 0,4% całkowitej liczby fanów, co jest stosunkowo dużą liczbą.

Załóżmy teraz, że chcemy porównać ze sobą trzech artystów. Obecnie indeks struktury DataFrame jest po prostu nudnym numerem wiersza. Możemy jednak go zmienić na coś bardziej opisowego, tak by można było łatwiej manipulować danymi. Zmodyfikujemy nieznacznie strukturę DataFrame poprzez ustawienie indeksu wielokrotnego.

Indeks wielokrotny to indeks hierarchiczny. Najwyższym poziomem będzie dla nas nazwisko artysty: Taylor Swift, Drake lub Beyoncé. Na bezpośrednio niższym poziomie będzie numer posta (od 1 do 10). Podanie artysty i numeru posta łącznie w unikatowy sposób wskaże wiersz w strukturze DataFrame.

Aby ustawić indeks wielokrotny, można posłużyć się kodem z listingu 2.11.

Listing 2.11. Ustawienie indeksu wielokrotnego struktury DataFrame

```
# Zresetuj indeks do indeksu wielokrotnego
musicians = musicians.set_index(['Name', 'Post Number'])
```

Po ustawieniu indeksu wielokrotnego możemy również tworzyć interesujące tabele przestawne za pomocą metody `unstack`, tak jak pokazano w listingu 2.12. Na rysunku 2.9 przedstawiono początkowe wiersze struktury DataFrame po zastosowaniu operacji `unstack`.

Listing 2.12. Użycie metody `unstack` do utworzenia wykresu przestawnego dla struktury DataFrame

```
# Metoda unstack obraca etykiety indeksu
# i pozwala uzyskać kolumny danych pogrupowane według artysty
musicians.unstack(level=0)['Likes']
```

Name	Beyoncé	Drake	Taylor Swift
Post Number			
1	8328	23938	33134
2	18545	4474	8282
3	21589	44887	11083
4	10717	19003	39237
5	25497	17109	60721
6	17744	20328	41359
7	8934	9178	54012
8	10605	157515	29189
9	72254	27186	38439
10	10889	78674	33159

Rysunek 2.9. Obiekt DataFrame uzyskany po uruchomieniu kodu z listingu 2.12

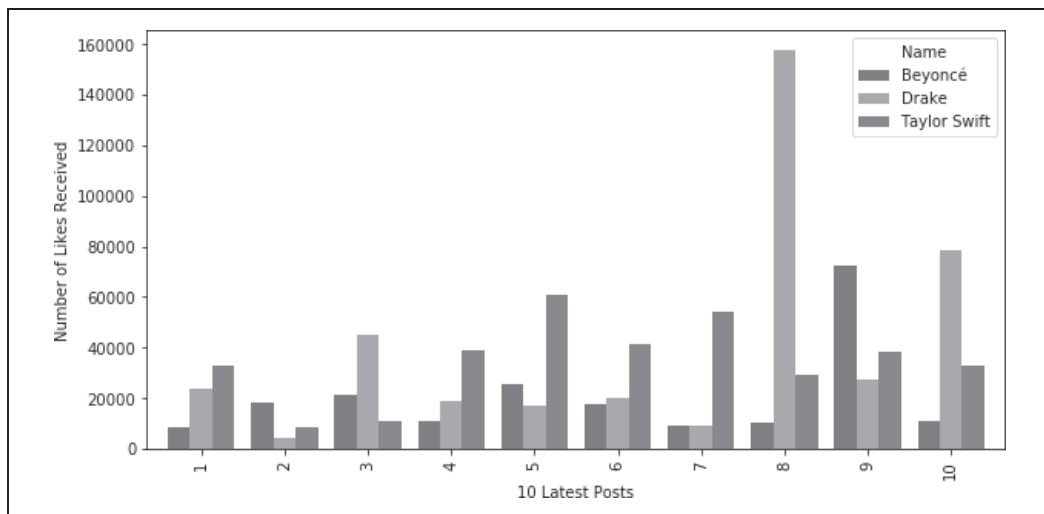
Oczywiście tego rodzaju informacje łatwiej zrozumieć, jeśli zostaną zwizualizowane. Z tego powodu skorzystamy z wbudowanych operacji wykresiania i utworzymy kolejny wykres słupkowy, tak jak w listingu 2.13.

Listing 2.13. Generowanie wykresu słupkowego wszystkich polubień według posta i wykonawcy dla 10 ostatnich postów

```
# Wykreśl reakcje porównawcze dla 10 ostatnich postów na Facebooku dla każdego artysty
plot = musicians.unstack(level=0)['Polubień'].plot(kind='bar', subplots=False,
                                                    figsize=(10,5), width=0.8)

plot.set_xlabel('10 Latest Posts')
plot.set_ylabel('Number of Likes Received')
```

Wyniki zaprezentowane na wykresie pokazano na rysunku 2.10.



