METODY AUTOMATYCZNEGO POZYSKIWANIA DANYCH Z INTERNETU DLA CELÓW BIZNESOWYCH I CYBERBEZPIECZEŃSTWA

ZESPÓŁ AIGOCRACY INSTITUTE

SYGN. WIB PAB 11/2020

Raport opracowany na zlecenie Programu Analityczno-Badawczego Fundacji Warszawski Instytut Bankowości

Poznań, 14.03.2020

PROGRAM ANALITYCZNO BADAWCZY

O Raporcie

Raport **Metody automatycznego pozyskiwania danych z internetu dla celów biznesowych i cyberbezpieczeństwa** został opracowany na zlecenie Programu Analityczno-Badawczego Fundacji Warszawski Instytut Bankowości.

Autorzy:

Zespół autorów raportu:

prof. Witold Abramowicz
dr Elżbieta Lewańska
dr Milena Stróżyna
dr Marcin Szmydt
Martyna Chmielewska
Oskar Dołżkiewicz
Wojciech Dopieralski
Alex Drożdż
Damian Klimarczyk
Eryk Rutkowski
Joanna Sobkowiak
Miłosz Sojka

AIGOCRACY INSTITUTE
POWERED BY SYNERISE

Aigocracy Institute sp. z o.o. – instytut powstały w 2018 roku zorientowany jest na działania z obszaru Badań i Rozwoju (R&D), doradztwa oraz implementacji najnowszych rozwiązań w dziedzinie zaawansowanej analityki biznesowej, wykorzystującej rozwiązania z obszaru data science, sztucznej inteligencji oraz uczenia maszynowego. W zespole AGI znajdują się osoby posiadające wieloletnie doświadczenie w zróżnicowanych obszarach, takich jak biometria, wykorzystanie danych z IoT, analiza oraz prognozowanie ryzyka i anomalii, m.in. w transporcie morskim, technologiach semantycznych, profilowaniu behawioralnym, ekstrakcji danych z nieustrukturyzowanych źródeł, technologiach Big Data, algorytmach sztucznej inteligencji. Doświadczenie zdobyto zagranicą i w Polsce w trakcie realizacji szeregu projektów przemysłowych i badawczych. Aigocracy Institute zapewnia interdyscyplinarne oraz zindywidualizowane podejście do problemów biznesowych. Dzięki doświadczeniu pracowników AGI we współpracy z podmiotami z różnych branż, dysponują oni multidyscyplinarnym know-how i są w stanie w sposób holistyczny adresować problematykę analizy danych w zróżnicowanych zastosowaniach.

PROGRAM ANALITYCZNO BADAWCZY

O Programie

Program Analityczno-Badawczy przy Fundacji WIB powstał w 2019 roku jako odpowiedź na potrzeby sektora bankowego w zakresie analizy zjawisk, tworzenia opracowań i porządkowania wiedzy w obszarach cyberbezpieczeństwa i nowych technologii, a także szeroko rozumianego otoczenia sektora bankowego, kształtującego warunki działania banków w Polsce. Prace analityczno-badawcze w ramach programu realizowane są pod kątem możliwości praktycznego wykorzystania ich wyników w celu rozwoju sektora bankowego, podnoszenia poziomu bezpieczeństwa usług oraz kreowania wartości dla klientów bankowości. PAB WIB kładzie duży nacisk na rozwój współpracy ze środowiskami akademickimi i eksperckimi, poszukując synergii w zakresie zainteresowań badawczych autorów oraz potrzeb rozwojowych sektora bankowego.

W ramach programu realizowane są analizy i badania w następujących obszarach:

- **Nowe technologie i cyberbezpieczeństwo**
- **Zdolność banków do finansowania gospodarki**
- **Bankowość spółdzielcza**
- **Rynek nieruchomości**
- **Zielony ład i finansowanie energetyki odnawialnej**
- **Finansowanie projektów innowacyjnych**

Więcej na temat działalności programu na stronie www.pab.wib.org.pl

Kontakt:

dr Andrzej Banasiak
Koordynator Programu
m: 696 405 104
e: abanasiak@wib.org.pl

Jacek Gieorgica
m: 603 626 254
e: jgieorgica@wib.org.pl

Warszawski Instytut Bankowości
00-394 Warszawa, ul. Solec 38, lok 104
t: (22) 182 31 70
e: pab@wib.org.pl

Agnieszka Nierodka
m: 607 484 391
e: anierodka@wib.org.pl

dr Tomasz Pawlonka
m: 505 917 778
e: tpawlonka@wib.org.pl

Spis treści

Streszczenie kierownicze 5
Zawartość raportu i jego cele 5
Wnioski 5

ROZDZIAŁ I. Wstęp 6

ROZDZIAŁ II. Potrzeby informacyjne sektora bankowego 8
2.1 Potrzeby informacyjne 9
2.2 Propozycje struktury raportów 10

ROZDZIAŁ III. Klasyfikacja źródeł danych 13
3.1 Rankingi 16
3.2 Fora internetowe 17
3.3 Portale branżowe 18
3.4 Strony domowe banków 19
3.5 Media społecznościowe 20
3.6 Strony instytucji 21
3.7 Podsumowanie analizy źródeł 21

ROZDZIAŁ IV. Narzędzia do automatycznego pozyskiwania danych z Internetu 23
4.1 Web Scraping 24
4.2 Uwarunkowanie prawne, etyczne oraz techniczne 26
4.3 Ograniczanie możliwości pobierania danych 28
4.4 Narzędzia 30
4.4.1 API 30
4.4.2 Beautiful Soup 4 32
4.4.3 Scrapy 33
4.4.4 Rvest 36
4.4.5 Selenium 36
4.5 Porównanie narzędzi 38

ROZDZIAŁ V. Przypadki użycia 40
5.1 Pobieranie danych ze strony HTML – przykład 1 41
5.2 Pobieranie danych przez API 42
5.3 Pobieranie danych z udostępnionych plików 43
5.4 Pobieranie danych ze strony HTML – przykład 2 44

ROZDZIAŁ VI. Wnioski i rekomendacje 47
Wnioski i rekomendacje 49

Bibliografia 51

Streszczenie kierownicze

Zawartość raportu i jego cele

Celem raportu jest przedstawienie przypadków użycia źródeł danych dostępnych w Internecie do wsparcia procesów biznesowych, marketingowych oraz związanych z cyberbezpieczeństwem w sektorze bankowym. Raport omawia podstawowe zagadnienia związane z technicznym pozyskiwaniem danych (np. opis wybranych narzędzi), jak również ujęcie ekonomiczne (np. możliwe do uzyskania efekty skali, szacunkowe koszty wykorzystania wybranych źródeł) i prawne (bieżące rozwiązania legislacyjne, licencje źródeł danych).

Wnioski

- Narzędzia do automatycznego pobierania treści ze źródeł internetowych są wysoce personalizowane (ściśle dostosowane do wybranego źródła), decyzję o stworzeniu takiego narzędzia powinna więc poprzedzać wnikliwa analiza potrzeb informacyjnych oraz struktury, charakterystyk i jakości możliwych do ich zaspokojenia źródeł.

- Przegląd źródeł danych wykazał, że wiele ze źródeł związanych z sektorem bankowym ma postać nieustrukturyzowaną lub półstrukturyzowaną, co utrudnia ich automatyczne przetwarzanie.

- Źródła związane z cyberbezpieczeństwem zawierają najczęściej dane nieustrukturyzowane: są to głównie newsy na portalach branżowych lub komentarze na forach i w mediach społecznościowych. Widoczna jest potrzeba stworzenia bazy zagrożeń związanych z sektorem bankowym (podobnej do baz gromadzących luki bezpieczeństwa w oprogramowaniu, np. https://nvd.nist.gov/vuln/search).

- Agregacja danych pochodzących z różnych źródeł może umożliwić podwyższenie jakości gromadzonych danych (np. dzięki weryfikacji tych danych w różnych źródłach lub uzupełnianiu brakujących elementów).

- Kluczowym ograniczeniem automatycznego pobierania danych z Internetu są przepisy prawa i licencje obejmujące źródła.

- Na rynku dostępnych jest wiele narzędzi służących do budowania web scraperów. Są to często narzędzia darmowe. Wybór konkretnego uwarunkowany jest (1) posiadanymi zasobami technicznymi i ludzkimi, (2) rodzajem i strukturą źródła, z którego mają być pobierane dane.

- Ponieważ koszty opracowania i utrzymania narzędzi do automatycznego pobierania danych z wielu źródeł będą rosły szybciej niż liniowo przy dodawaniu kolejnych źródeł (ze względu na konieczność integracji danych z różnych źródeł oraz ze względu na rosnące wymagania infrastrukturalne), może być zasadne stworzenie narzędzia służącego do pozyskiwania, agregacji i dostarczania danych różnym podmiotom z sektora bankowego. Pozwoliłoby to na uzyskanie efektów skali.

Słowa kluczowe

Ekstrakcja danych, pozyskiwanie danych, źródła danych, web scraping, web crawling

Wstęp

Zjawisko globalizacji w połączeniu z rozwojem Internetu przyczyniło się do powstania nowego rodzaju gospodarki, w której dane i informacje stanowią m.in. czynnik produkcji, zasób, produkt, a przedsiębiorstwa wykorzystują zdobyte informacje do budowy przewagi konkurencyjnej (Abramowicz, 2008)(Oleński, 2001). Rozwój technologii informacyjnych przyczynił się również do istotnych zmian modeli biznesowych opartych na zarządzaniu zbiorami danych oraz przetwarzaniu informacji, np. model infomediary (polegający na gromadzeniu oraz dostarczaniu klientom wyspecjalizowanych zbiorów danych) lub dostawcy treści (koncentrujący się na wytwarzaniu nowych danych lub informacji i dostarczaniu ich klientom). Wiele zbiorów danych jest dostępnych w postaci Open Data (tzw. otwarte dane, które charakteryzuje możliwość dowolnego, bezpłatnego wykorzystania). Inne udostępniane są na komercyjnych zasadach. Ponadto, możliwe jest pozyskiwanie danych publikowanych w formie tekstowej na stronach internetowych – taki proces nazywany jest z języka angielskiego *web scraping* (dosłownie: zdrapywanie danych).

Web scraping pozwala na zautomatyzowanie procesu pozyskiwania danych z witryn internetowych. Ręczne pobieranie danych polega na nawigowaniu po witrynie przez człowieka i kopiowaniu określonych fragmentów tekstów — taki proces jest czasochłonny i kosztowny, a jego możliwości są ograniczone, bowiem człowiek, w przeciwieństwie do narzędzi *web scrapingowych*, nie jest w stanie przetwarzać wielu stron jednocześnie, zachowując przy tym odpowiednie tempo pracy.

Kolejnym krokiem jest przekształcenie danych w informację, tj. nadanie im kontekstu, który pozwoli na sformułowanie wniosków. Informacje dają przedsiębiorstwom pełniejszy obraz rynku i potrzeb klientów, pozwalają podejmować trafniejsze decyzje, a w konsekwencji budować zasoby wiedzy. Chęć oszczędności czasu, automatyzacji ludzkiej pracy i zwiększenia dostępu do danych przyczyniły się do powstania pierwszych narzędzi do ekstrakcji danych internetowych. *Web scraping*, jako technika, służy do automatyzacji pozyskiwania treści z Internetu. Scraper znajduje określone elementy na stronie i zachowuje je w bazie danych bądź w uprzednio zdefiniowanych formatach plików, jak np.: csv czy xls. To, co dziś wymagałoby ogromu pracy człowieka, a przede wszystkim — zajęłoby dużo czasu, możliwe jest do wykonania automatycznie przez dedykowane narzędzia. Pomimo tego, że *web scraping* nie jest nową technologią, scrapery nigdy wcześniej nie były tak powszechnie używane, jak dzisiaj. Pozyskane dzięki nim dane, przekształcone następnie w informacje, mogą stanowić o przewadze konkurencyjnej danej firmy. Przekształcenie danych w informację, tj. nadanie im kontekstu, pozwala na sformułowanie wniosków. Informacje dają przedsiębiorstwom pełniejszy obraz rynku i potrzeb klientów. Dane uznawane są za jeden z najcenniejszych zasobów, jakie może posiadać firma (Wiliams, 2018).

Pozyskiwanie danych z Internetu związane jest również z popularnym w ostatnich latach pojęciem Big Data (Bhushan, i Kumar, 2012). Big Data oznacza zbiory danych o zróżnicowanej i złożonej strukturze, dużym wolumenie oraz szybkim tempie przyrostu nowych danych. Analiza Big Data (Jasim i in., 2015) pozwala na odkrywanie ukrytych wzorców oraz korelacji, a jej wyniki mogą być przydatną informacją dla firm lub organizacji, gdyż mogą pomóc w uzyskaniu przewagi konkurencyjnej. Big Data jest znaczącym trendem technologicznym i ma potencjał do radykalnej zmiany sposobu, w jaki organizacje wykorzystują informacje, tak aby poprawić jakość obsługi klientów i przekształcić swoje modele biznesowe. Wiele danych zaliczanych do Big Data pochodzi ze strumieni generowanych przez urządzenia i mierniki, a niektóre można pozyskać z Internetu (np. dane z mediów społecznościowych). W przypadku sektora bankowego, ze względu na szczególną rolę, jaką pełni on w gospodarce oraz rolę banków jako instytucji zaufania publicznego, wyjątkowo istotne jest szybkie identyfikowanie informacji, mogących w jakikolwiek sposób na niego wpływać. Przykładem może być informacja opublikowana w mediach społecznościowych, nawołująca do wypłacania gotówki z banków.

Celem raportu jest przedstawienie przypadków użycia źródeł danych dostępnych w Internecie do wsparcia procesów biznesowych, marketingowych oraz związanych z cyberbezpieczeństwem w sektorze bankowym. Raport omawia podstawowe zagadnienia związane z technicznym pozyskiwaniem danych (np. opis wybranych narzędzi), jak również ujęcie ekonomiczne (np. możliwe do uzyskania efekty skali, szacunkowe koszty wykorzystania wybranych źródeł) i prawne (bieżące rozwiązania legislacyjne, licencje źródeł danych).

Struktura raportu jest następująca. Rozdział 2 zawiera przegląd potrzeb informacyjnych manifestowanych przez podmioty sektora bankowego. Rozdział 3 przedstawia listę przykładowych źródeł danych, wraz z ich klasyfikacją oraz odwzorowaniem na zidentyfikowane potrzeby informacyjne. Rozdział 4 poświęcony jest narzędziom służącym do automatycznego pozyskiwania danych ze źródeł internetowych. Omówiono w nim również ograniczenia prawne i technologiczne takich procesów. Rozdział 5 zawiera przykładowe przypadki użycia wybranych narzędzi, ale różnych typów źródeł (stron w języku HTML, API oraz danych udostępnionych w plikach). Rozdział 6 podsumowuje raport i przedstawia rekomendacje związane z procesami automatycznego pozyskiwania danych z Internetu dla sektora bankowego.

Potrzeby informacyjne sektora bankowego

2.1 Potrzeby informacyjne

Sektor bankowy mierzy się z wieloma wyzwaniami. Dotyczy to zarówno rynku krajowego, jak i międzynarodowego. Jest to branża o wysokim poziomie wewnętrznej konkurencji, wymagająca ciągłego dostosowywania się do zmieniających się warunków funkcjonowania oraz zmian otoczenia. Istnieje szereg czynników wpływających na sposób funkcjonowania banku oraz wybór modelu biznesowego, które można podzielić na następujące kategorie (Klimontowicz i Pyka, 2016):

- ekonomiczne: poziom rozwoju kraju, koniunktura gospodarcza, sytuacja na globalnych rynkach finansowych, zmiany cen na rynkach;
- polityczno-prawne: polityka monetarna, polityka fiskalna, regulacje prawne, rekomendacje, dyrektywy unijne;
- technologiczne: rozwój technologii telekomunikacyjnych, rozwój technologii mobilnych, rozwój technologii wspomagania biznesu;
- społeczne: zmiany demograficzne, zmiany potrzeb i oczekiwań klientów, zmiany stylu życia.

W celu umocnienia pozycji konkurencyjnej należy stale poszerzać swoją ofertę, a także podnosić jakość usług. Analiza opinii klientów jest wyzwaniem samym w sobie, ponieważ są one wyrażane za pomocą różnych kanałów komunikacyjnych (np. na portalach internetowych) oraz na różne sposoby (np. jako komentarz, reakcja, przyznanie gwiazdek). Opinie mogą dotyczyć zarówno oferty banku, jak i jakości obsługi. Jedna opinia może zapoczątkować niekorzystną dla banku dyskusję, na przykład na portalu społecznościowym. Istnieje również asymetria pomiędzy pozytywnymi oraz negatywnymi opiniami zamieszczanymi w Internecie — użytkownicy mają tendencję do publikowania negatywnych opinii, jeżeli nie są zadowoleni z usługi, ale mniej chętniej wystawiają pozytywne komentarze, jeżeli usługa jest zgodna z ich oczekiwaniami. Opinie formułowane przez użytkowników Internetu są wyrazem subiektywnych odczuć, co może powodować problemy w ich kwantyfikowaniu. Powinny być one również poddane ocenie pod względem wiarygodności (choć w niektórych wypadkach nawet niewiarygodne opinie mogą być podstawą do podjęcia decyzji, np. konieczność reakcji na tzw. fake news).

Kolejnym aspektem jest bieżący monitoring oferty konkurencji. Przy czym konkurencja jest tu rozumiana bardzo szeroko. Po pierwsze są to inne banki oraz instytucje finansowe, które oferują produkty i usługi tego samego przeznaczenia. W związku z rozwojem technologii oraz powstawaniem nowych usług, konkurentem może stać się również podmiot, który wcześniej nie istniał na rynku finansowym. Mogą to być zarówno duże firmy technologiczne, takie jak Google czy Apple, nowe podmioty, np. fintechy, jak również pośrednicy oraz serwisy aukcyjne (np. Amazon, Alibaba, Allegro). Nie tylko firmy technologiczne rozwijają zakres swoich usług. Banki coraz częściej wkraczają na rynek ubezpieczeń, oferując produkty ubezpieczeniowe w ramach bancassurance.

Rozwój technologii sprawia, iż ciągłe zmiany są koniecznością w celu zaoferowania klientom tego, czego oczekują. Przykładem mogą być płatności zbliżeniowe lub za pomocą telefonu: jeszcze niedawno postrzegane jako innowacja, teraz są już standardem. Chcąc utrzymać swoich klientów, banki są zmuszone śledzić nowinki technologiczne w celu analizy ich potencjalnego wykorzystania na wybranych rynkach.

Jednym z najważniejszych wyzwań w sektorze bankowym jest bezpieczeństwo. Zabezpieczenia fizyczne nie są w stanie zapewnić bezpieczeństwa środków zgromadzonych na rachunkach klientów. Coraz częściej banki muszą zmierzyć się cyberatakami, co powoduje rosnące znaczenie rozwiązań z obszaru cyberbezpieczeństwa. Instytucje finansowe stosują różne formy zabezpieczeń, lecz czasami są one niewystarczające, aby powstrzymać hakerów. Stąd bierze się potrzeba monitorowania doniesień o naruszeniach bezpieczeństwa banków, w celu jak najszybszej reakcji. Badania pokazują, że to właśnie obszar finansów i bankowości jest najbardziej narażony na cyberataki (PICB, 2019). Co więcej, liczba incydentów bezpieczeństwa w sektorze finansowym z roku na rok zwiększa się – w 2019 roku banki zanotowały średnio 1750 nadużyć (EY i ZPF, 2019).

Kolejnym wyzwaniem jest zmieniające się otoczenie prawne. Powstające nowe przepisy oraz regulacje, a także negatywne dla banków decyzje UOKiK sprawiają, że zarządzanie bankiem staje się coraz trudniejsze. Ponadto sposoby uwzględnienia tych zmian są różne, więc warto śledzić jak z daną regulacją radzą sobie inne podmioty (np. z regulacją PSDII).

Otoczenie banku to nie tylko konkurenci, klienci oraz regulacje. To również inne branże, których rozwój ma znaczący wpływ na sektor bankowy. Wśród takich branż warto podkreślić istotność branży budowlanej, która ma wpływ na wartość zaciągniętych kredytów hipotecznych oraz inwestycyjnych. Z pozoru niezwiązane z bankowością wydarzenia mogą mieć znaczący wpływ na zachowania klientów. Przykładem mogą być katastrofy oraz klęski żywiołowe (takie jak powodzie, trzęsienia ziemi, tsunami), które sprawiają, że ludzie tracą oszczędności życia i częściej decydują się na kredyty konsumpcyjne lub pożyczki hipoteczne. Z kolei poważne epidemie (np. koronawirus) zaburzają funkcjonowanie gospodarki, wpływając na wzrost cen, a co za tym idzie również

kosztów utrzymania. Mogą również powodować problemy z płynnością, straty finansowe przedsiębiorstw oraz pogorszenie relacji z inwestorami.

Monitorowanie wszystkich istotnych dla banków informacji jest zadaniem żmudnym. Częściowym rozwiązaniem jest automatyczne ich pobieranie z wielu źródeł internetowych. Do niedawna banki korzystały głównie z danych zgromadzonych przez własne systemy transakcyjne i hurtownie danych, uzupełniane przez dane zakupione od zewnętrznych dostawców (Ramotowski, 2016). Wskazuje się jednak na potrzebę rozszerzenia analizowanych źródeł danych np. o dane z powiązanych sektorów (ubezpieczeniowego, energetycznego, rynku nieruchomości), mediów społecznościowych czy stron instytucji rządowych. Umożliwiłoby to bardziej kompleksową analizę, opartą na danych pozyskiwanych w czasie rzeczywistym. Ręczne pozyskanie danych z tak wielu zróżnicowanych źródeł byłoby nieefektywne kosztowo i czasowo, stąd potrzeba wykorzystania automatycznych narzędzi.

Raport przeprowadzony na zlecenie Thomson Reuters (Greenwich Associates, 2018) wskazuje, że automatyczne pobieranie danych z sieci (*web scraping*) jest najczęściej stosowanym źródłem danych na potrzeby badania rynku pod kątem inwestycji.

Podsumowując, wśród najważniejszych potrzeb informacyjnych banków, zidentyfikowanych na podstawie analizy literatury, i które można zaspokoić (przynajmniej w pewnym stopniu) wykorzystując informacje dostępne w Internecie wyróżnia się:

- opinie klientów na temat oferty własnej banku;
- opinie klientów na temat jakości obsługi klienta;
- monitorowanie ryzyka reputacyjnego na podstawie dyskusji w Internecie;
- bieżąca oferta konkurencji (w tym również podmiotów inne niż banki, np. firmy technologiczne, takie jak Amazon i Alibaba);
- monitorowanie bieżącej konkurencji ze strony instytucji ubezpieczeniowych lub innych podmiotów typu bancassurance;
- trendy technologiczne w sferze finansów (np. płatność telefonem, kryptowaluty) oraz ich odbiór przez potencjalnych klientów;
- monitorowanie informacji o naruszeniach bezpieczeństwa banków i ich klientów;
- nowe regulacje i przepisy prawne;
- decyzje UOKiK w sprawie praktyk oraz produktów bankowych;
- trendy na rynku nieruchomości;
- dane makroekonomiczne;
- informacje o klęskach żywiołowych, katastrofach oraz epidemiach.