Rysunek 2.10. Łączna liczba otrzymanych polubień dla każdego z 10 ostatnich postów dla poszczególnych artystów

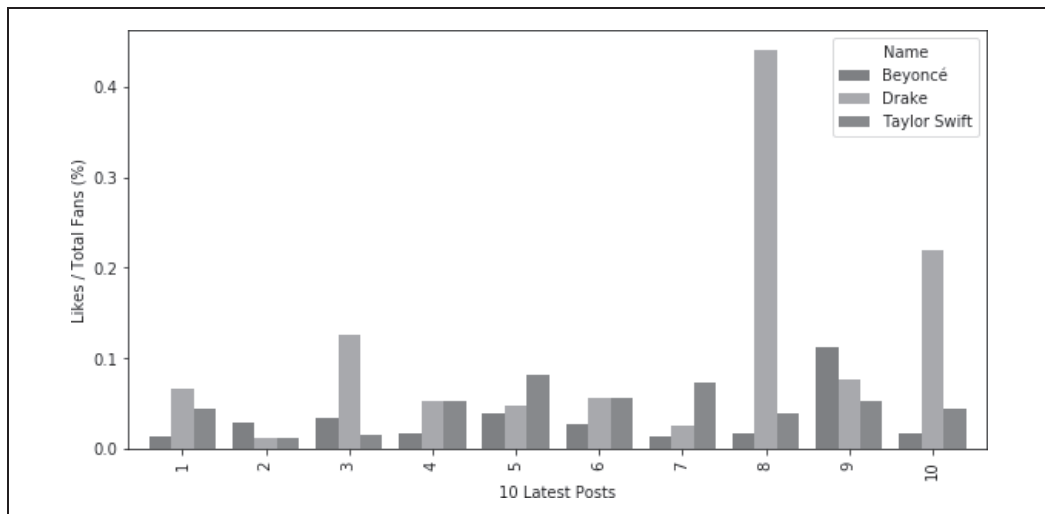
Jest to naprawdę wygodny sposób porównywania ze sobą różnych zestawów danych. Następnie, ponieważ bazy fanów poszczególnych artystów różnią się rozmiarami, znormalizujemy liczby polubień w stosunku do całkowitej liczby fanów (śledzących stronę) każdego artysty, tak jak pokazano w listingu 2.14.

Listing 2.14. Generowanie wykresu słupkowego względnej liczby polubień według posta i wykonawcy dla 10 ostatnich postów

```
# Wykreśl zaangażowanie bazy fanów na Facebooku każdego artysty dla 10 ostatnich postów
plot = musicians.unstack(level=0)['Rel. Likes'].plot(kind='bar', subplots=False,
                                                    figsize=(10,5), width=0.8)

plot.set_xlabel('10 Latest Posts')
plot.set_ylabel('Likes / Total Fans (%)')
```

Wynikowy wykres słupkowy pokazano na rysunku 2.11.



Rysunek 2.11. Względna liczba polubień otrzymanych dla każdego z 10 ostatnich postów dla każdego artysty

Widzimy, że chociaż Drake ma w porównaniu z Beyoncé lub Taylor Swift znacznie mniej fanów na Facebooku, wiele postów na jego stronie na Facebooku potrafi zdobyć sympatię większej części bazy fanów. Chociaż nie jest to rozstrzygające, może to sugerować, że fani Drake’a, choć jest ich mniej, są bardziej lojalni i aktywni na Facebooku. Z drugiej strony może to oznaczać, że Drake (lub jego menedżer ds. mediów społecznościowych) jest bardzo dobry i publikuje na Facebooku bardzo interesujące informacje.

Dokładniejsza analiza może uwzględniać treść każdego posta, typ treści (tekst, obraz lub wideo), język używany w komentarzach oraz inne wyrażane reakcje (poza polubieniami).

Obliczanie średniego zaangażowania

Inną przydatną cechą obiektów DataFrame z indeksami wielokrotnymi jest możliwość obliczania statystyk według indeksu. Przyjrzelśmy się wydajności ostatnich 10 postów dla każdego spośród trzech muzyków. Możemy obliczyć średnią względną liczbę polubień, udostępnień lub komentarzy, które otrzymuje każdy artysta, za pomocą kodu z listingu 2.15.