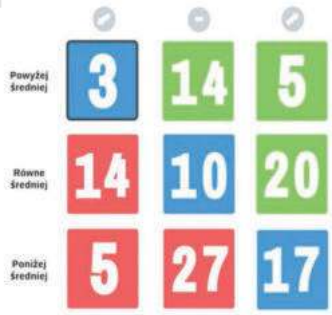
2.2 Propozycje struktury raportów

Aby możliwe było zaspokojenie potrzeb informacyjnych zidentyfikowanych w poprzednim rozdziale, niezbędna jest ich pogłębiona analiza, która powinna być przeprowadzana dla konkretnego użytkownika/podmiotu i związana z określonymi problemami biznesowymi.

Zaspokojenie potrzeb informacyjnych wymaga określenia dwóch składowych: źródeł danych lub informacji, które na te potrzeby odpowiedzą, oraz sposobu wizualizacji wyników użytkownikowi. Wyniki sektora bankowego mogą być zaprezentowane w różnej formie, zależnej od typu danych czy informacji, jakie ze sobą niosą. Pierwszym czynnikiem, branym pod uwagę przy ocenie kondycji banku czy instytucji finansowej, są wskaźniki, których wartości odzwierciedlają osiągnięte wyniki. Ponadto, w oparciu o analizę wskaźnikową, osoby zarządzające mogą odpowiednio ukierunkować działalność podmiotu oraz zidentyfikować obszary nieprawidłowo zarządzane, które mogą mieć niekorzystny wpływ na funkcjonowanie banku i osiągane w przyszłości rezultaty.

Wskaźniki dzielą się na trzy główne grupy:

- Finansowe — przedstawiają wielkości wynikające ze sprawozdań finansowych. Podstawowym źródłem informacji są: bilans oraz rachunek zysków i strat, które jako elementy sprawozdania finansowego zawierają najwięcej istotnych informacji potrzebnych do analizy sytuacji finansowej.
- Ryzyka — lista składa się z około 350 wskaźników opracowanych przez EBA (European Banking Authority). Dotyczą one między innymi ryzyka związanego z brakiem płynności, niską jakością aktywów czy brakiem wypłacalności.
- Operacyjne — wskaźniki opracowane przez OPSDOG2 (firma zajmująca się między innymi standaryzacją danych oraz przeprowadzaniem analiz i benchmarków), w skład których wchodzi 50 wskaźników, odnoszących się do danych wrażliwych.

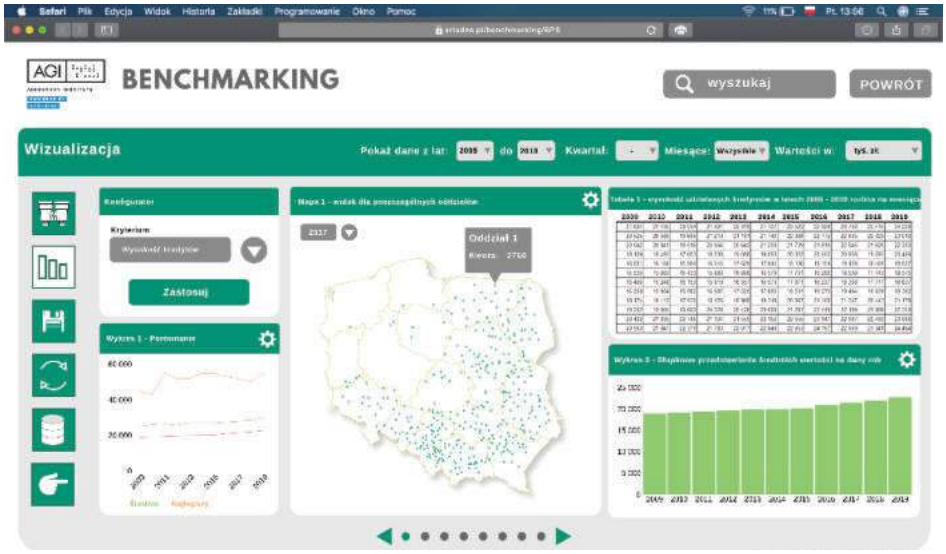


Rysunek 1. Overall Results Matrix.
Źródło: opracowanie własne

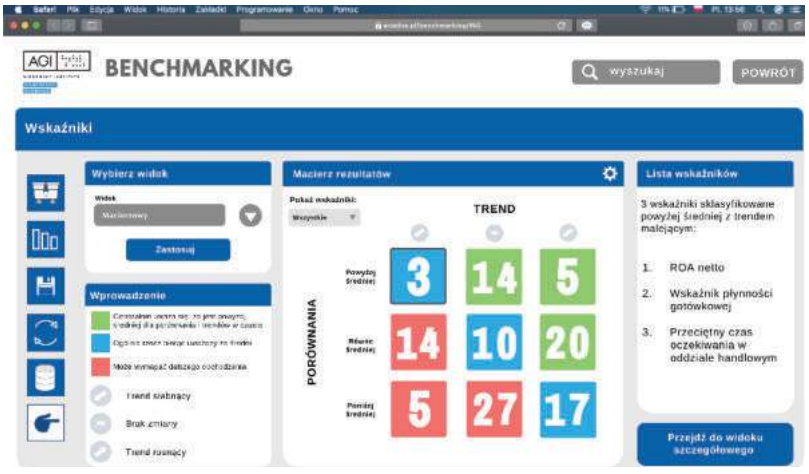
Interesującą formą przedstawienia kondycji banku na tle innych podmiotów jest Overall Results Matrix, czyli macierz prezentująca wyniki z uwzględnieniem trendu zachodzących zmian, która została zaprezentowana na rysunku 1. oraz omówiona w dalszej części rozdziału.

W celu prezentacji pomysłu zostały stworzone mockupy zaprezentowane na rysunkach 2. i 3. Przedstawiają one przykładową analizę porównawczą banków i ich konkurencyjności, przeprowadzoną przy użyciu danych otwartych.

Mockup zaprezentowany na rysunku 2. przedstawia porównanie wybranego banku pod względem wysokości udzielonych kredytów. Zestawienie banku ze średnimi wynikami sektora oraz z liderem dobrze obrazuje pozycję instytucji. Na mapie są przedstawione wszystkie oddziały zlokalizowane w Polsce. Tabela z prawej strony obrazuje



Rysunek 2. Przykładowy mockup.
Źródło: opracowanie własne



Rysunek 3. Przykładowy mockup.
Źródło: opracowanie własne

wysokość udzielonych kredytów w poszczególnych miesiącach na przestrzeni kilku lat, a wykres poniżej przedstawia łączną wartość kredytów udzielonych przez bank w kolejnych latach.

Mockup zaprezentowany na rysunku 3. pokazuje macierzowe porównanie kilku instytucji bankowych. Po lewej stronie znajduje się legenda do odczytu wyniku z macierzy rezultatów znajdującej się pośrodku. W wierszach od góry zaprezentowane są wyniki powyżej, równe i poniżej średniej wszystkich banków. W kolumnach natomiast jest zaprezentowany trend, od lewej malejący, stały i rosnący. Wynika z tego, że najkorzystniejsze wyniki zaznaczone są na zielono, a najmniej korzystne na czerwono. Okno znajdujące się po prawej stronie zawiera listę z wybranymi wskaźnikami, w przykładzie są one powyżej średniej z trendem malejącym.

ROZDZIAŁ III

PROGRAM ANALITYCZNO BADAWCZY

Klasyfikacja źródeł danych

W rozdziale 2 zostały opisane potrzeby informacyjne banków. Większość z nich można zaspokoić danymi pochodzącymi z Internetu. Niniejszy rozdział przedstawia przegląd przykładowych źródeł internetowych, wraz z ich wstępną analizą.

Rozróżniamy dwie podstawowe klasy źródeł w Internecie: Shallow Web i Deep Web (Abramowicz, 2008).

Shallow Web (Surface Web, Visible Web, Internet płytki) to źródła dostępne dla standardowych wyszukiwarek internetowych, których treści zostają zaindeksowane. Do tej kategorii należą np. amazon.com, empik.com, olx.pl, wikipedia.org, publicznie dostępne blogi, większość stron statycznych (Handschuh, Staab, 2003). W przypadku Shallow Web, scraper nawiguje pomiędzy stronami i pobiera dane z formatu html wyświetlanego przez przeglądarkę. Przeciwieństwem Shallow Web jest Deep Web (Invisible Web, Internet głęboki). Zawartość tych źródeł nie jest indeksowana przez wyszukiwarki internetowe. Powodem tego są między innymi:

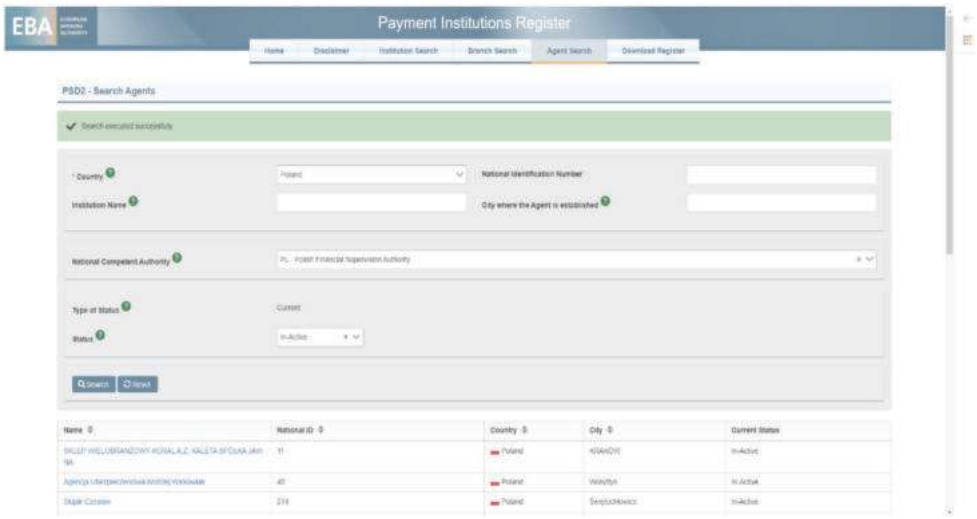
- wymagane jest uwierzytelnienie,
- strony są generowane dynamicznie,
- witryny są napisane w innych językach niż HTML.

Oznacza to, że dostęp dla robotów internetowych do treści takich jak: bankowość internetowa, poczta internetowa, dane rządowe, płatne bazy danych czy strony generowane dynamicznie (np. katalogi biblioteczne) jest utrudniony, ponieważ wymaga na przykład zalogowania się lub wypełnienia formularza wyszukiwania.

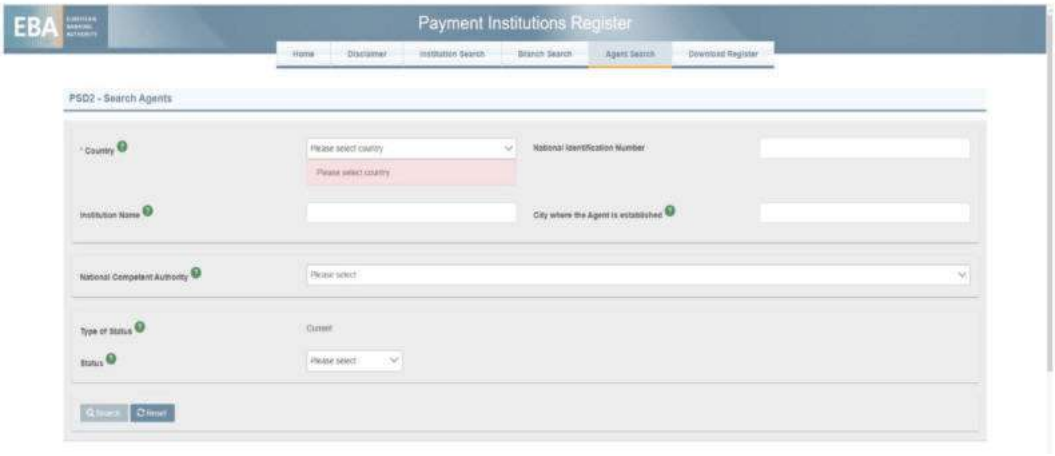
Szacuje się, że zasoby Internetu głębokiego mogą być nawet ok. 400 razy większe niż Internetu płytkiego (Clarke, 2018). Bez posiadania odpowiednich danych (np. login, hasło) nie jest możliwe scrapowanie z tej części Internetu. W skład Internetu głębokiego wchodzi również Dark Web (DarkNet). Ta podstrefa charakteryzuje się wysoką anonimowością, co powoduje, że często jest wykorzystywana przez przestępców. Dostęp do niej można uzyskać tylko za pomocą specjalnego oprogramowania (np. TOR), które jest wykorzystywane ze względu na możliwość zapewnienia użytkownikowi anonimowości. W tym przypadku ekstrakcja danych jest możliwa, lecz wymaga bardziej skomplikowanych działań (np. łączenia z serwerem pośredniczącym). Ze względu na charakter Dark Web należy mieć na uwadze, że powyższe działanie może wzbudzać zainteresowanie policji (Fu, Abbasi, Chen, 2010).

Zidentyfikowane źródła internetowe, które mogą być potencjalnie interesujące dla sektora bankowego, to m.in.:

- rankingi (zarówno podmiotów, czyli banków, jak i wybranych ofert, np. kont osobistych),
- fora internetowe (np. branżowe fora bankowe lub indywidualne fora banków),
- portale branżowe informujące o nowościach ze świata finansów (w tym trendy technologiczne, aktualna kondycja rynku nieruchomości),
- media społecznościowe,
- strony domowe banków,
- strony instytucji (np. KNF).



Rysunek 4. Brak dostępu do danych przed wypełnieniem formularza.
Źródło: (European Banking Authority, 2020)



Rysunek 5. Dostęp do danych po wypełnieniu formularza.
Źródło: (European Banking Authority 2020)

Inne kryterium dotyczy pochodzenia danych. Wśród wymienionych grup można wyróżnić takie, w których za powstanie oraz utrzymanie źródeł odpowiedzialne są:

- osoby trzecie, tj. niebędące pracownikami instytucji finansowych (rankingi tworzone przez dziennikarzy, portale branżowe),
- banki i pracownicy banków (strony domowe banków, indywidualne fora bankowe),
- niezależne instytucje,
- użytkownicy indywidualni, w tym np. klienci banków (fora branżowe, media społecznościowe),
- podmioty i instytucje państwowe i samorządowe (np. KNF, EBA, ZBP).

Dane w niektórych źródłach mogą być tworzone przez różnych autorów. Przykładem takiego źródła są rankingi, które mogą być tworzone zarówno przez osoby trzecie (tj. niezwiązane z podmiotami, które w takim rankingu się znajdują), konkurencję, jak i pracowników banków. Różne potrzeby informacyjne mogą wymuszać korzystanie ze źródeł danych określonego pochodzenia, np. rankingi tworzone przez konkurencję nie będą dobrą przesłanką do badania opinii klientów o ofercie własnej banku, ale mogą być pomocne przy monitorowaniu oferty konkurencji lub trendów technologicznych.

Kolejnym elementem analizy źródeł jest ich stopień formalizacji. Można wyodrębnić dwa podstawowe typy źródeł: zawierające dokumenty ustrukturyzowane oraz nieustrukturyzowane. Dokumenty ustrukturyzowane charakteryzują się, jak wskazuje nazwa, wyraźną strukturą, w której poszczególne elementy (np. tekst komentarza, autor, data publikacji, lokalizacja, itp.) są wyraźnie wyodrębnione i mogą być przetwarzane w sposób automatyczny przez narzędzia do pozyskiwania danych. Źródła ustrukturyzowane to również takie, które gromadzą dane pozyskiwane przez formularze, w których poszczególne pola mają jasno określony format. Źródła ustrukturyzowane to np. katalogi biblioteczne lub rejestry przedsiębiorstw. Dokumenty nieustrukturyzowane nie mają określonej struktury, często są wręcz w całości pisane w języku naturalnym. Dokumenty nieustrukturyzowane są znacznie trudniejsze do przetworzenia przez maszyny, gdyż konieczne jest zastosowanie technik przetwarzania języka naturalnego (NLP) w celu zidentyfikowania poszczególnych elementów takiego dokumentu (Wang i in. 2015) (Platt i Soens, 2018). Przykładem dokumentu nieustrukturyzowanego może być dokument tekstowy zawierający opinię klienta na temat określonej usługi, w postaci ciągłego tekstu. Dokumenty o bardzo prostej strukturze, w których

Rysunek 6. Fragment dokumentu w języku HTML. Nazwy klas (class) przypisane do poszczególnych znaczników „div" pozwalają na określenie struktury pliku.
Źródło: opracowanie własne.

większość treści ma postać tekstów w języku naturalnym, ale możliwe jest wyodrębnienie w nich pewnych stałych elementów, nazywa się półstrukturyzowanymi. Są to np. strony internetowe z artykułami. Znaczniki wykorzystane w językach znaczników wspomagają zidentyfikowanie takiej strukturalizacji. Języki znaczników to np. HTML (rysunek 6) i XML. Możliwy jest również opis zasobów przy pomocy technologii semantycznych, które pozwalają na wnioskowanie na zbiorze danych przez maszyny, np. o istnieniu relacji pomiędzy określonymi podmiotami na rynku (Lewańska, 2016). Takich źródeł dedykowanych dla obszaru bankowości i cyberbezpieczeństwa jest jednak niewiele.

Poniżej przedstawiono grupy źródeł, które zostały wybrane jako przykłady zaspokajające potrzeby informacyjne. Nie są to wszystkie źródła, które mogą okazać się użyteczne dla sektora bankowego. Stworzenie pełnej listy nie jest możliwe ze względu na dużą zmienność źródeł internetowych.

3.1 Rankingi

Internetowe rankingi są łatwym i szybkim sposobem na porównanie dostępnych na rynku ofert. Ich celem jest syntetyczne przedstawienie ofert różnych podmiotów, w celu usprawnienia procesu decyzyjnego klienta. Rankingi mogą stanowić również analizę porównawczą podmiotów działających w określonym sektorze.

Ranking to uporządkowana pod względem zdefiniowanych kryteriów lista obiektów. Kryteria, które są podstawą tworzenia rankingu, są dobierane zazwyczaj metodą ekspercką. Ponadto, część kryteriów przyjmuje wartości jakościowe, których ocena jest subiektywna. Kryteria mogą być oceniane przy pomocy różnych typów i zakresów wartości. Powoduje to, że rankingi tworzone przez różne podmioty trudno jest ze sobą porównać. Niejednokrotnie zdarza się, że rankingi tych samych obiektów, nawet zawierające te same kryteria, ale tworzone przez różne podmioty, różnią się.

W obszarze bankowości rankingi zwykle zawierają porównanie banków (jako całości) lub poszczególnych typów oferowanych produktów (np. porównanie kont osobistych dla studentów).

Trudnością przy wykorzystywaniu rankingów jest ocena ich wiarygodności. Niektóre rankingi to tzw. materiały sponsorowane, które często faworyzują określony podmiot lub jego ofertę. Należy również zwrócić uwagę na sposób pozyskiwania danych do rankingów. Mogą to być dane zbierane ręcznie przez autora rankingu (wówczas istnieje duże ryzyko, że taki ranking nie obejmuje wszystkich obiektów), mogą być budowane na podstawie publicznie dostępnych danych, agregowanych przez instytucje sektora lub na podstawie danych deklarowanych przez podmioty, które dobrowolnie chcą zostać poddane analizie.

Problem rankingowania obiektów oraz porównywania rankingów między sobą jest przedmiotem badań naukowych (Frigo i Kocsis, 2019) (You i Hwang, 2007) (Cuncha, Soares i de Carvalho, 2018) (Webber, Moffat i Zobel, 2010). Jednakże, analizy w tym zakresie są możliwe do przeprowadzenia po uprzednim pozyskaniu danych ze źródeł (tj. analizowanych rankingów) i sprowadzeniu ich do postaci ustrukturyzowanej, co umożliwiają narzędzia opisane w dalszej części raportu.

Dodatkową trudnością jest fakt, iż rankingi często publikowane są w formie artykułów pojawiających się nieregularnie lub jednorazowo, a także struktura stron przedstawiających rankingi jest zróżnicowana. Konsekwencją tego jest konieczność tworzenia osobnego scrapera dla każdej ze stron (patrz Rozdział 4 — Narzędzia do automatycznego pozyskiwania danych z Internetu).

Pomimo wskazanych wad, rankingi pozwalają w pewnym stopniu na zidentyfikowanie swojej pozycji konkurencyjnej względem innych dostawców produktów i usług finansowych lub poznanie opinii o rozwiązaniach dostępnych na rynku. Ponadto, bank może przeanalizować swoje produkty pod wieloma aspektami, na które klienci najbardziej zwracają uwagę oraz które stanowią kryteria w rankingach. Może to być: cena, jakość, dostosowanie oferty do indywidualnych potrzeb itp. Wyniki w rankingach mogą również wskazywać, jaką reputacją cieszy się bank i jak jest postrzegany przez klientów. Kolejnym ważnym aspektem jest uwzględnianie przez rankingi rozwiązań alternatywnych dla banków, takich jak oferty firm technologicznych oraz bancassurance.

Przykładowe strony z rankingami, tworzonymi przez osoby/podmioty trzecie, znajdują się w Tabeli 1.

Tabela 1. Przykładowe strony z rankingami.

Nazwa	Potrzeby informacyjne
eBroker (https://www.ebroker.pl/instytucje/banki)	• opinie klientów na temat oferty własnej banku; • opinie klientów na temat jakości obsługi klienta; • monitorowanie ryzyka reputacyjnego na podstawie dyskusji w Internecie; • bieżąca oferta konkurencji;
Tanie-Konto (https://www.tanie-konto.pl/)	• opinie klientów na temat oferty własnej banku; • opinie klientów na temat jakości obsługi klienta; • monitorowanie ryzyka reputacyjnego na podstawie dyskusji w Internecie; • bieżąca oferta konkurencji;
Złoty Bankier (https://www.bankier.pl/zlotybankier/)	• opinie klientów na temat oferty własnej banku; • opinie klientów na temat jakości obsługi klienta;
Ranking kont osobistych (https://www.rankingkontosobistych.pl/konta-osobiste)	• opinie klientów na temat oferty własnej banku; • opinie klientów na temat jakości obsługi klienta; • monitorowanie ryzyka reputacyjnego na podstawie dyskusji w Internecie; • bieżąca oferta konkurencji;
Bankowe-Konta (http://bankowe-konta.info/)	• opinie klientów na temat oferty własnej banku; • opinie klientów na temat jakości obsługi klienta; • monitorowanie ryzyka reputacyjnego na podstawie dyskusji w Internecie; • bieżąca oferta konkurencji

Źródło: opracowanie własne.