Listing 2.15. Obliczanie średniego poziomu zaangażowania w ostatnich 10 postach

```
print('Average Likes / Total Fans')
print(musicians.unstack(level=0)['Rel. Likes'].mean())

print('\nAverage Shares / Total Fans')
print(musicians.unstack(level=0)['Rel. Shares'].mean())

print('\nAverage Comments / Total Fans')
print(musicians.unstack(level=0)['Rel. Comments'].mean())
```

Aby przestawić tabelę, ponownie użyliśmy metody unstack. Dzięki temu każda z pierwotnych kolumn („Likes”, „Shares”, „Rel. Comments”, itp.) jest indeksowana nazwiskami artystów. Jedynymi wierszami w obiekcie DataFrame są teraz pojedyncze posty, posortowane według numeru.

W listingu 2.15 pokazano, w jaki sposób można wybrać jedną z interesujących nas zmiennych, na przykład „Rel. Likes”, i obliczyć średnią dla tej kolumny.

Oto wynik działania listingu 2.15:

```
Average Likes / Total Fans
Name
Beyoncé      0.032084
Drake        0.112352
Taylor Swift 0.047198
dtype: float64
```

```
Average Shares / Total Fans
Name
Beyoncé      0.000945
Drake        0.017613
Taylor Swift 0.001962
dtype: float64
```

```
Average Comments / Total Fans
Name
Beyoncé      0.001024
Drake        0.016322
Taylor Swift 0.002238
dtype: float64
```

2.4. Uwagi końcowe

Celem tego rozdziału było zapoznanie się z interfejsem API Graph, pokazanie, w jaki sposób za pomocą protokołu Open Graph można tworzyć połączenia pomiędzy dowolnymi stronami internetowymi i grafem społecznościowym Facebooka oraz jak programowo odpytywać graf społecznościowy, aby uzyskać wgląd w strony Facebooka i własną sieć społecznościową. Jeśli śledziłeś przykłady w tym rozdziale, to nie powinieneś mieć problemów z odpytywaniem grafu społecznościowego w poszukiwaniu odpowiedzi na nurtujące Cię pytania. Pamiętaj, że kiedy eksplorujesz tak obszerny i interesujący zbiór danych, jak wykres społecznościowy Facebooka, potrzebujesz jedynie dobrego punktu wejścia. W trakcie analizowania, jaka powinna być odpowiedź na początkowe pytanie, prawdopodobnie będziesz podążać naturalnym tokiem eksploracji, który stopniowo wpłynie na lepsze zrozumienie przez Ciebie danych i przybliży Cię do odpowiedzi, których szukasz.

Możliwości eksploracji danych na Facebooku są ogromne. Należy jednak szanować prywatność i zawsze respektować warunki korzystania z serwisu Facebook (<http://on.fb.me/1a1lMXM>). W przeciwieństwie do danych z Twittera i niektórych innych źródeł, które są z natury bardziej otwarte, dane z Facebooka mogą być dość wrażliwe, zwłaszcza jeśli analizujesz własną sieć społecznościową. Mamy nadzieję, że ten rozdział pokazał, że istnieje wiele ekscytujących możliwości względem tego, co można zrobić z danymi społecznościowymi, oraz że w Facebooku tkwi ogromna wartość.



Kod źródłowy przykładów z tego rozdziału oraz ze wszystkich innych rozdziałów jest dostępny w serwisie GitHub (<http://bit.ly/Mining-the-Social-Web-3E>) w wygodnym formacie Jupyter Notebook. Zachęcamy do jego wypróbowania we własnej przeglądarce internetowej.

2.5. Zalecane ćwiczenia

- Pobierz dane z fanpage'a poświęconego tematowi, który Cię interesuje na Facebooku, i spróbuj wyciągnąć wnioski dotyczące języka naturalnego w strumieniu komentarzy. Jakie tematy są omawiane najczęściej? Czy możesz wskazać tematy, które powodują, że fani są szczególnie zadowoleni lub zdenerwowani?
- Wybierz dwa różne fanpage'e o podobnym charakterze i je porównaj. Na przykład zobacz, jakie istnieją podobieństwa i różnice pomiędzy fanami Chipotle Mexican Grill i Taco Bell? Czy widzisz coś zaskakującego?
- Wybierz celebrytę lub markę, bardzo aktywne w mediach społecznościowych. Pobierz dużą liczbę ich publicznych postów i sprawdź, jak bardzo fani angażują się w treść. Czy potrafisz wskazać jakieś wzorce? Jakie posty są szczególnie wydajne pod względem polubień, komentarzy lub udostępnień? Czy najbardziej wydajne posty mają ze sobą coś wspólnego? A co z postami o niższym poziomie zaangażowania?
- Liczba obiektów Facebooka dla API Graph jest ogromna. Czy potrafisz przeanalizować takie obiekty jak zdjęcia lub odwiedziny, aby wyciągnąć wnioski dotyczące kogoś w Twojej sieci? Na przykład kto publikuje najwięcej zdjęć i czy na podstawie strumienia komentarzy możesz powiedzieć, o czym one są? Jakie strony najczęściej odwiedzają Twoi znajomi?
- Użyj histogramów (wprowadzonych w punkcie „Wizualizacja danych częstości za pomocą histogramów” w rozdziale 1.), aby dokładniej dzielić i analizować dane ze stron na Facebooku. Utwórz histogram liczby postów według pory dnia. Czy posty pojawiają się o każdej porze dnia i nocy? A może istnieje preferowana pora dnia na publikację?
- Ponadto spróbuj zmierzyć liczbę polubień, które otrzymują posty w zależności od pory dnia, w której zostały opublikowane. Marketerzy mediów społecznościowych bardzo dbają o maksymalizację oddziaływania posta, a czas odgrywa ważną rolę, chociaż — biorąc pod uwagę algorytm publikowania informacji na Facebooku — trudno powiedzieć, kiedy dokładnie publikowane posty dotrą do odbiorców.