3.2 Fora internetowe

Forum to internetowa forma grupy dyskusyjnej, służąca do wymiany informacji i poglądów. Taka forma komunikacji jest nadal używana, aczkolwiek obecnie ustępuje miejsca innym sposobom komunikacji, co można stwierdzić na podstawie malejącej liczby aktywnych użytkowników oraz zakładanych wątków.

Fora zakładane są zarówno na stronach domowych banków, jak i jako osobne witryny – w tym drugim przypadku często trudno jest określić, czy właścicielem forum jest osoba prywatna, czy instytucja.

Wadą forów, z punktu widzenia automatyzacji pozyskania z nich informacji, jest ich różnorodna strukturalizacja, która utrudnia pozyskiwanie informacji według jednego, predefiniowanego schematu, co wymusza tworzenie osobnego scrapera dla każdego forum. Ponadto, choć na forach można wyróżnić pewne strukturalizowane elementy (np. tytuł posta, autor, data publikacji), to same posty są pisane w języku naturalnym, często zawierając błędy językowe, gramatyczne i ortograficzne. Powoduje to trudności w ich dalszym przetwarzaniu – konieczne jest wykorzystywanie narzędzi przetwarzania języka naturalnego (NLP) (Spina i in., 2012) (Wanas i in., 2008) (Wang i in. 2015) (Platt i Soens, 2018).

Bardzo często, pomimo łatwego i szybkiego dostępu do ofert poszczególnych banków i instytucji finansowych, ludzie kierują swoje wątpliwości i pytania za pośrednictwem forów internetowych. Taka sytuacja może wynikać z zaufania do doświadczeń innych osób, z niechęci do samodzielnego przeszukiwania ofert lub też z szybkości założenia nowego wątku oraz wielkości społeczności na forach. Choć niejednokrotnie łatwiej jest uzyskać odpowiedź lub radę od innego użytkownika, niż znaleźć szukaną informację na stronie banku lub w jednym z wielu regulaminów, to należy pamiętać, że odpowiedzi udzielane na forach nie są w żaden sposób weryfikowane.

Innym często spotykanym przypadkiem użycia forum jest dyskusja użytkowników o podejrzeniach cyberataków. Klienci niejednokrotnie przed zgłoszeniem anomalii szukają potwierdzenia, że ich spostrzeżenia co do wystąpienia nietypowych zdarzeń

lub zmian są słuszne. Ponadto, użytkownicy dokładnie opisują napotkane odstępstwa od norm, co jest istotne z punktu widzenia instytucji finansowych, gdyż pozwala to na zweryfikowanie domniemanego cyberataku, jego skali i oznak wystąpienia u konkretnego użytkownika. Wynika z tego, że fora mogą być źródłem istotnych informacji o naruszeniu bezpieczeństwa cybernetycznego banków. Fora pełnią zatem funkcję informacyjną, dostarczając informacji na temat ofert banków i instytucji finansowych wraz z ocenami i opiniami osób, które już z tych usług lub produktów skorzystały.

Na podstawie wydźwięku komentarzy użytkowników można ocenić, czy ryzyko reputacyjne określonej instytucji ulega zmianom pozytywnym czy negatywnym. Użytkownicy forum niejednokrotnie wyrażają opinię na temat jakości obsługi klientów w poszczególnych bankach lub nawet oddziałach. Pomimo tego, nie można stwierdzić, że przedstawia to rzeczywisty odbiór danej instytucji przez klientów, gdyż często opinie na forach wyrażają wyłącznie klienci niezadowoleni z usług, a usatysfakcjonowani obsługą czy jakością usług nie zabierają głosu na forach. Mimo powyższych wad, bieżące monitorowanie forów internetowych pozwala na obserwowanie opinii klientów, szybką identyfikację pojawiających się słabych punktów działalności instytucji finansowej i reakcję na nie. Fora pozwalają również na zapoznanie się z konkurencją spoza sektora stricte bankowego, gdzie dominującą pozycję zajmują firmy technologiczne, gdyż użytkownicy wymieniają te instytucje jako alternatywy dla banków.

Komentarze na forach są pisane w języku naturalnym, zatem do oceny wyrażanych przez nich opinii konieczne jest zastosowanie metod analizy sentymentu. Analiza sentymentu (nazywana też analizą wydźwięku) jest wymagającym obszarem, jednak prowadzone są intensywne badania w tym zakresie, również na potrzeby sektora bankowego (Yu, Pan, Zhou, 2019) (Esichaikul i Phumdontree, 2018). Analiza sentymentu jest kolejnym etapem pracy z danych pozyskanymi dzięki web scrappingowi.

Przykładowe strony z forami znajdują się w Tabeli 2.

Tabela 2. Przykładowe strony z forami.

Nazwa	Potrzeby informacyjne
Forum-Bankowe (http://www.forum-bankowe.pl/)	• opinie klientów na temat oferty własnej banku; • bieżąca oferta konkurencji;
Banki-Opinie (https://www.banki-opinie.info/)	• opinie klientów na temat oferty własnej banku; • opinie klientów na temat jakości obsługi klienta; • monitorowanie ryzyka reputacyjnego na podstawie dyskusji w Internecie; • bieżąca oferta konkurencji

Źródło: opracowanie własne.

3.3 Portale branżowe

Portale branżowe mają na celu publikację newsów ze świata finansów oraz informacji o sytuacji na rynku nieruchomości i trendach technologicznych, ze szczególnym uwzględnieniem ich wpływu na sektor finansowy. Wiele z takich branżowych portali, chcąc podążać za nowymi technologiami i rozwiązaniami, skupia się również na publikowaniu informacji związanych z cyberbezpieczeństwem, o zmianach w przepisach prawnych wraz z określeniem ich istotności dla przedsiębiorstw czy wybranych sektorów rynkowych, jak również powiadomienia o klęskach żywiołowych, katastrofach oraz epidemiach.

Portale branżowe publikują informacje o podobnej tematyce, można jednak wyznaczyć kilka znaczących, z punktu widzenia pobierania danych, aspektów, które je różnicują.

Po pierwsze, portale można klasyfikować pod względem struktury technicznej stron. Pomimo faktu, iż strony pozornie wydają się być zbliżone wyglądem, to ich struktura jest zupełnie inna. Pobieranie z nich danych wymagałoby więc dostosowania scraperów i opracowania reguł ekstrakcji pod konkretne źródło. Ma to swoje uzasadnienie w działaniu scraperów, które zostało opisane w dalszej części raportu (patrz Rozdział 4 — Narzędzia do automatycznego pozyskiwania danych z Internetu).

Po drugie, format pobieranych danych może okazać się trudny do dalszego przetwarzania. Pozyskiwanie informacji z opisywanych portali wymaga bowiem

pobierania całych artykułów, czyli ciągu tekstu, które następnie muszą zostać poddane dalszej analizie, np. metodami przetwarzania języka naturalnego (NLP). Niekiedy portale kategoryzują publikacje, tak aby ułatwić ich wyszukiwanie, co może być wykorzystywane również przez roboty internetowe do identyfikowania treści określonego typu. Przykładem takiej strony jest ObserwatorFinansowy.pl, na której każdy artykuł został oznaczony odpowiednimi tagami, pomocnymi przy selekcji treści do pobrania.

Treści zamieszczone na portalach są obwarowane różnymi zapisami licencyjnymi lub innymi, które określają, w jaki sposób (i czy w ogóle) można je przetwarzać. Przykładowo, ObserwatorFinansowy działa na otwartej licencji (otwarta licencja zezwala na korzystanie z udostępnianych zbiorów pod warunkiem uznania autorstwa, czyli oznaczania źródła pochodzenia), jednak wiele innych portali nie udziela informacji o możliwości pobierania zasobów. W takich okolicznościach, przed rozpoczęciem scrapowania należy skontaktować się z odpowiednim działem danego portalu z prośbą o ustosunkowanie się do tej kwestii (patrz rozdział 4.2 Uwarunkowanie prawne, etyczne oraz techniczne).

Przykładowe portale branżowe znajdują się w Tabeli 3.

Tabela 3. Przykładowe portale branżowe.

Nazwa	Potrzeby informacyjne
Obserwator Finansowy (https://www.obserwatorfinansowy.pl/)	• trendy technologiczne w sferze finansów oraz ich odbiór przez potencjalnych klientów; • monitorowanie informacji o naruszeniach bezpieczeństwa banków i ich klientów; • nowe regulacje i przepisy prawne; • trendy na rynku nieruchomości; • informacje o klęskach żywiołowych, katastrofach oraz epidemiach.
PwC (https://www.pwc.pl/pl/branze/bankowosc-ubezpieczenia.html)	• trendy technologiczne w sferze finansów oraz ich odbiór przez potencjalnych klientów; • trendy na rynku nieruchomości;
aleBank.pl (https://alebank.pl/)	• trendy technologiczne w sferze finansów oraz ich odbiór przez potencjalnych klientów; • nowe regulacje i przepisy prawne; • trendy na rynku nieruchomości;
CyberDefence24 (https://cyberdefence24.pl/biznes-i-finanse/)	• trendy technologiczne w sferze finansów oraz ich odbiór przez potencjalnych klientów; • monitorowanie informacji o naruszeniach bezpieczeństwa banków i ich klientów;
Forbes (https://www.forbes.pl/rynek-nieruchomosci)	• trendy w branży budowlanej

Źródło: opracowanie własne.

3.4 Strony domowe banków

Strona internetowa jest ważnym elementem dla każdej instytucji. Poza kreowaniem wizerunku, umożliwia ona prezentowanie oferty banku oraz za jej pośrednictwem użytkownicy uzyskują dostęp do bankowości internetowej. Ponadto, banki tworzą osobne portale edukacyjno-informacyjne, na których publikują posty tematyczne, analizy lub informacje o nadchodzących wydarzeniach.

Poniżej przedstawiono opis strony banku na przykładzie PKO BP, który oprócz dostępu do bankowości internetowej, publikuje informacje prasowe, oferuje opcję zapisu do newslettera oraz zawiera przekierowania do mediów społecznościowych. W posiadaniu banku są również powiązane z nim portale, tj. Centrum Analiz, Serwis Rynkowy Biura Maklerskiego oraz Bankomania. Centrum Analiz publikuje analizy makroekonomiczne i sektorowe, informacje o stopach procentowych i rynkach zagranicznych. Serwis Rynkowy

Biura Maklerskiego prezentuje analizy giełdowe spółek oraz artykuły edukacyjne o tematyce giełdowej i podatkach. Z kolei portal Bankomania skupia się na przedstawieniu możliwości wykorzystania usług banku PKO BP w sytuacjach z życia codziennego. Informacje publikowane na powyższych portalach są aktualne, a częstotliwość publikacji duża.

Dostępne na stronach banków informacje nie są możliwe do pobrania przez API lub w formie importu z pliku csv czy xlsx, zatem żeby móc z nich korzystać, niezbędne jest wykorzystanie scraperów opisanych w dalszej części. Wyzwaniem jest fakt, że struktura strony każdego banku jest inna, zatem nie ma uniwersalnego schematu przeszukiwania ich.

3.5 Media społecznościowe

Media społecznościowe (social media) przyciągają coraz więcej użytkowników, są więc źródłem danych charakteryzującym się bardzo szybkim przyrostem. Należy zaznaczyć, że ludzie korzystają z portali społecznościowych nie tylko na komputerze, ale również na urządzeniach mobilnych.

Na podstawie zbieranych i przechowywanych przez portale społecznościowe danych, można następnie przeprowadzić analizy, np. dotyczące analizy opinii społecznych na dany temat. Do mechanizmów dostępnych w portalach społecznościowych, które dodatkowo ułatwiają poznanie opinii społecznej należą między innymi: możliwość dynamicznego dodawania komentarzy, reagowanie na posty/komentarze za pomocą emotikonów, udostępnianie postów. Poza informacją tekstową, większość portali społecznościowych umożliwia użytkownikom dodawanie zdjęć. Wraz z rozwojem technologii związanych z Internetem, rozwijały się również portale społecznościowe i zaimplementowały możliwości dodawania GIFów i krótkich video. Obecnie poprzez media społecznościowe można również streamować, tj. udostępniać na żywo obraz wideo i w czasie rzeczywistym otrzymywać komentarze oraz reakcje od użytkowników. Choć elementy graficzne i wideo są trudniejsze do przetworzenia (ze względu na konieczność zastosowania innych technologii, rozmiar plików oraz trudności w identyfikacji obiektów na zdjęciach czy filmach), to ich atrybuty (np. nazwa, znaczniki lokalizacyjne) mogą być wykorzystane w celu badania wydźwięku postów.

Do istotnych elementów związanych z mediami społecznościowymi należą grupy, strony fanowskie i tzw. hasztagi. Wszystkie te mechanizmy ułatwiają użytkownikom udostępnianie atrakcyjnych dla nich treści. Dodatkowo są to narzędzia, które pozwalają podmiotom (np. bankom) w prosty sposób komunikować się z klientem. Za pomocą grupy bądź strony fanowskiej dowolny podmiot może wystosować oświadczenie do określonej grupy odbiorców (np. o nowej ofercie). Mechanizmy zaimplementowane w social media pozwalają klientom przedstawić swoją opinię na temat obserwowanych instytucji i podejmowanych przez nie aktywności. Do analizy w tym zakresie wykorzystuje się techniki analizy sentymentu.

Kluczowe wydają się też narzędzia analityczne, które są dostarczane przez część portali społecznościowych (np. Facebook). Dają one możliwość dokonania podstawowej analizy odbiorców strony fanowskiej bądź wydarzenia. Niestety sporym utrudnieniem jest mocne ograniczenie dostępu do API przez Facebooka. Trend zamykania API przez portale społecznościowe jest od kilku lat bardzo popularny, wzmacniają go ustawy powiązane z ochronią danych osobowych (np. RODO). Przykładem portalu społecznościowego, który wciąż zapewnia dostęp do swojego API, jest Twitter. Dodatkową zaletą Twittera jest jego konstrukcja, oparta na hasztagach, która pozwala na prostszą analizę zależności i sieci społecznościowej niż np. w przypadku Facebooka (tzn. łatwiej jest utworzyć graf zależności). Dodatkowo społeczność Twittera jest bardziej dojrzała, tym samym informacje tam dostępne mogą być znacznie cenniejsze i są częściej wykorzystywane jako newsy (von Nordheim, Boczek, Koppers, 2018).

Przykłady mediów społecznościowych przedstawia Tabela 4.

Tabela 4. Przykładowe media społecznościowe.

Nazwa	Potrzeby informacyjne
Twitter (https://developer.twitter.com/en/docs)	Wszystkie zidentyfikowane, ale informacje są bardzo zróżnicowanej jakości.
Facebook (https://developers.facebook.com/)	
LinkedIn (https://www.linkedin.com/developers/)	

Źródło: opracowanie własne.

3.6 Strony instytucji

Instytucje państwowe i samorządowe są ważnym źródłem danych dla sektora bankowości, ponieważ część potrzeb informacyjnych powinna być spełniona poprzez pobranie zasobów z potwierdzonych źródeł. Instytucje są najpewniejszym źródłem informacji, ponieważ nie są tam wyrażane subiektywne opinie, jak w przypadku wielu innych omówionych grup źródeł, jak również zawierają informacje sformułowane przez ekspertów branżowych. Jest to szczególnie istotne w przypadku takich potrzeb, jak informacje o nowych regulacjach i przepisach prawnych, decyzjach UOKiK w sprawie praktyk oraz produktów bankowych, informacje o klęskach żywiołowych, katastrofach oraz epidemiach.

Pomimo oczywistej korzyści, jaką jest wiarygodność źródeł w tej grupie, wynikająca z faktu, że są to strony prowadzone przez niezależne instytucje, takie jak KNF, EBA, ZBP, należy wskazać na poważną przeszkodę, jeśli chodzi o pozyskiwanie danych i informacji z tych źródeł. Każda instytucja specjalizuje się w jednej określonej dziedzinie (np. EBA zajmuje się rynkiem finansowym UE), a co za tym idzie, jedno źródło może spełnić tylko jedną potrzebę informacyjną. Jest to znaczące utrudnienie, ponieważ wymaga monitorowania wielu źródeł, tak aby uzyskać pełny obraz użytecznych informacji. Z kolei pozyskiwanie informacji z wielu źródeł wiąże się z koniecznością opracowania wielu scraperów, dostosowanych pod strukturę strony konkretnej instytucji.

Przykładowe strony instytucji znajdują się w Tabeli 5.

Tabela 5. Przykładowe strony instytucji.

Nazwa	Potrzeby informacyjne
Urząd Ochrony Konkurencji i Konsumentów (https://www.uokik.gov.pl/)	• decyzje UOKiK w sprawie praktyk oraz produktów bankowych;
Komisja Nadzoru Finansowego (https://www.knf.gov.pl/)	• nowe regulacje i przepisy prawne;
Biuro Informacji Kredytowej (https://www.bik.pl/)	• monitorowanie informacji o naruszeniach bezpieczeństwa banków i ich klientów;
Narodowy Bank Polski (https://www.nbp.pl/)	• trendy na rynku nieruchomości; • dane makroekonomiczne;
World Health Organization (https://www.who.int/)	• informacje o klęskach żywiołowych, katastrofach oraz epidemiach.

Źródło: opracowanie własne.

3.7 Podsumowanie analizy źródeł

Najważniejsze zidentyfikowane źródła danych dla sektora bankowego, to: strony domowe banków, rankingi, fora i komentarze w mediach społecznościowych, strony prowadzone przez instytucje z sektora bankowego oraz portale branżowe. Źródła te różnią się strukturą danych, podmiotem, który generuje dane oraz zasięgiem.

Źródła zawierają dane ustrukturyzowane, nieustrukturyzowane lub półstrukturyzowane. Każdy z tych typów wymaga innego podejścia do pobierania danych, co zostało szerzej omówione w kolejnych rozdziałach.

Wybierając źródła, z których pobierane będą dane, należy zwrócić uwagę na podmiot, który te dane utrzymuje lub tworzy. Mogę to być dane, które podmioty publikują na swój temat (np. zestawienia finansowe, oferty własne banków na ich stronach internetowych) – tego typu informacje charakteryzują się wysoką jakością, choć ze względów marketingowych będą koncentrowały się na przedstawieniu podmiotu w jak najlepszym świetle, mogą więc być nieobiektywne. Część źródeł utrzymywanych jest przez niezależne instytucje sektora bankowego (np. rejestry utrzymywane przez jednostki rządowe) lub podmioty zajmujące się dostarczaniem informacji o sektorze (np. portale branżowe, blogi tematyczne). Tego typu źródła mają zwykle zadowalającą jakość, tj. zawierają informacje charakteryzujące się określonymi atrybutami jakości informacji (Abramowicz, 2008), choć w przypadku portali branżowych czy blogów można trafić na tzw. teksty sponsorowane, które nie są w pełni obiektywne i mogą nie być oceniane jako wiarygodne. Wreszcie, wiele źródeł danych

gromadzi opinie klientów. Są to źródła szczególnie cenne dla podmiotów sektora bankowego, ponieważ zawierają subiektywne opinie klientów o ofercie własnej i konkurencji, porównania ofert, itp. Takie opinie są publikowane m.in. na portalach społecznościowych lub forach internetowych. Dane z serwisów społecznościowych mogą służyć wczesnemu wykrywaniu potrzeb rynkowych i identyfikacji kryteriów, jakimi kierują się klienci przy podejmowaniu decyzji związanych z wyborem ofert usług bankowych. Ponadto, to właśnie w mediach społecznościowych pojawiają się zwykle pierwsze sygnały o problemach w zakresie cyberbezpieczeństwa – użytkownicy dzielą się anomaliami, które zauważą, przestrzegają przed napotkanymi próbami wyłudzeń czy nawet zauważonymi błędami w aplikacjach mobilnych banków. Jednakże dane z tych źródeł najczęściej wymagają dalszego przetwarzania metodami analizy sentymentu i metodami przetwarzania języka naturalnego (NLP).

Przeprowadzona analiza źródeł danych wykazała, że większość potrzeb informacyjnych można (w różnym stopniu) zaspokoić danymi z różnych typów źródeł. Z tego względu przed przystąpieniem do tworzenia narzędzia pobierającego dane, należy przeprowadzić dokładną analizę potrzeb informacyjnych danego podmiotu oraz potencjalnych źródeł, związanych z określonym problemem biznesowym.

Ponadto, łącząc dane z różnych źródeł, możliwe jest uzyskanie pełniejszego obrazu analizowanego zjawiska oraz podwyższenie jakości danych (np. poprzez weryfikację danych w różnych źródłach lub uzupełnienie brakujących elementów).

Kluczowym ograniczeniem automatycznego pobierania danych z Internetu są przepisy prawa i licencje obejmujące źródła. Właściciele źródeł decydują o możliwościach ich dalszego wykorzystania oraz ewentualnych kosztach, które mogą się z tym wiązać. Koszty licencyjne źródła muszą być każdorazowo analizowane w odniesieniu nie tylko do danego źródła, ale też konkretnego zastosowania, gdyż często są uzależnione od liczby wykonywanych zapytań. Należy wziąć jednak pod uwagę, że jeżeli źródło po raz pierwszy udziela licencji na wykorzystanie danych, może zmienić charakter działalności na komercyjny, a tym samym zdecydować o wprowadzeniu bardziej restrykcyjnej moderacji treści.

ROZDZIAŁ IV

PROGRAM ANALITYCZNO BADAWCZY

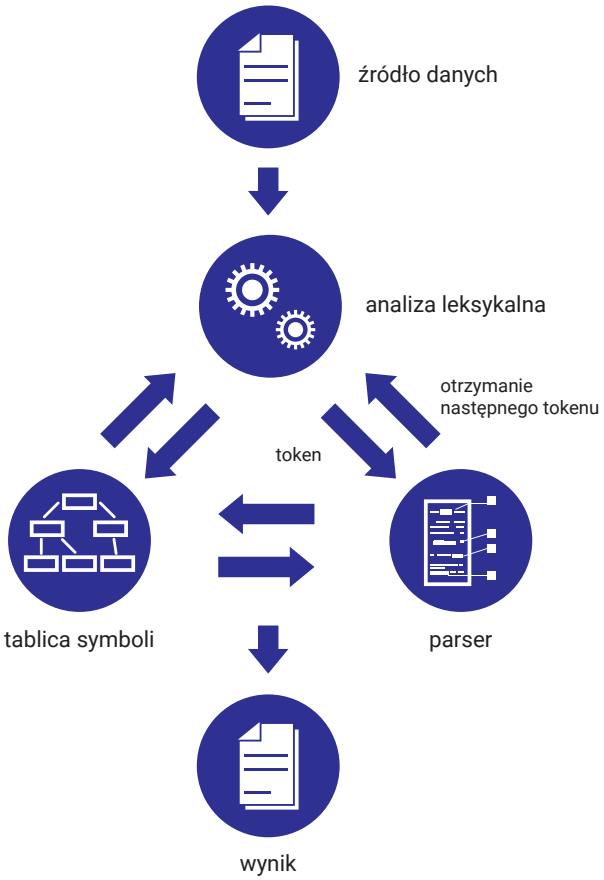
Narzędzia do automatycznego pozyskiwania danych z Internetu

Kolejnym krokiem w procesie pobierania danych, po identyfikacji potrzeb informacyjnych i źródeł, które mogą te potrzeby zaspokoić, oraz ocenie kosztów i korzyści pozyskania danych z określonych źródeł, jest wybór właściwego narzędzia. Niniejszy rozdział zawiera wyjaśnienie pojęcia *web scraping* (rozdział 4.1). Następnie omówiono techniczne i prawne aspekty automatycznego pobierania danych ze źródeł internetowych (rozdziały 4.2 i 4.3). Pozostała część rozdziału przedstawia popularne narzędzia do pozyskiwania danych oraz ich porównanie. Po przeanalizowaniu zalet i wad dostępnych narzędzi oraz sprawdzeniu, czy ich funkcjonalności pozwalają na efektywne pozyskiwanie danych z wybranego źródła, można przystąpić do budowy dedykowanego scrapera (patrz rozdział 5).

4.1 Web Scraping

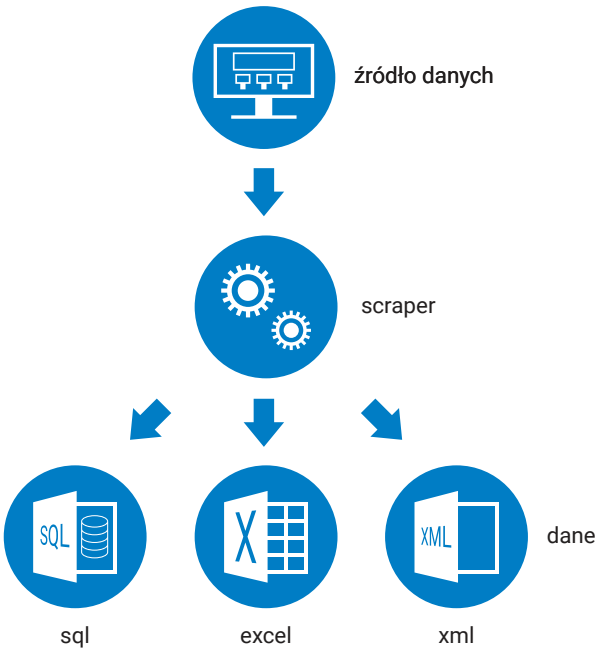
W niniejszym opracowaniu pojawiają się następujące terminy: *parsing* (analiza składniowa), *web scraping* (ekstrakcja danych) i *web crawling* (indeksowanie stron internetowych), których definicje przedstawiono poniżej.

Parsing (analiza składniowa) (rysunek 7.) to element analizy tekstu, związany z określeniem jego struktury gramatycznej. Jest to drugi etap tworzenia tzw. drzewa składniowego (*abstract syntax tree*). Drzewo składniowe jest sposobem reprezentacji wyników analizy składniowej zdania lub słowa. Każdy węzeł reprezentuje pewne konstrukcje językowe (zbudowane zgodnie z gramatyką danego języka), a potomkowie to części składowe tych konstrukcji. Drzewa składniowe są wykorzystywane przez kompilatory oraz do analizy tekstów. Pierwszym etapem jest analiza leksykalna, polegająca na generowaniu tokenów. Token to najmniejszy element, który jest poddawany analizie, zazwyczaj pojedyncze słowo. W zależności od języka, wyszczególnienie tokenów może przybierać różne poziomy trudności. Przykładem łatwiejszego wydzielania tokenów w języku polskim jest wyraz „ciemnoróżowy", który da się przedstawić w postaci dwóch tokenów „ciemny" oraz „różowy". Rezultatem tej części procesu może być podział strumienia znaków, będącego elementem wejściowym, na posiadające znaczenie symbole. Wykorzystuje się do tego zasady gramatyczne danego języka bądź wyrażenia regularne. Proces parsowania polega na sprawdzeniu, czy dany token tworzy dopuszczalny w danym języku wyraz. Zazwyczaj analiza zachodzi w odniesieniu do gramatyki bezkontekstowej (formalnej) bądź notacji BNF (postać Backusa-Naura) (Aho, Sethi, Ullman, 2002). W trakcie sprawdzania tokenów rekurencyjnie tworzone są komponenty, które składają się na wyrażenia. Co więcej, uwzględniana jest też kolejność ułożenia komponentów. Trzeci etap tworzenia drzewa składniowego to tzw. *semantic parsing* (analiza semantyczna), który polega na zbadaniu znaczenia wyrażenia powstałego w poprzednim etapie (Mulik, Shinde i Kapase, 2011).

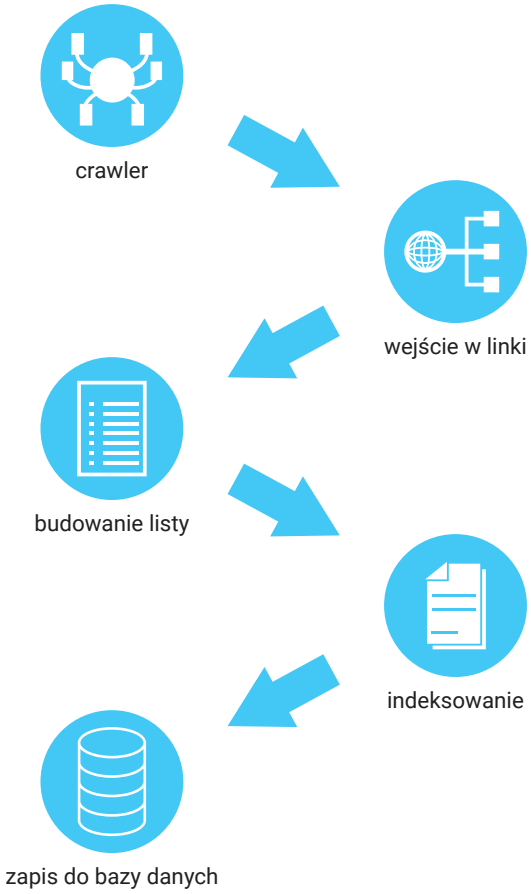


Rysunek 7. Schemat procesu parsowania.
Źródło: opracowanie własne.

Web scraping (screen scraping, web harvesting, ekstrakcja danych) (rysunek 8.) polega na automatycznym pobieraniu danych ze stron WWW (World Wide Web) i zapisywaniu ich w pliku bądź bazie danych w celu dalszego przetworzenia (Krotov i Silva, 2018, Maślankowski, 2019). Zazwyczaj dane są pobieranie z sieci przy wykorzystaniu protokołu HTTP (Hypertext Transfer Protocol). Ze względu na ilość danych znajdujących się w Internecie, *web scraping* jest uznany za efektywny sposób na uzyskanie wyjątkowo dużych zbiorów danych (tzw. Big data) (Landers i in., 2016). Podczas procesu pobierania, program wykorzystywany do ekstrakcji powinien naśladować ruchy i zachowania użytkownika strony, co szerzej zostało opisane w rozdziale 2.3.1. (Techopedia, 2019).



Rysunek 8. Schemat procesu web scrapingu.
Źródło: (ProWebScraping, 2020)



Rysunek 9. Proces web crawlingu.
Źródło: (ProWebScraping, 2020).

Web crawling (rysunek 9.) polega na automatycznym i systematycznym nawigowaniu programu po stronie internetowej, korzystając z zawartych na niej linków (Chatterjee i Nath, 2017, Thelwall, 2001). *Web crawler* jest określany również jako: web spider, spiderbot, web wanderer, robot internetowy, robot indeksujący, pająk, pełzacz. Głównym zastosowaniem tego typu oprogramowania jest tworzenie kopii odwiedzanych stron internetowych, aby umożliwić ich dalsze przetwarzanie. Pająki używane są przez wszystkie wyszukiwarki internetowe, które następnie indeksują pobrane w ten sposób strony. Proces crawlingu wiąże się z wieloma wyzwaniami, kluczowym jest ogrom sieci Web. Robot indeksujący musi mieć możliwość indeksowania stron z dużą szybkością, przy czym nie może on blokować serwerów odwiedzanych stron (Najork i Heydon, 2002).

Robot internetowy rozpoczyna indeksowanie stron od podanego, początkowego adresu URL, z którego nawiguje dalej. Istotne jest nałożenie określonych warunków wyszukiwania na skrypt wyszukujący kolejne linki (np. typy plików, które ignoruje; liczbę linków, które ma odwiedzić). Wiele stron w swoim katalogu głównym zawiera plik robots.txt, który określa w jaki sposób crawlery powinny odwiedzać stronę. Web crawler jest również wykorzystywany do (Science Daily, 2019, Chatterjee i Nath, 2017):

- sprawdzania poprawności linków na stronie,
- sprawdzania poprawności kodu HTML,
- zbierania określonych rodzajów informacji (np. adresów e-mail),
- filtrowania spamu,
- wykrywania naruszenia praw autorskich.

Poza tym *web crawler* jest pośrednio wykorzystywany w procesie *web scrapingu*, gdyż pozwala na automatyczne nawigowanie po stronach i indeksowanie treści. Może być powiązany ze scrapingiem, który analizuje strukturę zaindeksowanej strony, co pozwala na wyodrębnienie z niej określonych danych. Jednakże scraping można przeprowadzać również jako osobny proces, w którym to użytkownik wskazuje adres strony do pobrania.

Crawlery są stosowane również w cyberbezpieczeństwie, np. do wykrywania prania brudnych pieniędzy dzięki badaniu powiązań pomiędzy witrynami internetowymi (w tym metadanych, takich jak dane rejestrowe domen).

Tabela 6. przedstawia różnice oraz podobieństwa omówionych terminów.

Tabela 6. Porównanie parsingu, web scrapingu oraz web crawlingu.

ZASTOSOWANIE/TERMIN	Parsing	Web Scraping	Web Crawling
Analiza gramatyczna tekstu	TAK	NIE	NIE
Dzielenie treści na tokeny	TAK	NIE	NIE
Ekstrakcja wybranych informacji	NIE	TAK	TAK
Tworzenie kopii odwiedzanych stron internetowych	NIE	TAK	TAK
Automatyczne poruszanie się po stronach internetowych	NIE	NIE	TAK
Filtrowanie zduplikowanych danych	NIE	NIE	TAK

Źródło: opracowanie własne.

Web scraping, zwany również ekstrakcją danych, nie powinien być pierwszym wyborem, gdy przychodzi do pobierania danych. Jeszcze przed rozpoczęciem prac nad scraperem (programem służącym do *web scrapingu*), podstawą jest sprawdzenie, czy strona lub jej administrator oferuje interfejs programu aplikacyjnego (API, patrz rozdział 3.2.), który pozwala na szybkie pobieranie danych bezpośrednio z bazy danych bez używania zewnętrznych narzędzi (tj. scraperów). Jeżeli oferowane API daje dostęp do pożądanych informacji, to jest to znacznie łatwiejsze rozwiązanie niż tworzenie scrapera. Jeśli jednak API dla strony nie istnieje, należy sprawdzić, czy interesujące nas dane nie są dostępne do bezpośredniego pobrania (import z pliku np.: CSV, XML). W przypadku braku bezpośredniego dostępu do danych (import z pliku lub dostęp do API) należy przystąpić do procesu tworzenia scrapera.

W celu stworzenia narzędzi do automatycznej ekstrakcji danych (scraper) niezbędne jest zrozumienie pojęć związanych z *web scrapingiem*, a co za tym idzie wiedza na temat sieci WWW oraz technologii internetowych. Wyjątkowo użyteczne w tym procesie są wyrażenia regularne (regular expression), dzięki którym możliwe jest zlokalizowanie określonego ciągu znaków w tekście. Wyrażenia regularne zwracają na wyjściu ciąg znaków, na podstawie którego można podjąć dalsze działania np. podzielić tekst, usunąć niepotrzebne znaki, zamienić kropki na przecinki etc. (Becchi i Crowley, 2008).

4.2 Uwarunkowanie prawne, etyczne oraz techniczne

Pracę nad pozyskaniem danych za pomocą zautomatyzowanych metod należy rozpocząć od zapoznania się z uwarunkowaniami prawnymi, tzn. sprawdzeniem, czy administrator strony zezwala na pozyskiwanie z niej danych w automatyczny sposób. Informacje na ten temat można znaleźć w publikowanych na większości stron Warunkach Użytkowania (ang. terms of use, terms and conditions) lub pliku robots.txt (Sun, Zhuang, Lee Giles, 2007, Yang, Liao, 2010). Warunki użytkowania stron zawierają takie elementy, jak prawa użytkowania, prawa własności intelektualnej, obowiązki podmiotu przetwarzającego dane (np. konieczność podania pierwotnego źródła) itp. Dla przykładu przedstawiony został fragment praw użytkowania ze strony YIL: *„Jak określono w niniejszych warunkach oraz we wszystkich obowiązujących przepisach i regulacjach, firma YIL udziela użytkownikowi prawa do przeglądania i pobierania materiałów udostępnianych na witrynie internetowej firmy YIL wyłącznie dla celów osobistych i niekomercyjnych* [...]." (YIL, 2020). „Robots.txt" to plik tekstowy przekazujący informacje robotom wyszukiwarek, jakie dane mogą indeksować w witrynie (np. strony, pliki). Głównym jego celem jest odpowiednie zarządzanie ruchem indeksowania, by nie przeciążać serwera żądaniami. Dodatkowo, może posłużyć do wykluczenia plików graficznych, wideo i dźwiękowych z wyników wyszukiwania wyszukiwarek (Google, 2020). W przypadku nieuzyskania dostatecznych informacji na temat warunków użytkowania na stronie, w pierwszej kolejności rekomenduje się skontaktowanie z autorem strony. Działanie to ma na celu wyjaśnienie polityki dotyczącej uzyskania dostępu do danych bądź kupna zezwolenia na ich pobieranie. Pomocne może okazać się zapoznanie z działem FAQ (Frequently Asked Questions). W razie wątpliwości co do treści regulaminu na stronie, konieczne jest kierowanie się przepisami prawnymi obowiązującymi w danym państwie (Allen, Burk, Davis, 2006, Sun, Councill, Lee Giles, 2010). Wśród administratorów stron możemy wyróżnić następujące grupy: administratorzy, którzy nie sprawdzają czy ktoś scrapuje dane z ich witryn oraz administratorzy, którzy traktują to bardzo poważnie

i wyciągają konsekwencje, tak jak np. American Airlines (American Airlines, Inc. vs. FareChase, Inc., 2003) oraz Southwest Airlines (Southwest Airlines Co., 2004), którzy złożyli pozew przeciwko firmie FareChase na początku 2000 roku. FareChase, w celu poprawy oferty lotów dla swoich klientów, scrapowało strony linii lotniczych. Sąd, powołując się na ustawę „Computer Fraud and Abuse Act (CFAA)" (Hagen i in.m 2012), uznał działania FareChase za naruszenie amerykańskiego prawa, a wkrótce po rozstrzygnięciu sprawy firma upadła (Krotov, Silva, 2018). Sprawa FareChase została uznana za pogwałcenie zasad etyki, co w tym przypadku wiązało się z konsekwencjami prawnymi.

Scrapowanie danych jest wciąż obszarem, w którym nie wszystkie nieetyczne działania zostały już skodyfikowane, a przepisy prawne różnią się w zależności od kraju. W Polsce zasady wykorzystania danych reguluje ustawa z dnia 27 lipca 2001 roku o ochronie baz danych (Ustawa, 2001). Obecnie w Unii Europejskiej należy przestrzegać rozporządzenia o ochronie danych osobowych (RODO, 2016). Powinno się unikać pobierania danych obywateli UE w przypadku braku wyraźnej zgody i uzasadnionego powodu. Imię i nazwisko, adres zamieszkania, e-mail zawierający imię i nazwisko, numer dowodu/paszportu, IP, informacje o zatrudnieniu, informacje medyczne, nagrania w formie wideo lub audio to przykłady danych osobowych. Natomiast dane anonimowe, numer firmy, adresy e-mail niezawierające imienia i nazwiska mogą być pobrane i przechowywane. RODO może nie mieć zastosowania, jeżeli pobrane dane osobowe należą do mieszkańców spoza Unii Europejskiej oraz Europejskiego Obszaru Gospodarczego. (Goodman, Flaxman, 2017)

Należy pamiętać, że scrapowanie samo w sobie nie jest nieetyczne, ale wszystko zależy od tego, jaki jest cel pozyskania danych. Jeśli scrapowanie dotyczy, np. treści strony (tj. komentarzy, recenzji, opisów, postów) w celu pozyskania potencjalnych klientów, to jest to nieetyczne działanie, o ile profil działalności jest zbliżony do profilu firmy, z której strony pobierane są dane. Jeśli natomiast celem jest stworzenie katalogu potencjalnych klientów dla własnej firmy, to nie jest to naruszenie zasad etyki, pod warunkiem, że prowadzona działalność jest niekonkurencyjna (https://www.datahen.com/legal-ethical-aspects-data-scraping/). Trzeba również pamiętać, iż działania nieetyczne jak i te etyczne mogą naruszać np. prawa autorskie.

Przykładem tego typu działania jest spór powstały pomiędzy portalem społecznościowym LinkedIn, a firmą hiQ zajmującą się analizą danych dotyczących zasobów ludzkich (Goldfein, Keyte, 2017). Model biznesowy hiQ opierał się na analizie pobranych danych w celu określenia między innymi jakie umiejętności posiadają pracownicy oraz jakie szkolenia powinni odbyć. W maju 2017 roku LinkedIn wysłał pismo do hiQ z prośbą o zaprzestanie automatycznego pobierania danych z profili publicznych LinkedIn twierdząc, że takim działaniem hiQ narusza między innymi wspomnianą powyżej ustawę CFAA. Odpowiedzią hiQ był nakaz sądowy, dotyczący usunięcia wszelkich barier technicznych w dostępie hiQ do informacji na publicznych profilach w serwisie LinkedIn. Finalnie sąd odrzucił wszelkie roszczenia ze strony LinkedIn. Historia ta pokazuje, że pobieranie danych osobowych z witryn internetowych, które udostępniają dane osobowe publicznie, nie zawsze wiąże się z łamaniem prawa, nawet jeśli autorzy stron nie wyrażają na to zgody (Goldfein, Keyte, 2017).

Scrapowanie stron internetowych może powodować problemy nawet wówczas, gdy nie jest wykonywane dla celów komercyjnych, a na przykład w celu prowadzenia badań naukowych. Przykładem jest nadmierne obciążenie strony internetowej poprzez wysyłanie zbyt dużej liczby zapytań scrapera do serwera. Zbyt duża aktywność może spowodować brak dostępu do witryny dla innych użytkowników. Może to zostać uznane jako cyberatak, tak zwany Denial of Service.

Denial of Service (DoS) jest atakiem na system komputerowy lub usługę sieciową w celu uniemożliwienia jej działania poprzez zajęcie wszystkich wolnych zasobów. Natomiast atak DDoS (Distributed Denial of Service) jest jego rozszerzeniem i polega na zaatakowaniu ofiary z wielu miejsc jednocześnie. Przygotowując atak DoS na stronę internetową, haker tworzy sztuczny ruch na stronie, blokując tym samym do niej dostęp dla innych użytkowników. Może to spowodować straty finansowe, jak również wizerunkowe. DoS, jak również DDoS uznawane są za naruszenie cyberbezpieczeństwa (McDowell, 2009). Bez względu na przesłanki, powodowanie nieuzasadnionego obciążenia serwera może wiązać się z konsekwencjami prawnymi. Użytkownik rzadko posiada informację o wydajności i przepustowości serwera, tak więc przystępując do scrapowania stron internetowych, dobrą praktyką jest stosowanie odstępów czasowych pomiędzy kolejnymi zapytaniami. Przy braku stosowania stosownych pauz, scraper może zostać zablokowany przez administratora za zbyt duże obciążanie strony spowodowane zbyt częstymi zapytaniami (Haque, Singh, 2015).

Głównym argumentem, który przemawia za używaniem scraperów, jest szybkie pobieranie zawartości strony internetowej. Roboty pobierające informacje działają znacznie szybciej niż człowiek, który musiałby ręcznie nawigować między witrynami i kopiować potrzebne informacje. Warto jednak podkreślić, że samo przygotowanie scrapera

(implementacja, zdefiniowanie reguł ekstrakcji informacji) oraz jego późniejsze utrzymanie (dostosowanie reguł do zmieniającej się struktury witryny) może być bardzo czaso- i kosztochłonne.

Jako podsumowanie przedstawiona została Tabela 7., w której zamieszczone zostały pomocne praktyki podczas ekstrakcji danych wraz z ich krótkimi opisami.

Tabela 7. Praktyki przy ekstrakcji danych.

Praktyka	Opis
Warunki użytkowania	Sprawdzić warunki użytkowania strony bądź regulamin.
Robots.txt	Jeżeli istnieje, dokładnie przeczytać plik, aby uzyskać informacje, do jakich treści scraper/crawler ma dostęp.
FAQ	Sprawdzić na stronie sekcje „Najczęściej zadawane pytania”.
Kontakt z autorem strony	W razie potrzeby skontaktować się z autorem strony w celu doprecyzowania, w jaki sposób można wykorzystywać dane dostępne na stronie.

Źródło: opracowanie własne.

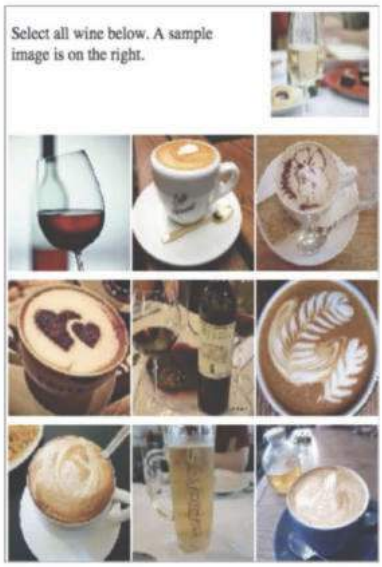
4.3 Ograniczanie możliwości pobierania danych

Administratorzy stron, chcąc ograniczyć dostęp do zawartości stron, poza określeniem podstawowych reguł wykorzystywania danych z witryny, stosują dodatkowe ograniczenia techniczne. Przykładem są rozwiązania bazujące na teście Turinga. Najpopularniejszym rozwiązaniem tego typu, jak podkreśla w swojej pracy McKeanna (2016), jest wykorzystanie techniki zabezpieczenia stron WWW — CAPTCHA oraz jej następcy ReCAPTCHA. CAPTCHA wymaga wykonania zadania, które pozwala odróżnić użytkownika-człowieka od robota i na tej podstawie zarządzać dostępem do treści stron. Przykładami takich zadań są:

- wprowadzenie we wskazanym miejscu tekstu, który został zniekształcony w sposób uniemożliwiający programom OCR (Optical Character Recognition) ich odczytanie (rysunek 10.);

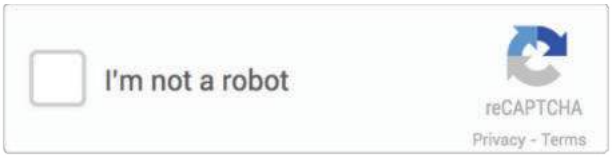
Rysunek 10. Captcha polegająca na odczytaniu i wpisaniu w pole podanych wyrazów.
Źródło: (Von Ahn, Maurer, i inni, 2008)

- wybranie określonych obrazów zgodnych z poleceniem CAPTCHA, np. przejść dla pieszych, autobusów, pomarańczy (rysunek 11.);



Rysunek 11. Captcha polegająca na wybraniu obrazków przedstawiających wino.
Źródło: (Sivakorn, Polakis, Keromytis, 2016)

- zaznaczenie pola potwierdzającego, że osoba korzystająca z serwisu nie jest robotem (zadanie trywialne do wykonania dla człowieka, lecz robotowi trudno jest odnaleźć takie pola, ponieważ w skrypcie strony jest ono przed nim ukrywane) (rysunek 12.).