2.6. Zasoby online

Warto przejrzeć następujące zasoby online:

- Facebook Developers (<http://bit.ly/1a1lm3Q>);
- Facebook Developers — dokumentacja stronicowania (<http://bit.ly/1a1ltMP>);
- Facebook Platform Policy (<http://bit.ly/1a1lm3C>);
- Graph API Explorer (<http://bit.ly/2jd5Xdq>);
- API Graph — przegląd (<http://bit.ly/1a1lobU>);
- dokumentacja API Graph (<http://bit.ly/1a1lvEr>);

- metaelementy HTML (<http://bit.ly/1a1lBMa>);
- OAuth (<http://bit.ly/1a1kZWN>);
- protokół Open Graph (<http://bit.ly/1a1lu3m>);
- biblioteka requests Pythona (<http://bit.ly/1a1lrEt>);
- RDFa (<http://bit.ly/1a1lujR>).

A

- adres URL, 105
- algorytmy
 - klasteryzacji, 147
 - Luhna, 220
 - tworzenia streszczenia dokumentu, 214
- analiza
 - bigramów, 184
 - częstości, 330
 - danych, 193
 - fanpage'a, 63
 - grafów zainteresowań, 280
 - internetowej poczty, 264
 - klastrów, 133
 - korpusu Enron, 249
 - nadawców wiadomości, 253
 - obiektów, 222
 - obrazów, 108, 116
 - połączeń grafu społecznościowego, 75
 - retweetów, 53
 - runtime, 151
 - stron Facebooka, 78
 - tweetów, 49, 346, 351
 - własnych danych pocztowych, 258
 - znaków, 46
- API, 13
- API Dashboard, 116
- API Enterprise Twittera, 323
- API Facebooka, 76
- API GitHuba, 270
 - analiza grafów zainteresowań, 280
 - krawędzie „śledzi”, 287
 - modelowanie danych, 277
 - połączenie do serwisu, 272
 - śledzenie użytkowników, 287
 - token dostępu, 272
 - tworzenie wydajnych zapytań, 296
 - żądania HTTP, 274
- API Google Cloud Vision, 116
 - rozpoznawanie twarzy, 121
- API Graph, 64, 66
 - debugowanie, 67
 - eksploracja, 81
 - format odpowiedzi JSON, 67
 - identyfikatory węzłów, 67
 - ograniczenia polubień, 67
 - ograniczenia połączenia, 67
 - token dostępu, 66
 - wykonywanie żądań, 69
- API Graph Facebooka, 64
- API Instagrama, 101
 - anatomia posta, 105
 - odczytywanie własnego kanału, 103
 - struktura adresu URL, 105
 - tworzenie żądań, 101
 - uwierzytelnianie, 101
 - wyświetlanie zdjęć i podpisów, 104
 - zarządzanie klientami, 102
- API LinkedIn, 128
 - dostęp do własnych danych, 130
 - grupowanie danych, 132
 - informacje o lokalizacji, 158
 - kartogram, 143
 - klasteryzacja sieci, 159
 - liczenie
 - firm, 135
 - lokalizacji, 140
 - nazw stanowisk, 137
 - limity dostępu, 131
 - mierzenie podobieństwa, 145
 - normalizacja danych, 135
 - tworzenie żądań, 129

API Retweet, 53
API Streaming Twittera, 323
API Twittera, 30, 33
 dostęp do interfejsu, 310, 311
 pobieranie danych szeregów czasowych, 324
 Trends, 315
 wyodrębnianie podmiotów, 326
API Yahoo!, 60
aplikacja, *Patrz* narzędzie
artefakty statystyczne, 226
Atom, 200
autoryzacja aplikacji Jupyter Notebook, 36

B

baza danych MNIST, 110
bazy danych dużych grafów, 280
biblioteka
 boilerpipe, 199, 200
 json, 106
 matplotlib, 55, 88, 113
 pandas, 88, 247, 250
 indeksowanie, 257
 odpytywanie obiektów DataFrame, 257
 requests, 103
 scikit-learn, 111
 tweepy, 33
 twitter, 33
bigramy, 184, 188
błędy HTTP, 334
BSON, 353

C

centralność
 bliskości, 285
 pośrednictwa, 284
 stopniowa, 284
centroid, 160
chi-kwadrat, 192
crawling, 199, 202
 grafu znajomości, 344
częstość
 odwrotna częstość dokumentu, 168
 surowa, 191
 terminu, 166
 wykres rozkładu, 175
czułość, 231
czyszczenie zawartości HTML, 165

D

dane
 Enron, 243
 pocztowe, 238
 analiza, 258
 z plików tekstowych, 163
debugowanie, 67
dekodowanie składni, 205
Docker, 25
dodawanie repozytoriów do grafu, 293
dokument
 Facebook Platform Policy, 65
 Graph API Overview, 65
dokumentacja
 Instagrama, 105
 PyGithub, 276
dostęp
 do API, 35
 GitHuba, 272
 LinkedIn, 129
 Twittera, 310, 311
 do platformy Facebook, 65
 do skrzynki Gmail, 260

E

eksploracja
 API Graph, 81
 danych, 17
 danych z plików tekstowych, 163
 Facebooka, 63
 grafu, 294
 Instagrama, 99
 serwisu GitHub, 269
 sieci LinkedIn, 127
 skrzynek pocztowych, 237
 stron internetowych, 197
 Twittera, 27
ekstrakcja podmiotów, 224
Enron
 analiza korpusu, 249
 konwersja korpusu, 245
 pobieranie danych, 243

F

Facebook

- analizowanie

 - fanpage'y, 63

 - stron, 78

- API Graph, 64, 66

- dostęp do platformy, 65

- eksploracja, 63

- informacje o OAuth, 66

- liczba fanów, 83

- manipulowanie danymi, 88

- mierzenie

 - popularności, 82

 - zaangażowania fanów, 85

 - protokół Open Graph, 70

- fanpage, 63

- folksonomie, 30

- format

 - CSV, 133

 - JSON, 39, 67, 319

 - mbox, 239, 245

- FQL, Facebook Query Language, 66

- framework Immersion, 264

- funkcje indeksowania, 250

G

- generowanie wykresu wizualizacji, 183

- Gephi, 302

- GitHub, 16

 - eksploracja serwisu, 269

- Google Earth, 158

- graf

 - latawca Krackhardta, 285

 - społecznościowy, 70, 75

 - zainteresowania, 280, 287

 - miary centralności, 290

 - repozytoria, 292

 - wizualizacja, 301

 - znajomości

 - crawling, 344

- grafy

 - eksplorowanie, 294

 - krawędzie „śledzi”, 288

 - miary centralności, 284, 286, 290

 - właściwości, 277

- Graph API Explorer, *Patrz* API Graph

- Graphviz, 301

- grupowanie danych, 132

H

- hashtag, 37, 105

- histogram, 55

 - częstości retweetów, 58

- HTML, 201

I

- identyfikatory węzłów, 67

- IDF, Inverse Document Frequency, 164

- iloczyn skalarny wektorów, 180

- IMAP, 259

- Immersion, 264

 - wizualizacja poczty elektronicznej, 264

- implementacja Open Graph, 70

- indeksowanie, 247, 248, 250, 257

- informacje

 - o lokalizacji, 158

 - o maszynie wirtualnej, 357

 - o OAuth, 66

 - o profilu użytkownika, 337

- Instagram

 - eksploracja, 99

- instalowanie pakietu scikit-learn, 111

- interakcje, 227

- interfejs API, *Patrz* API

J

- jakość analiz, 230

- język

 - FQL, 66

 - Python, 14

- JSON, 39, 67

- Jupyter Notebook, 14, 363

 - autoryzacja aplikacji, 36

 - eksploracja API Graph, 81

 - wykresy, 55

K

- kartogram, 143

 - Dorlinga, 144

klasa
 BigramAssocMeasures, 186
 Extractor, 200
 GraphAPI, 75
 LinkedInApplication, 130
 PunktSentenceTokenizer, 212

klasteryzacja, 133
 algorytmy, 147
 hierarchiczna, 152
 metodą k-średnich, 154
 postów, 180
 redukcja wymiarów, 134
 zachłanna, 148