Rysunek 12. Captcha polegająca na naciśnięciu przycisku.
Źródło: (Sivakorn, Polakis, Keromytis, 2016)

Zadania CAPTCHA są proste do wykonania dla człowieka, natomiast bardzo skomplikowane dla maszyny. Automatyczne rozwiązywanie zadań CAPTCHA jest co prawda możliwe, ale wymaga opracowania dedykowanych skryptów i ich częstego uaktualniania. Celem CAPTCHA jest również wymuszenie opóźnienia działania automatycznych skryptów.

Innymi metodami ograniczającymi dostęp robotów do treści opublikowanych na stronie są formularze, które należy wypełnić lub konieczność logowania. W celu ominięcia takich ograniczeń twórca scrapera zmuszony jest do utworzenia pliku, w którym powinien opisać dokładne instrukcje wypełnienia formularza lub wprowadzić dane potrzebne do logowania.

Poza wyżej opisanymi ograniczeniami, administratorzy korzystają również z innych sposobów umożliwiających blokowanie robotów, o czym w swojej pracy piszą Haque i Singh (2015). Jedną z technik jest refaktoryzacja (Flower, 2018), czyli regularne zmienianie nazw klas i identyfikatorów w kodzie źródłowym strony. Zmiana nie wpływa na funkcjonalność strony, przez co użytkownik nie jest w stanie jej odczuć. Przygotowany scraper ma z góry określone reguły, dotyczące identyfikacji elementów na stronie, zatem nie będzie on przygotowany na zmiany kodu źródłowego i nie zdoła wykonać zadania w należyty sposób. Warto jednak pamiętać, że mając do czynienia z rozbudowanymi stronami, zmiany nazw klas i identyfikatorów nie będą najlepszym rozwiązaniem ze względu na ilość miejsc, które powinny zostać poddane refaktoryzacji.

Dodatkową praktyką stosowaną przez administratorów witryn jest konwersja tekstu na obrazy graficzne. Sprawia ona, że osoba chcąca wyodrębnić dane ze strony jest zmuszona do korzystania z narzędzi OCR, służących do rozpoznawania tekstu na obrazach.

Kolejnym sposobem na rozpoznanie i zablokowanie działalności skryptów pozyskujących dane na stronie, jest analiza częstości odwiedzin. Jest to metoda polegająca na analizie liczby wejść na stronę z określonego adresu IP w ustalonym przedziale czasowym (Moore, Zuev, 2005). Na podstawie przeprowadzonej analizy wyróżnia się trzy typy użytkowników:

- zachowujący się naturalnie,
- jednostki podejrzane o zautomatyzowane odwiedzanie stron, nad którymi należy utrzymać nadzór,
- jednostki z zablokowanym dostępem, których zachowanie ewidentnie świadczy o używaniu zautomatyzowanych narzędzi.

Pokrewnym sposobem, również rozróżniającym takie same trzy typy użytkowników, jest analiza zachowania użytkownika na stronie. Wynikiem takiej analizy może być np. odpowiedź na pytanie, czy odstępy czasowe między kolejnymi akcjami wykonywanymi przez użytkownika są równe lub wykonywane według pewnego algorytmu/schematu. W przypadku, gdy użytkownikiem strony jest człowiek, jego zachowanie nie odpowiada regularnym wzorcom i trudno je opisać algorytmicznie (np. czas pomiędzy wpisywaniem kolejnych znaków na klawiaturze jest różny, odstępy czasu między klikaniem w elementy strony są nieregularne).

Innym mechanizmem używanym do wykrywania scraperów są tzw. Honeypots (Spitzner, 2003). Są to pułapki, dzięki którym zdemaskowany robot może zostać zablokowany lub wprowadzony w błąd. Pułapki te możemy dzielić na różne sposoby (McKenna, 2016). Jednym z nich jest podział ze względu na cel działania mechanizmu:

- aktywny honeypot, którego celem jest „złapanie w pułapkę" (zablokowanie działania) robota;
- pasywny honeypot, mający na celu jedynie wykrycie bota i zebranie informacji na temat wykorzystywanych przez niego technik pobierania danych.

Inne podziały dotyczą np. sposobu implementacji mechanizmu czy też poziomu jego interakcji z systemem (Mokube, Adams, 2007). W praktyce, w wyniku działania takiej pułapki dostęp robota do strony może zostać zablokowany lub może on zostać przekierowany do nieskończonej pętli zapytań na stronie, przez co nie osiągnie swojego celu.

Specyfika działania crawlerów sprawia, że są one bardzo wrażliwe na błędy i zmiany w strukturze strony (powodują one, że reguły ekstrakcji danych nie będą działały poprawnie), jednak błędy w pobieranych danych mają mniejsze znaczenie. Większość danych jest pobierana ze stron w formacie tekstowym i ewentualnie później konwertowana do innych formatów (np. liczbowych, daty). Błędy językowe, gramatyczne czy literówki, pojawiające się w tekstach na stronie (wyjątkowo częste w przypadków postów w mediach społecznościowych), nie mają wpływu na działanie

robotów internetowych, ponieważ nie analizują one znaczenia tekstów, a jedynie zapisują pobrane dane w bazie. Jednakże niska jakość pobranych danych tekstowych może powodować, że ich dalsza automatyczna analiza będzie znacznie utrudniona. Przykładowo, algorytmy analizy sentymentu (które nie są przedmiotem niniejszego opracowania), mogą nie działać poprawnie na tekstach zawierających literówki czy błędy językowe.

4.4 Narzędzia

W przypadku *web scrapingu* (Kaczmarek, 2006, Kowalkiewicz, 2006) najistotniejszym elementem jest odpowiedni dobór narzędzi (gotowych rozwiązań dostępnych na rynku, charakteryzujących się wysoką złożonością). Ze względu na wysokie zróżnicowanie w konstrukcji współczesnych stron internetowych, niemal każdy scraper musi być „szyty na miarę", tj. dostosowany do konkretnej struktury strony i technologii, w której jest wykonana. Należy też mieć na uwadze fakt, że w Internecie znajdują się zasoby archiwalne, które mogą okazać się użyteczne w nauce i biznesie, lecz problematyczne do scrapowania ze względu na nietypową i wysoce przestarzałą strukturę strony. Jak zostało to już wspomniane w rozdziale 4.1, przed rozpoczęciem procesu scrapingu, należy sprawdzić, czy właściciel strony nie zdecydował się na udostępnienie potrzebnych informacji poprzez API – pozyskanie danych w ten sposób jest zazwyczaj efektywniejsze.

Pierwszym etapem procesu ekstrakcji jest analiza struktury strony, z której mają zostać pobrane dane. Istnieją gotowe narzędzia, które ułatwiają taki proces (np. Selector Gadget), jednak bez odpowiedniej wiedzy i umiejętności mogą się okazać bezużyteczne. Następnym krokiem jest wybór odpowiedniego języka programowania. Ze względu na rosnącą popularność analiz Big Data, warto wziąć pod uwagę dwa języki typu open source tj. Python i R. W Python korzystamy z plików zawierających gotowe funkcje, podprogramy i typy danych nazywane bibliotekami. W R tę samą funkcję pełnią pakiety. Dobrą praktyką w procesie scrapowania jest wykorzystanie tego samego języka, w którym wykonywana będzie dalsza praca na danych. Pozwoli to uniknąć konfliktów związanych z różnicami występującymi w konstrukcji bibliotek lub pakietów. Na podstawie dokonanej analizy strony, programista jest w stanie racjonalnie dobrać odpowiednie narzędzia. W przypadku braku zabezpieczeń na stronie, skorzystać można z biblioteki Beautiful Soup (Python) bądź Rvest (R). Są one stosunkowo proste w implementacji, oferując jednocześnie wysoką wydajność. Bardziej zaawansowanym rozwiązaniem jest Scrapy (Python), pozwalający na tworzenie specyficznych crawlerów, pełniących również funkcje web scraperów (tj. programów, które nawigują pomiędzy stronami i równocześnie pobierają treści).

W przypadku konieczności logowania się na stronę bądź zabezpieczeń wykorzystujących przyciski do wyświetlania nowej treści pod tym samym adresem url, możliwe jest zastosowanie narzędzia Selenium (dostępnego w Python i R), gdyż Rvest nie pozwala na automatyczne klikanie w przyciski, a jedynie na nawigację po adresach url. Proces scrapowania w przypadku złożonych i skomplikowanych stron można przyśpieszyć poprzez połączenie dwóch narzędzi, np. Selenium i biblioteki Beautiful Soup (bądź pakietu Rvest w przypadku pracy w R). Dzięki takiemu połączeniu wykorzystać można zalety obu rozwiązań, eliminując przy tym ich słabe strony. Kosztem tego typu łączonych rozwiązań jest jednak zwiększenie zależności, a tym samym wzrost skomplikowania kodu.

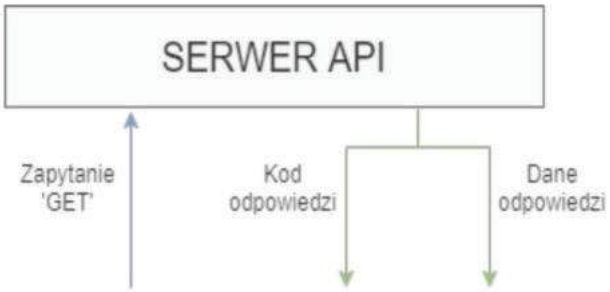
W dalszej części rozdziału szczegółowo zostały omówione wszystkie wyżej wymienione narzędzia (łącznie z API). Dokonano również porównania dostępnych technologii, które ma na celu uproszczenie procesu decyzyjnego związanego z wyborem określonego narzędzia do scrapowania.

4.4.1 API

Rozwiązanie znane jako API (ang. Application Programming Interface) wykorzystywane jest w wielu obecnych na rynku aplikacjach, ponieważ pozwala na łatwą ich integrację z zewnętrznymi źródłami danych. API jest zbiorem reguł, opisujących w jaki sposób programy komunikują się ze sobą (Ministerstwo Cyfryzacji, 2018). Koncepcja ta opiera się o architekturę klient-serwer oraz protokół komunikacyjny HTTP. W sieciach o architekturze klient-serwer urządzenia dzielą się na dwie grupy: serwery, które oferują usługi oraz klientów, którzy korzystają z tych usług (Zelent, 2017).

Zainteresowany pozyskaniem danych klient wysyła zapytanie (ang. request) do serwera (rysunek 13.). Zapytanie składa się z elementów umożliwiających rozpoznanie przez serwer zakresu danych interesujących klienta, a także weryfikację posiadanych przez niego uprawnień. W oparciu o zapytanie i walidację klucza API (klucz API to fragment kodu oprogramowania, który pozwala na zidentyfikowanie użytkownika lub maszyny (Rogalski, 2019), serwer przeszukuje posiadaną bazę danych, a następnie odsyła w odpowiedzi dane w oczekiwanym formacie.

Istnieje kilka rodzajów API, które powstały ze względu na wygodę ich użytkowania. Rodzaj API jest wybierany w zależności od potrzeb użytkowników, zakresu



Rysunek 13. Schemat działania API.
Źródło: opracowanie własne.

udostępnianych danych oraz preferencji dostawcy danych. Wyróżniamy następujące rodzaje API (Biehl, 2015):

- REST (Representational State Transfer),
- SOAP (Simple Object Access Protocol),
- GraphQL,
- Falcor,
- RPC Style (Remote-Procedure-Call),
- GRPC (general-purpose Remote Procedure Calls).

Większość różnic, jakie można wskazać w wymienionych rodzajach API, dotyczy bezpośrednio projektowania API. Podstawową różnicą dla użytkownika jest sposób oraz zakres pozyskiwania danych. Przykładowo, w GraphQL użytkownik sam wpływa na to, jakie dane zostaną pobrane poprzez skonstruowanie odpowiedniego zapytania (Buna, 2016). W REST jest odwrotnie, przy projektowaniu API zostają określone zapytania, jakich użytkownik może użyć przy podaniu odpowiednich parametrów (zostanie to dokładniej opisane w dalszej części rozdziału). Zatem, konsekwencją dla użytkownika jest różny poziom zaawansowania technologicznego, jaki jest od niego wymagany, aby odpowiednio obsłużyć API.

Aktualnie najczęściej używanym rodzajem API jest REST. REST to architektura, która określa sposób tworzenia usług w oparciu o protokół HTTP. Serwer usługi udostępnia zasoby, do których dostęp jest możliwy poprzez standardowe operacje http (Kaczmarek, 2014):

- GET – pobranie zbioru danych, informacji o zasobach,
- POST – tworzenie nowych zasobów,
- PUT – aktualizowanie istniejącego zasobu,
- DELETE — usunięcie zasobu.

Pobierając dane przez wykorzystanie REST API wykonywany jest ciąg czynności:

- zdefiniowanie zapytania (najczęściej przyjmuje format JSON, ale może również przyjąć format XML lub zwykłego tekstu) przez użycie odpowiedniej metody HTTP,
- wysłanie zapytania na adres usługi,
- zwrócenie odpowiedzi na zapytanie przez interfejs usługi,
- odebranie odpowiedzi w formacie JSON lub XML (Szczepaniak, 2018).

Wymiana zapytań i odpowiedzi następuje tak często, jak zostanie to określone przez dostawcę danych. W wielu przypadkach, określone limity pozwalają na dostateczne pozyskanie danych przez klienta, tak aby mógł spełnić swoje oczekiwania i potrzeby. Dostawcy API stosują różne modele biznesowe, np. nielimitowany darmowy dostęp, dostęp w całości płatny lub oferują pulę darmowych zapytań, po wyczerpaniu której kolejne są już płatne (może się ona odnawiać okresowo lub nie). Przykład zapytania w języku Python z wykorzystaniem formatu JSON przedstawia rysunek 14.

Przykład odpowiedzi uzyskanej od serwera w formacie JSON:

```
import requests
import json

response = requests.get( link_do_uslugi_w_formacie_json )
print(response.json())
```

Rysunek 14. Przykład zapytania do API.
Źródło: opracowanie własne.

```
{
    „typ”: „artykuł”,
    „id”: „1”,
    „atrybuty”: {
        „tytuł”: „tytuł danych 1”
        },
    „linki”: {
        „self”:”http://przykłady.com/artykuły/1”
    }
}
```

Dostawca API może liczyć na szereg korzyści, jakie daje udostępnienie posiadanych zasobów zewnętrznym oprogramowaniom, np.:

- Łatwość integracji pomiędzy systemami i źródłami danych.
- Unikanie niekontrolowanego scrapingu (udostępniając API dostawca nadzoruje zakres udostępnianych danych).
- Tworzenie sieci powiązań biznesowych (twórcy aplikacji opierających się na danych pobieranych przez API stają się partnerem biznesowym dostawcy danych).
- Możliwość analizowania trendów na rynku (badanie popytu na informacje konkretnej treści).
- Ułatwienie monetyzacji posiadanych zasobów i zwiększenie wartości posiadanych zbiorów zasobów (stosowanie opłat za dostęp do API, poprawa wskaźnika konwersji sprzedaży, podpisywanie długoterminowych umów za dostęp do danych) (Jeffery, 2014).

Z drugiej strony, utrzymywanie API, szczególnie oferowanego komercyjnie, wymaga ponoszenia przez dostawcę kosztów infrastruktury czy zachowania określonych umowami parametrów usługi (uzgodnionych w SLA).

4.4.2 Beautiful Soup 4

Beautiful Soup 4 (BS4) (Richardson, 2007) jest biblioteką języka Python opublikowaną w 2012 roku i służącą do pobierania danych z plików HTML oraz XML. BS4 umożliwia relatywnie szybką i prostą implementację w celu pobrania danych.

Beautiful Soup 4 przekształca dokument HTML w drzewo obiektów (hierarchiczną strukturę danych) języka Python. Rozróżnia się 4 główne typy obiektów:

- Obiekt Tag, odpowiada znacznikowi html w oryginalnym dokumencie HTML. Najważniejszymi cechami obiektu Tag jest nazwa oraz atrybuty.

  Przykład: Znacznik html posiada nazwę „b” oraz atrybut „id”, którego wartością jest „zgodnie”. Atrybuty te są łatwo modyfikowalne przez użytkownika.

- Obiekt NavigableString odpowiada fragmentowi tekstu w znaczniku html, zatem BS4 wykorzystuje NavigableString do przechowywania fragmentów tekstu.

Rysunek 15. w linii 1 przedstawia import metody „BeautifulSoup” z biblioteki „bs4”. W linii 2 zmienna „soup” wywołuje metodę „BeautifulSoup()” na znaczniku html, który ma zawartość „ścieżka” (przedstawiony na rysunku kolorem czerwonym). W kolejnym kroku (linia 3), do zmiennej „tag” zostaje przypisana zmienna „soup” ze znacznikiem o nazwie „b”. Następnie metoda „print()” wyświetla wartość zmiennej „tag” (linia 4). Wynik metody „print()” to wartość „ścieżka”, która znajduje się poniżej przedstawionego kodu. Obiekt BeautifulSoup reprezentuje dokument jako całość oraz obsługuje większość metod wymienionych w dalszej części dokumentu.

```
1 from bs4 import BeautifulSoup
2 soup = BeautifulSoup('<b class="boldest">ścieżka</b>')
3 tag = soup.b
4 print(tag.string)
```

```
ścieżka
```

Rysunek 15. Przykład użycia obiektu NavigableString.
Źródło: opracowanie własne.

- Obiekt Comment jest specjalnym typem obiektu NavigableString. Kiedy wystąpi jako część dokumentu HTML, obiekt Comment wyświetlany jest ze specjalnym formatowaniem.

Przed rozpoczęciem pracy z narzędziem BS4 należy wybrać parser, czyli program, który dokona analizy składniowej danych wejściowych tak, aby określić ich strukturę. Twórca biblioteki rekomenduje używanie parsera “lxml”, ze względu na jego szybkość działania. Należy jednak pamiętać, że inne parsery trzeba najpierw zainstalować.

Kolejnym argumentem funkcji BeautifulSoup() jest link do źródła danych. Ponadto dane należy zdefiniować zmienną, która zawiera atrybuty biblioteki Beautiful Soup 4 (Richardson, 2007). Rysunek 16. przedstawia przykład wyboru parsera. Z biblioteki „bs4” importowana jest metoda „BeautifulSoup”. Następnie zmienna „soup” wywołuje metodę „BeautifulSoup()” na znaczniku „html”, który odpowiada analizowanemu źródłu danych, oraz na wybranym parserze, którym w tym przypadku jest „lxml” (oznaczony kolorem czerwonym).

```
1 from bs4 import BeautifulSoup
2 soup = BeautifulSoup(html, 'lxml')
```

Rysunek 16. Przykład wyboru parsera w BS4.
Źródło: opracowanie własne.

W celu nawigacji po utworzonej strukturze danych, można wykorzystać nazwę znacznika zaraz po wystąpieniu zmiennej – przykład został przedstawiony na rysunku 17. Z biblioteki „bs4" importowana jest metoda „BeautifulSoup" (linia 1). Do zmiennej html_doc przypisujemy zawartość naszej strony w języku HTML, która została przedstawiona za pomocą koloru czerwonego (linie 2-14). Następnie zmienna „soup" wywołuje metodę „BeautifulSoup()" na znaczniku „html_doc", który jest naszym źródłem danych oraz na wybranym parserze, w tym przypadku „hmtl.parser" (linia 15). Metoda „print()" (linia 1 w drugim bloku kodu) wyświetla wartość zmiennej „soup" ze znacznikiem o nazwie „p". Wynikiem metody „print()" jest wartość" The Dormouse's story ", która znajduję się poniżej kodu.

```
1  from bs4 import BeautifulSoup
2  html_doc = """
3  <html><head><title>The Dormouse's story</title></head>
4  <body>
5  <p class="title"><b>The Dormouse's story</b></p>
6
7  <p class="story">Once upon a time there were three little sisters; and their names were
8  <a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
9  <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
10 <a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
11 and they lived at the bottom of a well.</p>
12
13 <p class="story">...</p>
14 """
15 soup = BeautifulSoup(html_doc, 'html.parser')
```

```
1  print(soup.p)
```

```
<p class="title"><b>The Dormouse's story</b></p>
```

Rysunek 17. Nawigacja po strukturze dokumentu w BS4.
Źródło: opracowanie własne.

Pozyskując wartość danego atrybutu w znaczniku należy podać nazwę atrybutu w nawiasach klamrowych, jak pokazuje przykład na rysunku 18. Z biblioteki „bs4" importowana jest metoda „BeautifulSoup" (linia 1). Do zmiennej html_doc przypisana zostaje zawartość analizowanej strony w języku HTML, która została przedstawiona za pomocą koloru czerwonego (linie 2-14). Następnie zmienna „soup" wywołuje metodę „BeautifulSoup()" na znaczniku „html_doc", który jest analizowanym źródłem danych oraz na wybranym parserze, w tym przypadku „hmtl.parser" (linia 15). Metoda „print" (linia 1 drugiego bloku kodu) wyświetla wartość zmiennej „soup" ze znacznikiem o nazwie „p" i atrybucie „class", który został przedstawiony kolorem czerwonym. Wynikiem metody "print" jest wartość "['title']", która znajduje się poza prostokątem, w którym umieszczony jest kod.

Najważniejszymi metodami biblioteki Beautiful Soup 4 są (Richardson, 2007):

- find_all() — znajduje wszystkie znaczniki o podanej nazwie,
- find() — znajduje pierwszy znacznik o podanej nazwie,

```
1  from bs4 import BeautifulSoup
2  html_doc = """
3  <html><head><title>The Dormouse's story</title></head>
4  <body>
5  <p class="title"><b>The Dormouse's story</b></p>
6
7  <p class="story">Once upon a time there were three little sisters; and their names were
8  <a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
9  <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
10 <a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
11 and they lived at the bottom of a well.</p>
12
13 <p class="story">...</p>
14 """
15 soup = BeautifulSoup(html_doc, 'html.parser')
```

```
1  print(soup.p['class'])
```

```
['title']
```

Rysunek 18. Pozyskanie wartości atrybutu w BS4.
Źródło: opracowanie własne.