klasyfikator wielowarstwowego perceptron, MLP, 111

klient
 Apple Mail, 259
 poczty, 241

kolokacja, 186

komunikacja nadawca-odbiorca, 253

kontyngencja, 188

konwertowanie formatu mbox, 247

korpus
 Enron, 245
 pocztowy, 239

L

liczba polubień, 94

liczenie
 firm, 135
 lokalizacji, 140
 nazw stanowisk, 137

limity dla API Twittera, 38

LinkedIn
 eksploracja, 127

M

manipulowanie danymi, 88

maszyna wirtualna, 357

mechanizm firehose Twittera, 323

metadane, 72

metoda k-średnich, 154, 157

miara dokładności, 231

miary centralności
 grafu, 284
 zastosowanie, 290

miejsca tweeta, 31

mierzenie
 podobieństwa, 145
 popularności, 81

modelowanie danych GitHuba, 277

MongoDB
 zapisywanie danych JSON, 320

N

narzędzie
 API Google Cloud Vision, 116
 API Upgrade Tool, 66
 Gephi, 302
 Graph API Explorer, 65, 66
 Graphviz, 301
 Immersion, 264
 Jupyter Notebook, 14
 Scrapy, 235
 TweetDeck, 31
 wiersza polecenia Twurl, 60

n-gramy, 185

NLP, Natural Language Processing, 197
 analiza danych, 198
 dzielenie na fragmenty, 209
 ekstrakcja, 209
 odkrywanie semantyki, 205
 oznaczanie części mowy, 208
 podsumowania danych, 226
 przetwarzanie danych, 230
 tokenizacja, 208
 wykrywanie
 zakończeń zdań, 207
 zdań, 210

NLTK, Natural Language Toolkit, 145, 163, 172, 234
 analiza
 bigramów, 184
 danych, 193
 obliczanie
 kolokacji, 186
 podobieństwa kosinusowego, 180
 wyodrębnianie podmiotów, 224
 zastosowanie współczynnika TF-IDF, 176

normalizacja danych, 133, 135

O

- OAuth, 359
 - 1.0a, 360
 - 2.0, 361
 - dostęp
 - do API Twitera, 311
 - do skrzynki Gmail, 260
- obiekty DataFrame, 247, 257
 - indeksowanie, 248
- obliczanie
 - kolokacji, 186
 - miar centralności, 284, 286, 296
 - podobieństwa, 134
 - podobieństwa kosinusowego, 180
 - średniego zaangażowania, 94
 - wskaźnika TF-IDF, 169
- OCR, 110
- odległość
 - edycji, 146
 - Jaccarda, 147
- odpytywanie
 - API Graph, 76
 - danych, 172
 - obiektów DataFrame, 257
- OGP, *Patrz* Open Graph
- ograniczenia
 - polubień, 67
 - połączenia, 67
- Open Graph, 70
 - dostarczanie metadanych, 72
 - implementacja, 70
- operacje na zbiorach, 254
- optyczne rozpoznawanie znaków, OCR, 110
- oś czasu, 31

P

- pakiet, *Patrz także* biblioteka
 - facebook-sdk, 75
 - feedparser, 200
 - mailbox, 242
 - matplotlib, 55, 88, 113
 - NLTK, 145, 163, 172, 234
 - pandas, 88, 247, 250
 - prettytable, 50
 - pydoc, 33
 - PyGithub, 276
 - twitter, 34

- parsowanie, 199
 - wiadomości e-mail, 262
- perceptron, 111
- piaskownica, 101, 107
- plik Dockerfile, 25
- pliki
 - CSV, 132
 - mbox, 239
 - tekstowe, 164
 - eksploracja danych, 163
 - zapisywanie danych JSON, 319
- PMI, Pointwise Mutual Information, 192
- pobieranie
 - danych
 - Enron, 243
 - szeregów czasowych, 324
 - informacji o profilu, 337
 - kanału strony, 83
 - medium, 105
 - połączeń LinkedIn, 132
 - próbek, 323
 - trendów, 37
 - wiadomości e-mail, 262
- podmioty tweeta, 31
- podobieństwo
 - Jaccarda, 190, 191
 - kosinusowe wektorów, 177, 179
 - n-gram, 146
- podsumowania danych, 226
- połączenie z API Twitera, 33
- pomiar zaangażowania fanów, 85
- POP3, Post Office Protocol, 259
- posty na Instagramie, 105, 119
- poświadczenia LinkedIn OAuth, 35, 130
- prawo Zipfa, 174
- precyzja, 231
- protokół
 - IMAP, 259
 - Open Graph, 70
 - POP3, 259
- próbkowanie mechanizmu firehose, 324
- przechowywanie danych, 89
- przepływy pracy, 55
- przestrzeń wektorowa, 177
- przeszukiwanie wszcz, 202–204
- pulpit API Dashboard, 117
- punkty przestawne, 296
- PyGithub, 276

- Python, 14, 363
 - dokumentacja, 15
 - moduł collections, 49
 - odpytywanie API Graph, 76
 - rozpakowywanie wartości, 42
 - system pomocy, 33
 - wizualizacja danych, 113

R

- repozytorium
 - Git, 16
 - GitHub, 25
- RODO, 17
- rozdzielczość encji, 81
- rozkład
 - częstości, 49, 175
 - normalny, 190
- rozpoznawanie
 - cyfr, 111
 - obiektów, 116
 - twarzy, 121
 - znaków, 110
- różnorodność leksykalna, 51
- RSS, Really Simple Syndication, 200

S

- scraping, 199
- Scrapy, 235
- semantyka, 205
- serwis GitHub, 269
- sieci neuronowe, 108
 - konwolucyjne, 112
 - neurony, 109
 - posty na Instagramie, 119
 - rozpoznawanie
 - cyfr, 111
 - obiektów, 116
 - znaków, 110
 - trenowanie, 109
 - warstwy, 109
 - wizualizacja
 - macierzy wag, 114
 - warstwy, 113
- skrzynki pocztowe
 - analiza nadawców i odbiorców, 254
 - eksploracja, 237
 - Gmail, 260

- konwersja
 - na format mbox, 245
 - na JSON, 242
 - na obiekty DataFrame, 247
- liczba wiadomości, 252
- parsowanie wiadomości, 262
- pobieranie wiadomości, 262
- przeglądanie nagłówków, 241
- słowa kluczowe, 257
- uniksowe, 239
- wyszukiwanie wiadomości, 257
- wyświetlanie treści wiadomości, 262
- zakres dat wiadomości, 250
- zapytania, 250

- słownik, 247
- słowo kluczowe pagination, 107
- streszczenie dokumentu, 214
 - algorytm, 214
 - algorytm Luhna, 220
 - wizualizacja wyników, 218
- stronicowanie, 107
- strony internetowe
 - crawling, 199, 202
 - eksploracja, 197
 - parsowanie, 199
 - scraping, 199
- struktura DataFrame, 89
- system Yahoo!, 36
- sztuczne sieci neuronowe, *Patrz* sieci neuronowe

T

- tabela kontyngencji, 188
- tabularyzacja analizy częstości, 330
- taksonomie, 30
- technologia
 - Docker, 25
- teoria zbiorów, 40
- TF-IDF, 166, 169
 - częstość terminu, 166
 - język naturalny, 176
 - obliczanie wskaźnika, 169
 - odpytywanie danych, 172
 - odwrotna częstość dokumentu, 168
- token dostępu, 66
- tokenizer, 213
- trendy, 36
 - na Twitterze, 315
- trening sieci neuronowej, 109

TweetDeck, 31
tweety, 30, *Patrz także* Twitter
analiza
 częstości, 49
 treści, 346
 ulubionych, 350
 znaków, 46
miejsce, 31
podmiot, 31
różnorodność leksykalna, 51
wyodrębnianie podmiotów, 47, 326, 338
wyszukiwanie, 40, 316
 najpopularniejszych obiektów, 329
 najpopularniejszych tweetów, 327

Twitter
analiza
 obserwatorów użytkownika, 341
 znajomych, 341
API, 30, 33
API Search, 316
autoryzacja Jupyter Notebook, 36
crawling grafu znajomości, 344
domowa oś czasu, 31
dostęp do danych, 36
eksploracja, 27
informacje o profilu użytkownika, 337
limit liczby żądań, 37
pobieranie wszystkich znajomych, 339
receptury, 307, 309
retweety
 badanie wzorców, 53
 wyodrębnianie przypisania, 333
streszczenia celów łączy, 347
tabularyzacja analizy częstości, 330
tweety, 30
wizualizacja danych, 55
zbieranie tweetów użytkownika, 342
znajdowanie użytkowników, 331
żądania odporne na błędy, 334

tworzenie
 kartogramu Dorlinga, 144
 połączenia do API GitHuba, 272
 streszczeń
 celów łączy, 347
 dokumentów, 214
 struktury, 89
 wydajnych zapytań, 296

żądań do API
 GitHuba, 275
 Instagrama, 101
 LinkedIn, 129

Twurl, 60
typ MIME, 241

U

uczenie maszynowe, 100
ujednoznacznienie encji, 81
układ
 dendrogramu, 155
 drzewa, 156
uwierzytelnianie w API Instagrama, 101