- find_parents() — znajduje wszystkie elementy nadrzędne danego znacznika w obiektowym modelu dokumentu (DOM),
- decompose() — usuwa znacznik z drzewa wraz z jego zawartością,
- replace_with() — usuwa znacznik lub łańcuch i zastępuje go zadeklarowanym znacznikiem bądź ciągiem znaków,
- get_text() — zwraca część tekstową znacznika lub dokumentu.

Biblioteka Beautiful Soup 4 posiada jasną i przejrzystą dokumentację, zapewnia elastyczność i przejrzystość kodu i sprawdza się przede wszystkim przy małych projektach. Z drugiej strony często jest to rozwiązanie zbyt wolne i wymagające dobrej znajomości koncepcji wielowątkowości (Palakollu, 2019). Wielowątkowość jest to zdolność centralnej jednostki obliczeniowej (CPU) lub jednego rdzenia w procesorze wielordzeniowym do wykonania wielu wątków jednocześnie, które są obsługiwane przez system operacyjny. (Akhter i Roberts, 2006).

4.4.3 Scrapy

Scrapy to wysokopoziomowa platforma programistyczna (framework), używana do przeszukiwania i wydobywania ustrukturyzowanych danych ze stron internetowych (web scraping). Charakteryzuje ją szeroki wachlarz zastosowań, począwszy od eksploracji danych (data mining), skończywszy na monitorowaniu i automatycznym testowaniu aplikacji (Scrapy, 2019b). Napisana jest w języku programowania Python oraz działa na systemach operacyjnych: Linux, Windows, Mac OSX oraz BSD. Korzystanie z niej podlega licencji Open Source. Cechuje ją nawet 20-krotnie szybsze działanie, a także mniejsze zużycie pamięci i procesora w porównaniu do konkurencyjnych narzędzi (Palakollu, 2020). Ponadto, jej ekosystem

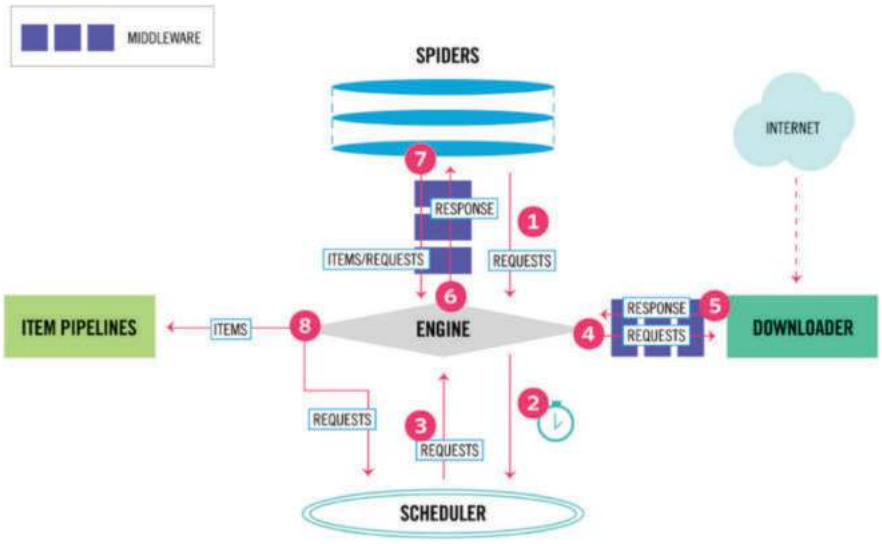
pozwala używać oprogramowania do wykonywania czynności w imieniu użytkownika, czyli proxy oraz wirtualnych prywatnych sieci (VPN) do automatyzacji zadań. Są to główne powody używania tej biblioteki w złożonych projektach. (Palakollu, 2020). Ze względu na swoją szybkość i stopień rozbudowania doskonale radzi sobie ze zbiorami Big Data. Możliwa jest również obsługa współbieżności i programowania asynchronicznego. Oznacza to, że Scrapy jest w stanie obsługiwać zapytania równolegle, co umożliwia prace przy dużych zbiorach danych (Gaurav Singhal, 2020).

Na szeroki wachlarz zastosowań Scrapy składa się również możliwość wykorzystania do ekstrakcji danych za pomocą API (application programming interface) lub jako ogólny web crawler (Dokumentacja Scrapy, 2019). Co więcej, posiada wbudowaną obsługę ekstrakcji danych ze źródeł HTML przy użyciu wyrażeń XPath i CSS (Palakollu, 2020). Wadą tego narzędzia jest dokumentacja, która nie jest tak przyjazna dla początkującego użytkownika, jak ma to miejsce w przypadku innych narzędzi (tj. zawiera bardzo specjalistyczne słownictwo). Powoduje to, że narzędzie jest jednym z trudniejszych do opanowania. Stopień rozbudowania i liczne funkcjonalności, jakie daje Scrapy, również nie ułatwiają nauki. Z drugiej strony, wszystko to sprawia, że Scrapy oferuje bardzo duże możliwości w zakresie web scrapingu czy web crawlingu (Palakollu, 2020).

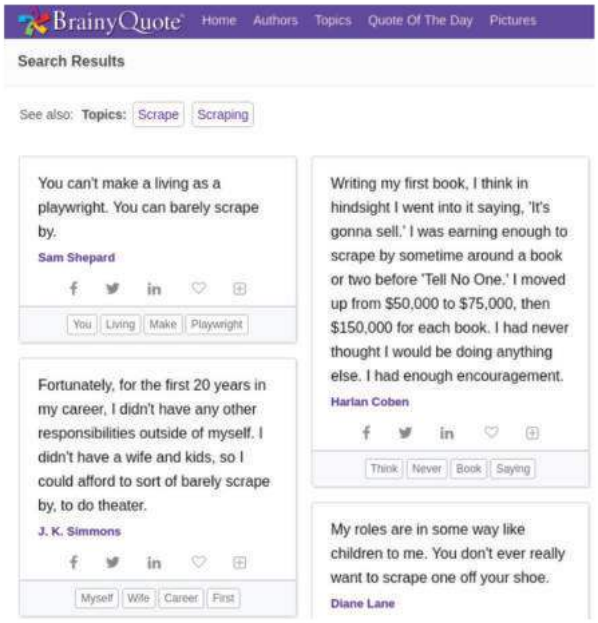
Większość stron internetowych w dzisiejszych czasach bazuje na dynamicznym wyświetlaniu treści. Taki typ witryny sprawić może wiele problemów narzędziom używanym do ekstrahowania zawartości stron internetowych. Wartą uwagi funkcjonalnością charakteryzującą Scrapy jest obsługa takich stron. To znaczy, że korzystając ze Scrapy możliwe jest przechwytywanie treści strony zakodowanej za pomocą języka JavaScript (Dokumentacja Scrapy, 2019b)

Komponenty Scrapy przedstawia rysunek 19.:

- Scrapy Engine jest odpowiedzialny za sterowanie przepływem danych pomiędzy wszystkimi komponentami systemu.
- Scheduler jest wykorzystywany przez silnik do ustalania harmonogramu wysyłania zapytań do witryn (ang. request).
- Downloader odpowiedzialny jest za pobieranie strony internetowej.
- Spiders są to klasy napisane przez użytkownika narzędzia, w których zdefiniowane są reguły pobierania danych z pożądanej strony (tj. jakie dane mają zostać pobrane, jak ma wyglądać nawigowanie po stronie, przechodzenie do następnych linków itp.).
- Item Pipelines odpowiedzialne są za przetwarzanie elementów po ich ekstrakcji. Typowe zadania to na przykład czyszczenie danych HTML, ich walidacja, sprawdzenie czy nie ma duplikatów oraz przechowywanie ich w bazie danych (Architecture overview — Scrapy 1.8.0 documentation, 2019).



Rysunek 19. Schemat przepływu danych między komponentami Scrapy.
Źródło: (Scrapy, 2019).



Rysunek 20. Strona BrainyQuote.
Źródło: (BrainyQuote, 2019).

Przykład tworzenia pająka pozyskującego cytaty ze strony internetowej BrainyQuote (rysunek 20.), zbudowanego przy pomocy Scrapy, został przedstawiony na rysunku 21.

W pierwszej kolejności następuje import biblioteki Scrapy (linia 1). Przy korzystaniu ze Scrapy niezbędne jest utworzenie klasy 'SpiderScrapy', uwzględniającej reguły działania scrapera. Aby mógł on zacząć prace, musi mieć zainicjalizowaną stronę, na której będzie operował — odpowiada za to zmienna 'start_urls' (linia 5). Następnie (linia 7), stworzona zostaje funkcja parse(), w której następuje parsowanie i ekstrakcja danych.

W prezentowanym przykładzie do parsowania wykorzystany został css selektor (Wikibooks, 2019). Na stronie internetowej (rysunek 20.) widzimy zrozumiały dla człowieka ciąg znaków (treść cytatu i jego autora). Aby komputer mógł go przetworzyć, należy wyświetlić kod źródłowy strony i skopiować odpowiednią ścieżkę selektora. Selektory pozwalają odnaleźć dany element w kodzie źródłowym strony. Są one unikalne dla każdego pojedynczego elementu strony, dlatego wykorzystuje się je podczas nawigacji po witrynie w celu odnalezienia interesującego jej fragmentu.

Alternatywą dla selektora css jest selektor xpath, który w praktyce przypomina ścieżkę do elementu na stronie (XML Path Language version 3.1., 2017), w tym przypadku zamiast polecenia quote.css() należy użyć funkcji quote.xpath().

W funkcji parse() (linia 9 na rysunku 21.) zastosowana została pętla for (pojęcie pętli w programowaniu umożliwia cykliczne wykonywanie ciągu instrukcji określoną liczbę razy, do momentu zajścia pewnych warunków, dla każdego elementu kolekcji lub w nieskończoność) w celu pobrania wszystkich znajdujących się na stronie cytatów.

Przykładowy cytat i jego autor pobrany przez pająka, zapisany w formacie JSON, wygląda następująco:

{„text": „I'll play until they have to scrape me off the stage.",
„author": „James Young"}

Zastosowanie selektora css zostało zaprezentowane w kodzie wewnątrz funkcji parse(). Selektor xpath dla cytatu powyżej prezentuje się następująco:

//*[@id="qpos_1_19"]/div/div[1]/div/a.

div[1], który występuje wewnątrz powyższego kodu xpath wskazuje na konkretny element, w tym przypadku pierwszy znajdujący się na stronie cytat. Dla kolejnych cytatów wartość w div[ ] wewnątrz ścieżki xpath rośnie (div[2], div[3] itd.).

```
1  import scrapy

1  class SpiderScrapy(scrapy.Spider):
2
3          name = 'MySpider'
4
5          start_urls = ['https://www.brainyquote.com/topics/scrape-quotes']
6
7          def parse(self, response):
8
9              for quote in response.css('div[class="m-brick grid-item boxy bqQt"]'):
10
11               yield {  'text': quote.css('a.b-qt::text').extract_first(),
12
13                        'author': quote.css('a.bq-aut::text').extract_first() }
```

Rysunek 21. Przykład scrapera napisanego w Scrapy.
Źródło: opracowanie własne.

4.4.4 Rvest

Web scraping wykonywany jest nie tylko przy użyciu języka programowania Python. Do czynności ekstrakcyjnych wykorzystywany jest również jeden z najpopularniejszych w dziedzinie data science (Muenchen, 2019) język programowania – R. Język ten działa w oparciu o licencję open source. Rvest jest najczęściej wykorzystywanym pakietem R w dziedzinie web scrapingu. Został napisany przez Hadleya Wickhama w 2014 roku (Wickham, 2019) i dostępny jest za pośrednictwem repozytorium CRAN (The Comprehensive R Archive Network). Pakiet rvest zawiera w sobie tzw. wrapper functions, których celem jest wywoływanie funkcji z innych pakietów. Szczególnie przydatne są pakiety httr i xml2. Xml2 został napisany przez Jima Hastera (2019) na podstawie biblioteki języka C „libxml2" i umożliwia pracę z plikami typu XML. Httr został stworzony przez Hadleya Wickhama (2019b) i służy do komunikacji klienta z serwerem. Połączenie rvest z funkcjonalnościami tych dwóch pakietów, umożliwia efektywne i stosunkowo proste tworzenie narzędzi do scrapowania danych. Rvest został stworzony do pobierania informacji ze stron internetowych, tak więc odpowiednio użyty może służyć do zebrania dowolnej ilości danych. Dodatkowo rvest jest pakietem powiązanym z innym popularnym pakietem języka R – tidyverse (tidyverse, 2020), który służy do „oczyszczania" pobranych danych (nadawania im jednolitej struktury i formatu). Możliwość wykorzystania funkcji zawartych w tidyverse znacznie przyśpiesza przetwarzanie danych scrapowanych przy pomocy bilbioteki rvest. Jednakże dla początkujących deweloperów R bądź użytkowników, którzy chcą skorzystać tylko i wyłącznie z możliwości pakietu rvest, jego silne powiązanie z innymi pakietami może okazać się problemem.

Rvest wyposażony jest w funkcje pozwalające na pobieranie i parsowanie danych. Umożliwia pozyskanie danych ze stron zbudowanych w schemacie DOM (Document Object Model). Większość współczesnych stron internetowych powstała zgodnie ze schematem DOM, stąd też Rvest jest pakietem uniwersalnym. Obiekty javascript, osadzone na stronach internetowych, są automatycznie wykrywane i konwertowane do obiektów, które język R jest w stanie przetworzyć (np. listy, data frames itd.). Korzystając z pakietu rvest użytkownik ma możliwość zaimplementowania automatycznego modyfikowania pobieranych wartości lub wysłania zapytań typu GET i POST do serwera (Tiru A, Toma E i Necula M, 2017). W ramach pakietu rvest możliwe jest również pobieranie danych przy pomocy selektrów css, jak i xpath.

Istotnym elementem procesu scrapowania danych jest ich czyszczenie, tj. pozbywanie się zbędnych elementów. Za te czynności odpowiadają zewnętrzne pakiety tidyverse i stringr, które można wywoływać z poziomu rvest [Boehmke, 2016].

W celu ekstrakcji danych ze strony z użyciem pakietu rvest, korzysta się m.in. z następujących funkcji (Boehmke, 2016, Bradley i James, 2019):

- read_html – funkcja pozwala na pobranie strony ze wskazanego przez użytkownika adresu URL. Najczęściej wywołanie funkcji skutkuje zapisaniem strony w obiekcie. Po przypisaniu pobranej strony do obiektu, jej pierwotna struktura nie ulega zmianie.

- html_nodes – funkcja jest w stanie pozyskać określoną wartość na podstawie węzła. Przykładowo, szukany tekst może znajdować się w nagłówku, tytule bądź innej sekcji strony internetowej. Warto zaznaczyć, że przy użyciu tej funkcji dokonana zostanie ekstrakcja tekstu wraz ze znacznikami języka HTML.

- html_text – funkcja odpowiada za pozyskanie samego tekstu, znaczniki HTML są pomijane.

- html_table – funkcja pozwala na pobranie tabeli ze strony internetowej. Dodatkowo tabela ta jest od razu konwertowana do formatu data.frame, który jest obsługiwany zarówno przez R, jak i Python.

4.4.5 Selenium

Selenium (Selenium, 2019b), opublikowane w 2004 roku, to darmowy zestaw narzędzi służących głównie do automatycznego testowania aplikacji. Automatyzacja testów pozwala w krótkim czasie zapewnić odpowiednią jakość produktowi końcowemu. Co więcej, wykorzystanie zautomatyzowanych narzędzi po wprowadzeniu zmian w aplikacji zdecydowanie ułatwia i przyspiesza proces ponownych testów. Poza testowaniem, Selenium można używać jako zestawu narzędzi do automatyzacji pozyskiwania różnego rodzaju danych z Internetu, automatycznego wypełniania formularzy na stronach internetowych czy też wykonywania powtarzalnych czynności w sieci. Umożliwia m.in. poruszanie się po witrynach np. za pomocą automatycznego klikania w odpowiednie przyciski umieszczone na stronie, co jest dużą zaletą tego narzędzia (pozostałe narzędzia tego nie umożliwiają).

Najważniejsze komponenty Selenium to (Dokumentacja Selenium,2019):

- Selenium WebDriver,

- Selenium IDE (Integrated Development Environment),

- Selenium Grid.

Selenium WebDriver jest narzędziem umożliwiającym tworzenie web scraperów, służących do poruszania się po stronach i pozyskiwania z nich danych. Jest ono dostosowane do wszystkich powszechnie używanych przeglądarek (Selenium, 2019c) oraz wielu języków programowania. Mimo, iż docelowym językiem jest Java, obsługują go także C#, Python, PHP, Pearl czy Ruby. Pobieranie konkretnych informacji ze strony za pomocą WebDrivera polega na odnalezieniu ich pozycji w kodzie źródłowym witryny. Następnie, do prawidłowego działania programu wymagane jest odwołanie do tego samego miejsca za pomocą selektorów (Sądel, 2018) (przykład takiego wyszukiwania został zaprezentowany na końcu tego podrozdziału, w części z podstawowymi zapytaniami Selenium Webdrivera). Zaletą rozwiązania Selenium jest możliwość korzystania z różnego typu selektorów, takich jak np. xpath (ang. XML Path Language), table_name, css_selector czy id.

Warunkiem koniecznym do ekstrakcji danych z konkretnej strony przy pomocy Selenium jest uprzednie przeanalizowanie jej struktury w celu poprawnego przygotowania scrapera. Jeżeli przy próbie dostania się do pożądanego miejsca występuje konieczność podania danych do logowania lub wypełnienia formularza, można stworzyć osobny plik zawierający wymagane dane (tzw. Input file) (Gojare, Joshi, Gaigaware, 2015).

Selenium IDE to zintegrowane środowisko deweloperskie, które udostępniane jest w postaci wtyczki do przeglądarek Mozilla Firefox oraz Google Chrome. Służy do testowania aplikacji webowych i stron internetowych. Nagrywa i zapisuje czynności wykonywane przez użytkownika (np. kliknięcia w dane przyciski, wprowadzanie znaków w pola do tego przeznaczone itp.). Z zapisanych w odpowiedniej sekwencji czynności na stronie można utworzyć „projekt", który jest odtwarzalny w dowolnym momencie. Wtyczka, po nagraniu sekwencji akcji użytkownika w konkretnym projekcie, umożliwia ich późniejsze odtworzenie, aby sprawdzić, czy aplikacja/strona działa poprawnie.

Selenium Grid jest narzędziem stworzonym do wykonywania testów rozproszonych. Oznacza to, że umożliwia ono przeprowadzanie testów na różnych maszynach oraz na różnych przeglądarkach równolegle. Powstało ono w celu skrócenia czasu przeprowadzanych testów. Dla przykładu, dzieląc zbiór różnych testów na dwie maszyny, czas wykonywania testów powinien skrócić się w przybliżeniu o połowę.

Głównym atutem Selenium jest niezależność wszystkich komponentów. Oznacza to, że korzystanie z jednego nie wymaga od użytkownika użycia pozostałych.

Tworzenie scrapera w Selenium zostanie przedstawione na przykładzie pozyskiwania cytatów ze strony internetowej BrainyQuote (ten sam przypadek, co w rozdziale 4.4.3, ale zamiast selektora css zostanie użyty selektor xpath).

Na początku (linia 1, rysunek 22) następuje import biblioteki Selenium.webdriver, aby można było z niej skorzystać w celu stworzenia scrapera. Następnie należy określić adres strony, z której mają zostać pozyskane dane. W tym przykładzie adres został przypisany do zmiennej start_url. Kolejno, konieczne jest określenie ścieżki, w której znajduje się webdriver. W tym przypadku jest to pulpit użytkownika o nazwie Admin. Zapisany plik ma nazwę chromedriver.exe, ponieważ zostanie on wykorzystany przez przeglądarkę Google Chrome (dla każdej przeglądarki należy pobrać osobny plik driver).

```
1  from selenium import webdriver
2
3  start_url = 'https://www.brainyquote.com/topics/scrape-quotes'
4
5  scraper = webdriver.Chrome("C:/Users/admin/Desktop/chromedriver.exe")
6  scraper.get(start_url)
7
8  def parse():
9
10     box = scraper.find_element_by_xpath('//*[@id="qpos_1_19"]/div/div[1]/div')
11
12     elements_in_box = box.find_elements_by_css_selector('a')
13
14     yield{ 'text': elements_in_box[0].text, 'author': elements_in_box[1].text}
```

Rysunek 22. Przykład wykorzystania narzędzia Selenium.
Źródło: opracowanie własne.

W linii 6 adres strony zostaje przekazany scraperowi. Wywołanie atrybutu get skutkuje otworzeniem w przeglądarce Google Chrome adresu strony zapisanego w zmiennej start_url. W linii 8 zostaje stworzona funkcja parse(). Dzięki niej dane mogą zostać poddane ekstrakcji. Na początku funkcji (linia 10) scraper odnajduje konkretny element na stronie za pomocą selektora xpath, wskazującego miejsce ekstrahowanego elementu w kodzie źródłowym strony. W tym przypadku będzie to cytat ze strony oraz jego autor. Odnaleziony element zostaje przypisany do zmiennej box. Aby móc wydzielić szczegółowe informacje, scraper przypisuje je do zmiennej elements_in_box, rozróżniając każdy element za pomocą tagu „a". Zmienna elements_in_box składa się w podanym przykładzie z dwuelementowej listy, której pierwszym elementem jest treść cytatu, a drugim jego autor. Przykładowy wynik odczytania pobranych elementów w formacie JSON wygląda następująco:

{ „text": „I'll play until they have to scrape me off the stage.", „author": „James Young"}

4.5 Porównanie narzędzi

Wybierając odpowiednie narzędzie do automatycznego pobierania danych, podmiot sektora bankowego powinien dokładnie przeanalizować swoje potrzeby. Można wyróżnić kilka aspektów, które mają determinujący wpływ na wybór właściwego narzędzia.

Do takich czynników można zaliczyć:

- ograniczenia (takie jak skala pozyskiwanych danych, wskazane języki programowania),
- stopień trudności opanowania narzędzia (przyjazność dokumentacji, dostępność tutoriali i stron z pomocą techniczną),
- stopień rozwinięcia narzędzia, tj. czy posiada funkcjonalności pozwalające na ominięcie ograniczeń technicznych stawianych przez twórców witryn,
- typ licencji, który determinuje koszty, jakie będą musiały zostać poniesione w celu pozyskania interesujących danych,
- popularność narzędzia wśród deweloperów, co przekłada się na koszt pozyskania programistów i ich dostępność.

Pierwszym, podstawowym ograniczeniem, jakie można wyróżnić, jest skala pozyskiwanych danych. Jest to jedna z ważniejszych kwestii, ponieważ dla każdego narzędzia wskazana jest ilości danych, które może ono przetworzyć w określonym przedziale czasowym. Wśród opisanych w tym rozdziale narzędzi znajdują się takie, które lepiej jest zastosować przy procesach pozyskujących duże ilości danych (Scrapy), jak i takie, które korzystniej użyć przy pobieraniu mniejszych ilości danych (Beautiful Soup 4). Szczególne narzędzie w tym aspekcie to Selenium, które jest głównie nakierowane na pobieranie danych w specyficznych sytuacjach, np. pobieranie danych ze stron o rozbudowanej strukturze czy też dużego zasobu danych rozmieszonych na kilku podstronach danej witryny. Nie jest w tym przypadku istotne, na jaką skalę Selenium ma zostać być użyte.

Trzy z omówionych narzędzi są przystosowane do wykorzystania w języku Python (Beautiful Soup 4, Scrapy, Selenium), natomiast Rvest przeznaczony jest dla języka R. Warto wspomnieć, że największą liczbą dostępnych języków charakteryzuje się Selenium, jest ich aż 6 (Java, C#, Python, PHP, Pearl, Ruby). Kryterium to jest istotne, jeżeli opracowane narzędzie miałoby być ściśle zintegrowane z istniejącą infrastrukturą podmiotu sektora bankowego. Jeżeli jednak zebrane dane miałyby być udostępniane innym komponentom infrastruktury poprzez API, język, w którym jest stworzone, nie ma znaczenia.

Kolejnym fundamentalnym zagadnieniem jest stopień trudności opanowania narzędzia. Dla początkujących użytkowników trafnym wyborem jest zastosowanie Beautiful Soup 4 lub Rvest ze względu na relatywnie prostą implementację. Dla użytkowników średnio zaawansowanych dobrym wyborem będzie zastosowanie Selenium, zaś zaawansowani, oprócz wymienionych wcześniej narzędzi, mogą zdecydować się także na użycie Scrapy. Istotne jest więc, czy podmiot sektora bankowego zleca stworzenie narzędzia zewnętrznej firmie, czy dysponuje własnymi zasobami do jego opracowania oraz kto będzie utrzymywał narzędzie. Jeżeli wszystkie prace będą zlecone zewnętrznej firmie, to trudność opanowania narzędzia nie ma dużego znaczenia. Natomiast jeżeli takie rozwiązanie miałoby być opracowane własnymi zasobami ludzkimi, konieczne jest uzależnienie wyboru od wiedzy i doświadczenia pracowników.

Kryterium, które jest często rozpatrywane przy wyborze narzędzi informatycznych, jest efektywność, rozumiana jako stosunek nakładów pracy poniesionych na opracowanie (lub wdrożenie) narzędzia i dokładności uzyskanych wyników. W przypadku narzędzi automatycznego pozyskiwania danych, nie da się określić efektywności przed wdrożeniem, ponieważ zależy ona od specyfiki źródła, wykorzystanych w nim technologii, zastosowanych środków ograniczających automatyczne przetwarzanie itp.

Narzędzia do automatycznego pozyskiwania danych z Internetu są intensywnie rozwijane. Pojawiają się również próby wykorzystania AI (sztucznej inteligencji) oraz metod uczenia maszynowego do zautomatyzowania działania crawlerów, np. (Jiang i in., 2010), (Cui i in., 2018), (Meegahapola i in., 2018), jednak nie są to obecnie rozwiązania szeroko stosowane, choć stanowią obiecujący kierunek badań.

Szczegółowe porównania, uwzględniające pozostałe czynniki, zostały zaprezentowane w Tabeli 8.

Reasumując, wybór odpowiedniego narzędzia nie należy do najłatwiejszych zadań. Każde z opisanych narzędzi ma swoje mocne, jak i słabe strony. Jednakże wnikliwa analiza kluczowych czynników powinna ugruntować potrzeby podmiotów sektora bakowego i ułatwić im wybór.

Tabela 8. Porównanie narzędzi do automatycznego pobierania danych.

	Beautiful Soup 4	Scrapy	Rvest	Selenium
Typ	Pakiet języka programowania Python	Wysokopoziomowa platforma programistyczna (framework)	Pakiet języka programowania R	IDE – wtyczka do przeglądarki WebDriver-web scraper Grid-centurm zarządzania
Licencja	Open Source	Open Source	Open Source	Open Source
Języki programowania	Python	Python	R	Java, C#, Python, PHP, Pearl, Ruby
Łatwość zastosowania	Początkujący użytkownicy	Zaawansowani użytkownicy	Początkujący użytkownicy	Średnio zaawansowani użytkownicy
Skala pozyskiwanych danych	Małe ilości	Małe i duże ilości (łatwość przetwarzania Big Data)	Dowolny wolumen danych	Problemy o specyficznej charakterystyce
Zalety	– wielowątkowość współbieżna – łatwość zastosowania – rozwinięta społeczność korzystająca z technologii	– obsługa współbieżności i programowania asynchronicznego – możliwość łączenia z API lub używania jako ogólny crawler – mniejsze zużycie pamięci i procesora	– możliwość łatwego oczyszczenia danych przez użycie pakietów R	– niezależność narzędzi – możliwość zastosowania w przeglądarkach: Mozilla Firefox, Google Chrome, Internet Explorer, Firefox, Opera, Safari
Wady (ograniczenia)	– może okazać się zbyt powolne do pozyskiwania dużych ilości danych o skomplikowanej strukturze	– dokumentacja nie jest przyjazna dla początkujących użytkowników	– połączenie kilku pakietów programistycznych może okazać się zbyt trudne dla początkujących użytkowników	– brak możliwości raportowania i dokumentowania błędów przy testowaniu
Ekstrakcja danych ze źródeł	HTML; XML	HTML	HTML, XML	HTML
Selektory	lxml	XPath, CSS	read_html, html_nodes, html_text, html_tabl	XPath, CSS_selector, table_name, id

Źródło: opracowanie własne.

ROZDZIAŁ V

Przypadki użycia

5.1 Pobieranie danych ze strony HTML – przykład 1

Przygotowanie do pobierania danych w sposób zautomatyzowany powinno obejmować analizę wszystkich czynników, które zostały opisane w raporcie, począwszy od uwarunkowań prawnych, którymi objęte są dane, po dobór narzędzia do scprapowania, biorąc pod uwagę specyfikację strony.

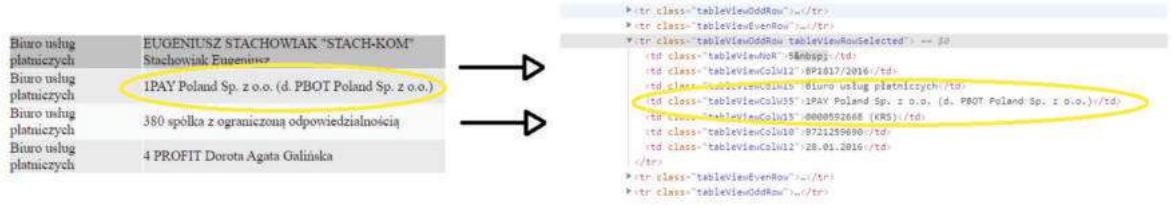
Przykładem strony o specyficznej budowie w sektorze bankowym może być rejestr podmiotów usług płatniczych na stronie KNF (KNF, 2019). Przyglądając się jej bliżej można zauważyć, że jest to tabela zawierająca 1438 rekordy podzielone na 144 strony (stan na dzień 25.11.2019 r.), przy czym URL jest niezmienny bez względu na stronę tabeli (jest to przykład źródła Internetu głębokiego, tj. poszczególne podstrony rejestru nie są indeksowane przez wyszukiwarki). Ponieważ URL jest stały, jedynym sposobem nawigowania między podstronami jest w tym przypadku naciśnięcie przycisku przejścia do następnej strony (co zostało zaznaczone na rysunku 23.).

Taki przycisk znajduje się najczęściej u góry lub u dołu tabeli. Jak widać w przykładzie, tak jest także w tym przypadku. Po przeprowadzonej analizie, najlepszym rozwiązaniem w tym przypadku byłoby użycie narzędzia Selenium WebDriver z zestawu narzędzi Selenium, gdyż to właśnie ono umożliwia poruszanie się za pomocą kliknięć w obiekty. Jest to typowy przykład strony, gdzie to właśnie Selenium wydaje się rozwiązaniem lepszym niż konkurencyjne narzędzia.

Po przeanalizowaniu budowy strony i wyborze narzędzia, można przystąpić do ekstrakcji pożądanych informacji. Każda informacja zawarta na stronie ma swoje miejsce w hierarchii strony, opisanej w kodzie źródłowym. To właśnie te miejsca w hierarchii należy wskazać narzędziu, określając przy tym reguły, które informacje powinny zostać pobrane, a które pominięte.

Reguły pozwalające pobrać jedynie interesujące użytkownika dane nie są zazwyczaj oczywiste i zależą zarówno od wykorzystywanego narzędzia, jak i poziomu skomplikowania kodu źródłowego strony. W tym przypadku, aby pobrać rekord zaznaczony na rysunku 24.,

Rysunek 23. Pierwsza strona tabeli Rejestr Usług Płatniczych.
Źródło: (KNF, 2019).



Rysunek 24. Rysunek przedstawiający odzwierciedlenie informacji z tabeli w kodzie źródłowym,
Źródło: (KNF, 2019).

można skorzystać z selektora xpath, jednak odwołanie się do niego wymaga dobrej znajomości hierarchii elementów na stronie. W tym przypadku wyglądałoby to następująco:

//*[@id="maintable"]/tbody/tr[5]/td[4]

co oznacza, że zaznaczony tekst znajduje się w tabeli o id=maintable w sekcji tbody. Tr[5] w tym przypadku oznacza, że jest to 5 wiersz w tabeli, a td[4] odnosi się do kolumny, w której znajduje się interesujący użytkownika tekst. Przy pomocy xpath można pobrać dowolny element strony html, np. całą tabelę. Pobieranie danych z różnych podstron wymaga zaimplementowania reguł nawigacji.

Gdy program scrapujący zakończy pracę, wszystkie informacje powinny zostać zapisane. Forma zapisu informacji zależy od preferencji użytkownika (dla przykładu informacje mogą zostać umieszczone w bazie danych czy też zapisane w postaci pliku csv).

Przykład strony KNF pokazuje, że scraper nie jest uniwersalnym narzędziem. Nawet niewielka zmiana w strukturze strony czyni go bezużytecznym i wymaga modyfikacji (Kaczmarek, 2006, Kowalkiewicz, 2006). Ponadto, aby informacje pobierane z danego źródła były aktualne, należy ponownie przeprowadzić proces scrapowania co jakiś czas, w celu ich weryfikacji i aktualizacji. W przytoczonym przykładzie strony KNF, takie regularne scrapowanie danych okazałoby się niemożliwe, ponieważ w ciągu kwartału struktura strony całkowicie uległa zmianie (rysunek 25.). Scraper nieprzygotowany na taką ewentualność nie byłby w stanie dalej pobierać informacji ze strony, ponieważ kod źródłowy uległ zmianie razem z wyglądem strony.

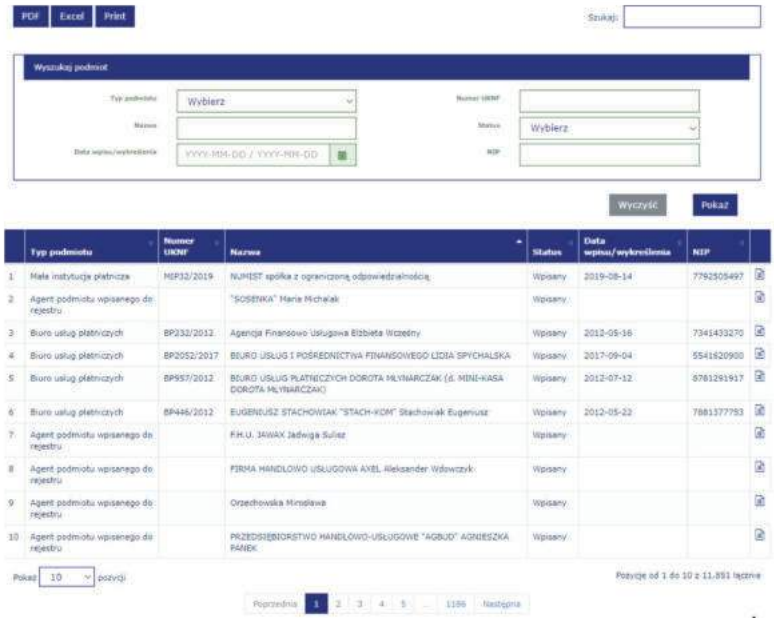
Podobna sytuacja miałaby miejsce, gdyby wygląd strony pozostał niezmienny, ale kod źródłowy zostałby poddany refaktoryzacji. Oznacza to, że aby scraper był skutecznym narzędziem, wymaga utrzymania, a stworzenie jednego narzędzia działającego poprawnie na wielu stronach jest niemalże niemożliwe.

5.2 Pobieranie danych przez API

Jak zostało to już kilkukrotnie zaznaczone w poprzednich rozdziałach, najprzystępniejszą formą pobierania danych jest ich pozyskiwanie poprzez API. Nie powinno się przystępować do procesu scrapingu, jeżeli dostawca danych udostępnia je w tej formie, a format API odpowiada zidentyfikowanym potrzebom informacyjnym (zdarzają się sytuacje, gdy API nie obejmuje wszystkich danych, a scrapowanie nie jest zabronione). W niniejszym rozdziale zostanie przedstawiony przykład pobierania danych z użyciem REST API.

Przed rozpoczęciem pobierania danych należy dokładnie przestudiować dokumentację określonego API, publikowaną i udostępnianą przez właściciela API. Struktura API będzie się różniła w zależności od charakteru danych oraz ich dostawcy.

Współcześnie bardzo dużo instytucji decyduje się na udostępnianie danych przez API. Przykładem takiej



Rysunek 25. Strona Rejestru Usług Płatniczych po zmianie jej struktury.
Źródło: (ERUP 2, 2020)

```
import requests
```

```
response = requests.get(" http://apps.who.int/gho/athena/api/GHO/WHOSIS_000001.json?profile=simple")
```

Rysunek 26. Zdefiniowanie i wysłanie zapytania do API.
Źródło: opracowanie własne.

```
data = response.json()
```

Rysunek 27. Odebranie odpowiedzi z API.
Źródło: opracowanie własne.

```
dzisiejsza_data <- format(Sys.Date(), "%Y%m%d")

euclid_url <- paste0("https://euclid.eba.europa.eu/register/downloads/PSDMD/",dzisiejsza_data,"/download-PSDMD-",dzisiejsza_data,"0000.zip")

download.file(euclid_url,destfile = "./data/data.zip")
```

Rysunek 28. Przykład pobierania danych z plików..
Źródło: opracowanie własne.

instytucji jest World Health Organization. Organizacja udostępnia szereg różnorodnych danych związanych z zadaniami, jakie wykonuje. Między innymi można znaleźć dane na temat (WHO, 2020):


```
- {
  - dim: {
      PUBLISHSTATE: "Published",
      GHO: "Life expectancy at birth (years)",
      COUNTRY: "Poland",
      YEAR: "2005",
      SEX: "Male",
      REGION: "Europe"
    },
    Comments: "WHO life table method: Vital registration",
    Value: "70.8"
  },
- {
  - dim: {
      PUBLISHSTATE: "Published",
      GHO: "Life expectancy at birth (years)",
      REGION: "Europe",
      YEAR: "2007",
      SEX: "Male",
      COUNTRY: "Poland"
    },
    Comments: "WHO life table method: Vital registration",
    Value: "71.0"
  },
- {
  - dim: {
      PUBLISHSTATE: "Published",
      SEX: "Male",
      YEAR: "2014",
      GHO: "Life expectancy at birth (years)",
      COUNTRY: "Poland",
      REGION: "Europe"
    },
    Comments: "WHO life table method: Vital registration",
    Value: "73.6"
  },
```


Rysunek 29. Wynik zapytania do API WHO.
Źródło: opracowanie własne.

- Systemów opieki zdrowotnej;
- Statystyki dotyczące zachorowalności na niektóre choroby (np. malarię, cholerę, globalny nadzór wirusologiczny w kierunku grypy);
- Bezpieczeństwa na drogach.

Takie dane mogą być przydatne np. przy analizie możliwych scenariuszy związanych z występującymi zagrożeniami epidemiologicznymi, np. obserwowanym obecnie koronawirusem.

W nawiązaniu do opisu przedstawionego w rozdziale 4.4.1 API, API można wykorzystać na przykład poprzez program napisany w języku Python. Przykład takiego programu został przedstawiony na rysunku 27. i rysunku 26. Wykonując dwa przedstawione na rysunkach polecenia w wyniku otrzymana zostanie lista w formacie JSON, w której można znaleźć informacje o długości życia według podziału na kontynent, kraj, płeć oraz rok (rysunek 29.).

5.3 Pobieranie danych z udostępnionych plików

Tworzenie scrapera jest zazwyczaj najbardziej czasochłonną i kosztowną metodą na pobranie danych ze źródeł internetowych. Dodatkowo scrapery muszą być stale utrzymywane, w celu wprowadzenia koniecznych zmian, gdy zmieni się struktura strony. Dużo szybszym i łatwiejszym rozwiązaniem jest

pobieranie danych z pliku dostępnego w danym źródle (o ile dana strona udostępnia dane w takiej formie). Poza samą możliwością pobrania danych, istotnym aspektem jest również ich aktualizacja. Przed skorzystaniem z gotowego zbioru danych należy sprawdzić, czy i jak często jest on aktualizowany i dostosować odpowiednio częstotliwość pobierania zaktualizowanych danych.

Kluczowym wyzwaniem w przypadku danych udostępnianych w formie plików jest ich dalsze przetworzenie w celu wyekstrahowania z nich tylko pożądanych informacji. Za przykład może posłużyć udostępniany przez European Banking Authority (EBA) spis instytucji płatniczych (EUCLID) (European Banking Authority, 2020). Spis jest możliwy do pobrania jako skompresowany plik typu JSON. Osoba zainteresowana danymi może pobierać go ręcznie bądź napisać program, który automatycznie będzie pobierał dany plik w określonych interwałach czasowych (np. raz dziennie). By umożliwić regularne pobieranie, konieczne jest przeanalizowanie zmian zachodzących w linku prowadzącym do pliku. W przypadku danych udostępnianych przez EUCLID, jedynym zmieniającym się elementem adresu url jest data w specyficznym formacie. W tym przypadku automatyczne pobieranie danego pliku sprowadza się do 3 linii kodu napisanych w języku R (rysunek 28.).

W pierwszym kroku (linia 5) pobierana jest aktualna data z systemu i modyfikowany jest jej format na odpowiadający formatowi występującemu w linku. Zmodyfikowaną datę przypisujemy do zmiennej dzisiejsza_data. Przykładowo, 26 lutego 2020 roku po dokonaniu przekształcenia będzie zapisany za pomocą 8 liczb: „20200226”. W drugim kroku (linia 7) tworzona jest zmienna, w której zapisywany jest adres URL uwzględniający aktualną datę. W tym celu wykorzystana jest funkcji paste0, która łączy ze sobą poszczególne fragmenty tekstu. Trzeci krok (linia 9) to skorzystanie z funkcji download.file(), za pomocą której pobierany jest plik z zadanego adresu URL. Drugim argumentem funkcji download.file() jest ścieżka do określonego folderu na urządzeniu, na którym plik ma zostać zapisany. Należy pamiętać, że po ostatnim znaku „/” należy podać wybraną przez użytkownika nazwę pliku. Istotne jest również uwzględnienie rozszerzenia pobieranego pliku po kropce. W przykładzie pobierany plik zostanie zapisany w folderze „data” pod nazwą data, a jego rozszerzenie to zip.

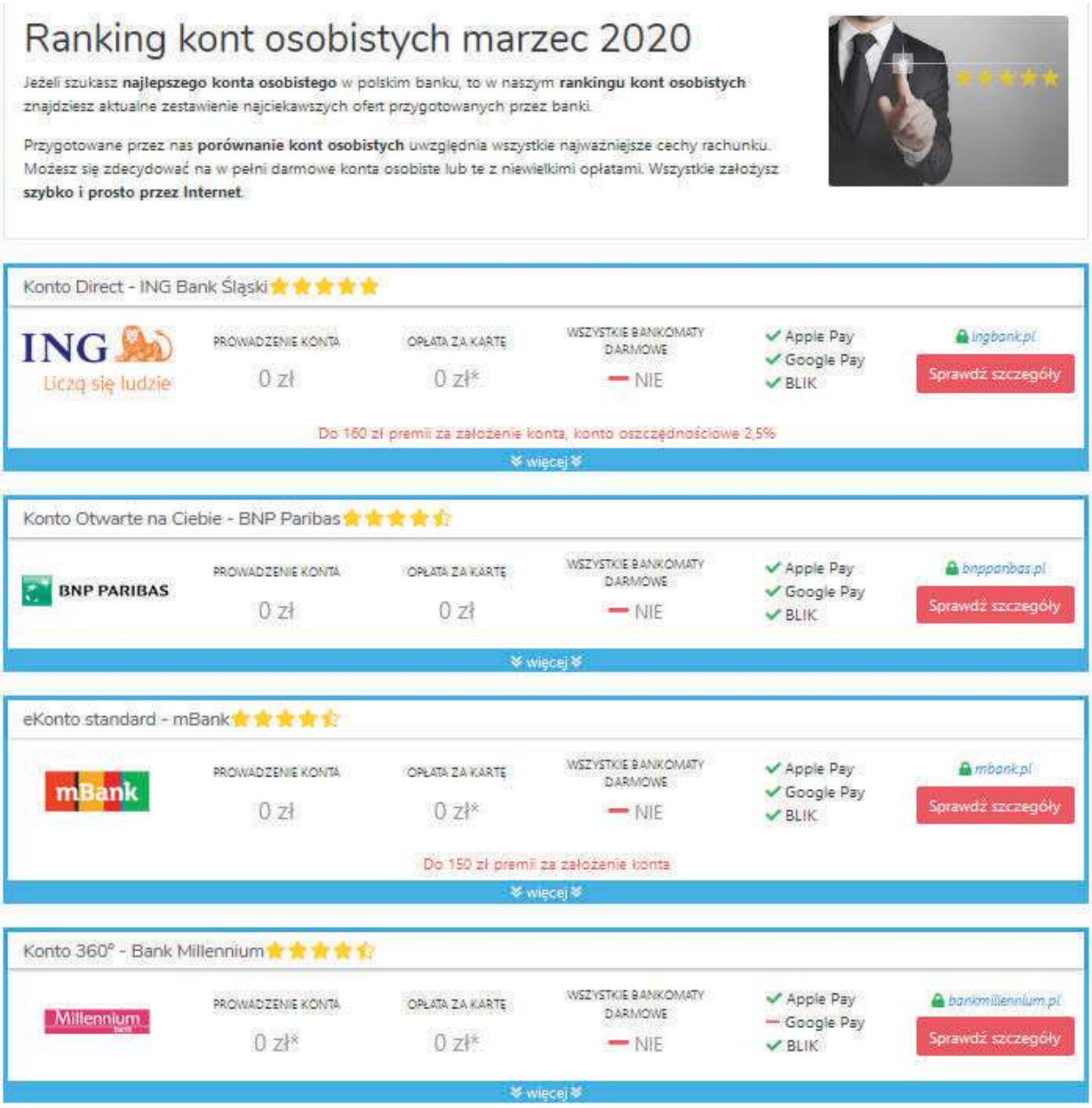
W ramach procesu przetwarzania danych po pobraniu należy rozpakować plik i załadować go do narzędzia, w którym pobrane dane mają zostać dalej przetworzone do pożądanej formy. Wymaga to uprzedniego zapoznania się ze strukturą pliku. W przypadku analizowanego przykładu, na rysunku 30. przedstawionych zostało pierwsze 10 pobranych rekordów w postaci data frame. Data frame, którego fragment przedstawiono na rysunku 30, posiada w sumie 6 kolumn i 228 326 wierszy. Dodatkowym problemem w tym zbiorze danych są zagnieżdżone tabele, znajdujące się w kolumnach services i properties.