W

wektor, 179
węzeł, 296
 search_metadata, 41
wiadomości e-mail
 analiza nadawców i odbiorców, 254
 parsowanie, 262
 pobieranie, 262
 wyszukiwanie według słów kluczowych, 257
 zakresy dat, 250
wizualizacja
 danych, 55, 113
 geoprzestrzenna lokalizacji
 centroidów, 160
 kontaktów, 160
 grafów zainteresowań, 301
 interakcji, 229
 klastrow geograficznych, 158
 lokalizacji, 143
 poczty elektronicznej, 264
 podobieństwa dokumentu, 182
 zaangażowania odbiorców, 88
wskaźnik podobieństwa
 odległość edycji, 146
 odległość Jaccarda, 147
 podobieństwo n-gram, 146
 TF-IDF, 169
współczynnik
 PMI, 192
 prawdopodobieństwa, 192

- współczynnik
 - Sørensen, 191
 - t-Studenta, 192
 - wiarygodności, 190
- wykres
 - macierzowy, 182
 - słupkowy polubień, 93
 - wizualizacji podobieństwa kosinusowego, 183
- wykrywanie
 - twarzy, 121
 - zdań, 210
- wyodrębnianie
 - podmiotów, 224
 - podmiotów tweeta, 326, 338
 - przypisania retweeta, 333
- wywołania funkcji, 318
- wywołanie functools.partial, 318

X

- XHTML, 201
- XML, 201

Z

- zapisywanie danych JSON
 - MongoDB, 320
 - plik tekstowy, 319
- zapytania do końcowego grafu, 298
- zastosowanie miar centralności, 290
- zbieranie tweetów użytkownika, 343

- zbiory
 - operacje, 254
 - trendów, 39
- zliczanie liczby fanów, 82
- zmiana paradygmatu, 222
- znajdowanie
 - najpopularniejszych obiektów, 329
 - najpopularniejszych tweetów, 327
 - podobnych dokumentów, 177, 180
 - tweetów, 40, 316
 - wiadomości e-mail, 257
 - retweetów, 53
 - użytkowników, 331

Ż

- żądania
 - do API
 - GitHuba, 275
 - Instagrama, 101
 - LinkedIn, 129
 - HTTP, 274
 - odporne na błędy, 335

PROGRAM PARTNERSKI

— GRUPY HELION —

1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 

Jakie informacje dziś znajdziesz dzięki danym z Facebooka?

Internetu nie można rozważać wyłącznie jako tworu techniki. Powstanie tej sieci doprowadziło do rozwoju różnych zjawisk społecznych. Z tej perspektywy na szczególną uwagę zasługują media społecznościowe. Są źródłem informacji, które, właściwie spożytkowane, mogą przynieść niezły dochód. Mogą też dać odpowiedzi na wiele pytań zadawanych przez naukowców z różnych branż. Sama eksploracja tych danych przynosi sporo satysfakcji i radości. Zaskakujące przy tym jest to, że przygotowanie zestawu potrzebnych narzędzi i nauka posługiwania się nimi zabiera naprawdę niewiele czasu i nie wymaga specjalnych talentów!

To trzecie, zaktualizowane wydanie popularnego podręcznika dla osób, które chcą zająć się wydobywaniem danych z sieci społecznościowych. Uwzględniono tu zmiany interfejsów API wprowadzone do poszczególnych platform i dodano rozdział o eksploracji Instagrama. Dowiesz się, jak dzięki danym z mediów społecznościowych określić sieć powiązań użytkowników, zorientować się, kto o czym mówi i gdzie się znajduje. Treść bogato zilustrowano przykładami kodu w Pythonie, a także plikami Jupyter Notebook lub kontenerów Dockera. Ciekawym elementem książki jest zbiór receptur dotyczących rozwiązywania konkretnych problemów z Twitterem.

Matthew A. Russell jest liderem technicznym. Wychowuje liderów i buduje zespoły, które rozwiązują trudne problemy. Pochodzi ze stanu Tennessee.

Dr Mikhail Klassen jest głównym inżynierem danych w startupie Paladin AI. Pasjonat sztucznej inteligencji i nowych rozwiązań w dziedzinie nauki o danych. Stosuje w praktyce techniki eksploracji danych i uczenia maszynowego.

W tej książce między innymi:

- wprowadzenie do świata mediów społecznościowych
- przybliżenie bogactwa danych zawartych w mediach społecznościowych
- eksploracja danych za pomocą narzędzi Pythona 3
- zaawansowane techniki eksploracji danych, w tym współczynniki TFIDF, podobieństwo kosinusów i rozpoznawanie obrazów

Helion

 helion.pl
 **HELION SA**
 ul. Kościuszki 1c
 44-100 Gliwice
 tel.: 32 230 98 63
 helion@helion.pl

Sprawdź nasze szkolenia

SZKOLENIA
AKADEMIA IT & BUSINESS
 WWW.SZKOLENIA.HELION.PL

KOD KORZYŚCI
 Sięgnij po więcej! ▶



ISBN 978-83-283-5554-5