5.4 Pobieranie danych ze strony HTML – przykład 2

Kolejnym przykładem pobierania danych ze źródeł internetowych jest strona internetowa odpowiedzialna za sporządzanie comiesięcznych rankingów kont osobistych (rysunek 31.). Zgodnie z opisem zawartym na stronie zestawienie jest aktualizowane, co więcej umożliwia porównywanie między sobą kont na podstawie różnych kryteriów. W prezentowanym

	ca_owner_id	entity_code	entity_type	services	properties	x_eba_entity_version
1	NL_DNB	NL_DNB!F0027	PSD_EMI	*31 variables*	*7 variables*	20200310170952034
2	NL_DNB	NL_DNB!R142613	PSD_EMI	*31 variables*	*8 variables*	20200310170952034
3	BG_BNB	BG_BNB!121554961	PSD_EMI	*31 variables*	*7 variables*	20200304093522567
4	GB_FCA	PSD_EMI!GB_FCA!900989	PSD_EMI	*1 variable*	*8 variables*	20200306184241164
5	GB_FCA	PSD_EMI!GB_FCA!900483	PSD_EMI	*31 variables*	*8 variables*	20200304183757200
6	CZ_CNB	CZ_CNB!24759023	PSD_EMI	*1 variable*	*7 variables*	20200302170342320
7	IT_BI	PSD_EMI!IT_BI!10917120965	PSD_EMI	*1 variable*	*8 variables*	20200302230924556
8	IE_CBI	IE_CBI!C95957	PSD_EMI	*22 variables*	*8 variables*	20200302234920851
9	GB_FCA	PSD_EMI!GB_FCA!900926	PSD_EMI	*30 variables*	*8 variables*	20200303193152100
10	LU_CSSF	LU_CSSF!W00000005	PSD_EMI	*31 variables*	*7 variables*	20200228095158251

Rysunek 30. Pobieranie danych z pliku.
Źródło: opracowanie własne.



Rysunek 31. Przykład strony HTML z informacjami o ofercie banków.
Źródło: https://www.rankingkontosobistych.pl/konta-osobiste.

```
22  url <- "https://www.rankingkontosobistych.pl/konta-osobiste"
23
24  bank_ranking_site <- read_html(url)
```

Rysunek 32. Pobieranie danych przy użyciu rvest (1).
Źródło: opracowanie własne.

niżej przykładzie pobrane zostaną nazwy kont, które według strony są uznawane za najlepsze. W tym celu napisany został program w języku programowania R oraz zastosowano pakiety rvest i tidyverse.

W pierwszym kroku (linia 22, rysunek 32.) wskazany został adres strony, z której mają zostać pobrane dane i przypisany do zmiennej o nazwie url. W drugim kroku (linia 24) utworzona została zmienna zawierająca w sobie strukturę pobieranej strony. W dalszej części przykładu nie ma już konieczności ponownego pobierania strony, co uważane jest za dobrą praktykę, gdyż serwer właściciela strony nie będzie ponownie odpytywany.

```
30  account_name <- bank_ranking_site %>%
31                  html_nodes(xpath = '//*[contains(concat( " ", @class, " " ), concat( " ", "text-dark", " " ))]') %>%
32                  html_text() %>%
33                  as.data.frame() %>%
34                  rename("name" = ".") %>%
35                  na_if("") %>%
36                  drop_na()
37
38  row.names(account_name) <- NULL
```

Rysunek 33. Pobieranie danych przy użyciu rvest (2).
Źródło: opracowanie własne.

Następnie w kroku trzecim (linia 30, rysunek 33.) działania wykonywane są na obiekcie bank_ranking_site. Cała wykonana operacja zostanie przypisana do obiektu account_name. Korzystając z funkcji html_nodes() (linia 31) i na podstawie selektora xpath pobierana jest pożądana informacja. Funkcja html_text() (linia 32) służy wydobyciu nazw kont, które były celem wybranego selektora xpath. W liniach 33 – 36 i 38 dokonywane są operacje porządkujące, które umożliwiają przedstawienie ostatecznego wyniku w postaci uporządkowanej tabeli (rysunek 34. i rysunek 35.).

Kolumna date została utworzona na podstawie nagłówka i pozwala na odróżnienie od siebie kolejnych edycji rankingu. Dodatkowo umożliwia proste grupowanie. Kolumny acc_conducting i card_cost są przedstawione jako dane tekstowe, w celu podkreślenia, że koszty zakładania kont i posiadania kart nie są w pełni darmowe. W ramach kolumny

	name
1.	Konto Direct–ING Bank Śląski
2.	Konto Otwarte na Ciebie–BNP Paribas
3.	eKonto standard–mBank
4.	Konto 360°–Bank Millennium
5.	Konto Przekorzystne–Bank Pekao
6.	Konto Jakie Chcę–Santander Bank Polska
7.	Konto Proste Zasady–Getin Bank
8.	Konto dla Ciebie–Credit Agricole
9.	Konto za Zero–PKO BP
10.	Konto Freemium–T-Mobile Usługi Bankowe
11.	Konto bez Granic–PKO BP

Rysunek 34. Pobieranie danych przy użyciu rvest (3).
Źródło: opracowanie własne.

	name	date	acc_conducting	card_cost	free_atm	star–score	apple–pay	google–pay	blik	promotion	site
1.	Konto Direct–ING Bank Śląski	2020-03-01	0 zł	0 zł*	nie	5.0	tak	tak	tak	Do 160 zł premii za założenie konta, konto oszczędnościowe 2,5%	ing.bank.pl
2.	Konto Otwarte na Ciebie–BNP Paribas	2020-03-01	0 zł	0 zł	nie	4.5	tak	tak	tak	Brak promocji	bnpparibas.pl
3.	eKonto standard–mBank	2020-03-01	0 zł	0 zł*	nie	4.5	tak	tak	tak	Nawet 200 zł za założenie konta, 3% na koncie oszczędnościowym przez 6 miesięcy	mbank.pl
4.	Konto 360°–Bank Millennium	2020-03-01	0 zł*	0 zł*	nie	4.5	tak	nie	tak	Brak promocji	bankmillennium.pl
5.	Konto Przekorzystne–Bank Pekao	2020-03-01	0 zł*	0 zł*	nie	4.0	tak	nie	tak	Dodatkowe promocyjne konto oszczędnościowe – 3,5%	pekao.com.pl
6.	Konto Jakie Chcę–Santander Bank Polska	2020-03-01	0 zł	0 zł*	nie	4.5	tak	tak	tak	Do 300 zł premii do wyboru w promocji	santander.pl
7.	Konto Proste Zasady–Getin Bank	2020-03-01	0 zł*	0 zł	tak	4.0	tak	tak	tak	Do 160 zł premii za założenie konta, konto oszczędnościowe 2,5%	getinbank.pl
8.	Konto dla Ciebie–Credit Agricole	2020-03-01	0 zł*	0 zł*	tak	4.0	tak	tak	tak	Do 150 zł premii za założenie konta	credit-agricole.pl
9.	Konto za Zero–PKO BP	2020-03-01	0 zł*	0 zł*	nie	4.0	tak	nie	tak	Brak promocji	pkobp.pl
10.	Konto Freemium–T-Mobile Usługi Bankowe	2020-03-01	0 zł	0 zł*	tak	4.0	tak	tak	tak	Brak promocji	t-mobilebankowe.pl
11.	Konto bez Granic–PKO BP	2020-03-01	17,9 zł*	0 zł	tak	3.5	tak	nie	tak	Brak promocji	pkobp.pl

Rysunek 35. Pobieranie danych przy użyciu rvest (4).
Źródło: opracowanie własne.

promocje pobierane są informacje wyróżnione przez twórcę rankingu, niektóre konta takowych informacji nie posiadają. Informację można by rozszerzyć o rozwijany element na stronie (przycisk „więcej"), jednak by to zrobić konieczne jest zastosowanie narzędzia Selenium.

Wnioski i rekomendacje

Analiza potrzeb informacyjnych sektora bankowego, związanych z podejmowaniem decyzji biznesowych oraz dotyczących cyberbezpieczeństwa pokazuje, że do ich zaspokojenia konieczne jest bieżące pozyskiwanie zróżnicowanych danych, pochodzących z różnych źródeł internetowych. Duża część takich danych jest powszechnie dostępna, jednak ich format powoduje, że są one łatwo zrozumiałe dla człowieka, ale trudne do automatycznego przetwarzania przez maszyny. Podstawowym problemem jest ekstrakcja oraz agregacja takich danych.

Proces pozyskiwania danych ze źródeł internetowych może przebiegać na kilka różnych sposobów. Po pierwsze, jeżeli dane są udostępniane poprzez dedykowany interfejs (API), to możliwe jest pobieranie danych o strukturze zdefiniowanej przez wystawcę API. Zaletą API jest niewątpliwie łatwość integracji z wewnętrznymi systemami i bazami danych. Za wadę, z punktu widzenia podmiotu pobierającego dane, można uznać, że wystawca API decyduje o zakresie i formie udostępnianych danych (np. może ograniczyć zbiór czasowo lub geograficznie), API nie musi też obejmować wszystkich danych z bazy. Utrzymanie API jest kosztem dla właściciela danych, ale pozwala kontrolować kto, kiedy i jakie dane pobiera. Umożliwia również szacowanie ruchu sieciowego, który jest z tym związany.

Drugą metodą jest udostępnienie danych w formie plików, np. csv lub pdf. W takim wypadku format pliku pełni kluczową rolę przy dalszym przetwarzaniu pobranych danych. Dane z plików csv są najczęściej ustrukturyzowane, a tym samym możliwe do przetworzenia przez większość dostępnych narzędzi (choć istotną rolę odgrywa też jakość samych danych, poprawność i spójność stosowanych formatów, jednolite kodowanie znaków, itp.). Pliki w innych formatach, np. pdf, mogą mieć postać nieustrukturyzowaną.

Ostatnim sposobem jest pozyskiwanie danych opublikowanych bezpośrednio na stronach internetowych. Taki proces nazywany jest *web scrapingiem*. *Web scraping* pozwala na przetwarzanie dowolnych treści z dokumentów HTML. Strony wykonane w innych technologiach (np. JavaScript lub mało popularny w ostatnich latach Flash) w znacznym stopniu utrudniają lub całkowicie uniemożliwiają pobieranie treści.

Najważniejsze zidentyfikowane źródła danych dla sektora bankowego, to: strony domowe banków, rankingi usług bankowych, fora i komentarze w mediach społecznościowych, strony prowadzone przez instytucje z sektora bankowego oraz portale branżowe. Źródła te różnią się strukturą danych, podmiotem, który generuje dane oraz zasięgiem. Analiza danych w źródłach wykazała, że większość potrzeb informacyjnych można (w różnym stopniu) zaspokoić danymi z różnych typów źródeł. Z tego względu, przed przystąpieniem do tworzenia narzędzia pobierającego dane, należy przeprowadzić dokładną analizę potrzeb informacyjnych oraz identyfikację i analizę źródeł, związanych z określonym problemem biznesowym.

W raporcie przedstawiono cztery narzędzia/pakiety programistyczne, które służą do budowy *web scraperów*: Selenium, rvest, Beautiful Soup 4 oraz Scrapy. Rozwiązania te różnią się językiem programowania, dla którego są przeznaczone, wsparciem dla użytkowników, łatwością użycia, zakresem funkcji. Wszystkie przedstawione narzędzia są bezpłatne, ale czas potrzebny na zaprogramowanie *web scrapera* zależy od doświadczenia programisty oraz skomplikowania strony, z której mają być pobierane dane, co powoduje trudność w oszacowaniu średnich kosztów stworzenia takiego narzędzia.

Istotną przeszkodę techniczną w procesie automatycznego pobierania danych z Internetu stanowi zróżnicowanie w budowie stron www. Powoduje ono, że nie da się napisać „uniwersalnego" *web scrapera*, ponieważ reguły ekstrakcji danych zawierają konkretne nazwy znaczników html, nazwy atrybutów, ścieżki prowadzące do wybranego obiektu. Elementy te różnią się na różnych stronach. Co więcej, zmieniają się na skutek wprowadzania zmian na stronach internetowych. Na przykład, dodanie nowej zakładki, zmiana struktury menu czy nawet wyglądu strony mogą pociągać za sobą konieczność aktualizacji reguł ekstrakcji oraz wzorców nawigowania po stronach przez opracowane *web scrapery*.

Choć technicznie możliwe jest przetworzenie i pobranie danych z większości stron, które są dostępne w Internecie, to należy pamiętać, że istnieją ograniczenia prawne i etyczne. Dane opublikowane na stronie internetowej są własnością jej twórcy, który może ograniczyć do nich dostęp, np. narzucając określone licencje lub zabraniając dalszego przetwarzaniach tych danych. Co więcej, część treści na stronach podlega ustawie o prawie autorskim, jako że mogą stanowić dzieła w rozumieniu tejże ustawy. W związku z tym, pierwszym wyborem powinny być źródła na tzw. otwartych licencjach (Open Data), które pozwalają na darmowe, również komercyjne ich wykorzystanie. Tzw. źródła zamknięte, np. artykuły ze stron internetowych, powinny być przetwarzane z poszanowaniem obowiązujących licencji.

Pomimo licznych trudności natury prawnej i technicznej, web scrapery są obecnie wykorzystywane w finansach i bankowości. Należy jednak podkreślić, że przykłady zastosowań można znaleźć raczej w nieoficjalnych materiałach (np. artykułach branżowych,

postach na blogach, dyskusjach na forach technicznych) niż w materiałach udostępnianych przez same banki. Wynika to z faktu, że tego typu rozwiązania stosowane są na wewnętrzne potrzeby instytucji finansowych i mogą stanowić przewagę konkurencyjną. Przykładowe zastosowania w sektorze bankowym, to:

- Wykrywanie trendów na potrzeby wspomagania podejmowania decyzji inwestycyjnych i związanych z zarządzaniem aktywami. Przykładowo, długoterminowa agregacja danych o wynikach stron www należących do określonego rynku, może umożliwić odkrycie trendów na tym rynku. Monitorowanie cenników i stanów magazynowych w e-sklepach partnerów lub klientów pozwala na wsparcie podejmowania decyzji inwestycyjnych.

- Ocena ratingowa. Gromadzenie informacji na temat wielu różnych podmiotów, z wielu stron na raz, może dostarczyć w czasie rzeczywistym kompleksowych danych, które zasilą silniki analityczne. Zgromadzone dane mogą być również przydatne w analizie zdolności kredytowej przedsiębiorstw.

- Ograniczanie ryzyka. Szybka reakcja na zmieniające się uwarunkowania w otoczeniu (prawnym, społecznym, ekonomicznym) wymaga bieżącego monitorowania informacji publikowanych na stronach rządowych, instytucji publicznych i samorządowych. Automatyzacja procesu monitorowania tych źródeł jest możliwa dzięki narzędziom web crawlingu.

- Analiza nastrojów rynkowych. Opinie społeczne o obiektach, podmiotach i zjawiskach są publikowane w mediach społecznościowych. Agregacja postów na wybrany temat pozwala na zidentyfikowanie zmieniających się nastrojów społecznych i rynkowych.

Proces automatycznego pozyskiwania danych z Internetu może być trudny do przeprowadzenia ze względu na koszty budowy i utrzymania narzędzi, występujące ograniczenia prawne i technologiczne oraz niekiedy konieczność zakupu dodatkowych licencji. Z tego względu, warto rozważyć budowę rozwiązań z tego zakresu i oferowanie ich klientom w modelu 'Software as a Service'. Dla klientów, np. banków, takie rozwiązanie pozwoliłoby na dostęp do zebranych i wstępnie przetworzonych (np. zagregowanych) danych, bez konieczności budowania rozwiązań od zera, a jedynie ponosząc jasno zdefiniowane koszty dostępu do usługi. Na poziomie sektora pozwoliłoby to uniknąć budowania takich samych lub podobnych narzędzi przez każdy podmiot niezależnie. Scrapowanie danych przez jeden podmiot jest też mniejszym obciążeniem dla serwerów, na których znajdują się źródła. Dostawca rozwiązania oferowałby odpłatnie produkt, zbudowany na podstawie zgłaszanych przez podmioty z sektora potrzeb.

Pod względem funkcjonalnym, narzędzie powinno pobierać i przechowywać dane z różnych źródeł, a następnie dawać użytkownikom możliwość przeszukiwania ich i agregacji na różnych poziomach, np. wyszukiwanie danych o wybranym obiekcie, pobranych z wielu różnych źródeł lub przeszukiwanie danych z konkretnego źródła. Użytkownicy mogliby przeszukiwać zgromadzone dane poprzez tworzenie własnych zapytań, jednak dostawca usługi mógłby też udostępniać gotowe raporty, np. dotyczące trendów na rynku lub kondycji całego sektora. W ramach usług uzupełniających, dostawca rozwiązania mógłby oferować opracowywanie raportów na indywidualne zamówienie klientów.

Proponowane rozwiązanie pozwoliłoby na uzyskanie następujących korzyści. Użytkownicy otrzymaliby możliwość dostępu do wielu różnych źródeł danych poprzez jeden, ujednolicony interfejs, co zwiększa wygodę użytkowania, pozwala na zmniejszenie kosztów integracji narzędzia z wykorzystywaną infrastrukturą oraz zastąpienie kosztów wdrożenia kosztami licencji lub abonamentu. Koszty utrzymania narzędzia i jego rozwoju ponosiłby dostawca usługi – powinny być one rekompensowane przychodami z licencji lub abonamentów uiszczanych przez użytkowników usługi. W takim ujęciu, koszty budowy i utrzymania rozwiązania zostałyby rozdzielone na wielu użytkowników, co umożliwiłoby uzyskanie efektów skali.

Integracja danych z wielu źródeł pozwala również na zwiększenie jakości posiadanych zbiorów danych, np. dzięki weryfikacji danych w różnych źródłach lub możliwość uzupełnienia luk występujących w jednym źródle przez dane z inne.

Istotnym jest, aby wybór źródeł oraz zakresu przechowywanych w bazie i dostarczanych w formie usługi danych, odpowiadał potrzebom informacyjnym potencjalnych klientów (tj. banków). Potrzeby informacyjne, zidentyfikowane na podstawie analizy literatury, zostały przedstawione w rozdziale 2, jednak przed przystąpieniem do implementacji należałoby przeprowadzić pogłębioną analizę potrzeb informacyjnych banków. Pozwoli ona na określenie priorytetów i kolejności dodawania źródeł do usługi.

Wnioski i rekomendacje

- Ze względu na fakt, że narzędzia do automatycznego pobierania danych ze źródeł internetowych są wysoce personalizowane (ściśle dostosowane do wybranego źródła), decyzję o stworzeniu

takiego narzędzia powinna poprzedzać wnikliwa analiza potrzeb informacyjnych oraz struktury, charakterystyk i jakości źródeł, które mogą te potrzeby zaspokoić.

- Przegląd źródeł danych wykazał, że wiele ze źródeł związanych z sektorem bankowym ma postać nieustrukturyzowaną lub półstrukturyzowaną, co utrudnia ich automatyczne przetwarzanie.

- Źródła związane z cyberbezpieczeństwem zawierają najczęściej dane nieustrukturyzowane: są to głównie newsy na portalach branżowych lub komentarze na forach i w mediach społecznościowych. W związku z tym, konieczna wydaje się potrzeba stworzenia bazy zagrożeń związanych z sektorem bankowym (podobnej do baz gromadzących luki bezpieczeństwa w oprogramowaniu, np. https://nvd.nist.gov/vuln/search). Zasadne wydaje się, aby takie rozwiązanie było utworzone dla całego sektora, gdyż zaufanie do niego jest uzależnione od zaufania klientów do pojedynczych banków i instytucji finansowych, zatem zapewnienie środków cyberochrony jest w interesie wszystkich uczestników sektora.

- Agregacja danych pochodzących z różnych źródeł może umożliwić podniesienie jakości zgromadzonych już danych (np. dzięki weryfikacji tych danych w różnych źródłach lub uzupełnianiu brakujących elementów).

- Kluczowym ograniczeniem automatycznego pobierania danych z Internetu są przepisy prawa i licencje obejmujące źródła.

- Na rynku dostępnych jest wiele narzędzi służących do budowania *web scraperów*. Są to często narzędzia darmowe. Wybór konkretnego uwarunkowany jest (1) posiadanymi zasobami technicznymi i ludzkimi, (2) rodzajem i strukturą źródła, z którego mają być pobierane dane.

- Koszty opracowania narzędzia do automatycznego pozyskiwania danych są trudne do oszacowania, gdyż zależą od (1) stawek programistów na rynku, (2) dostępności programistów znających wybrane technologie, (3) złożoności i struktury źródła, (4) zakresu pobieranych danych. Dodatkowo, konieczne jest uwzględnienie kosztów utrzymania narzędzia, tj. aktualizacji reguł ekstrakcji. Częstość tych aktualizacji zależy od tego, jak często źródło ulega zmianie.

- Ponieważ koszty opracowania i utrzymania narzędzi do automatycznego pobierania danych z wielu źródeł będą rosły szybciej niż liniowo przy dodawaniu kolejnych źródeł (ze względu na konieczność integracji danych z różnych źródeł pomiędzy sobą), może być zasadne stworzenie narzędzia służącego do pozyskiwania, agregacji i dostarczania danych różnym podmiotom z sektora bankowego. Pozwoliłoby to na uzyskanie efektów skali.

Bibliografia

Abramowicz, W. (2008). Filtrowanie informacji. Wydawnictwo AE w Poznaniu

Aho, A., Sethi. R. i Ullman, J. (2002). Kompilatory Reguły, metody i narzędzia. Wydawnictwa Naukowo Techniczne.

Akhter, S. i Roberts, J. (2006). Multi-core programming (Vol. 33). Hillsboro: Intel press.

Allen, G. N., Burk, D. L., Davis, G. B. (2006). Academic data collection in electronic environments: Defining acceptable use of internet resources. MIS Quarterly: Management Information Systems, 30(3), 599-610.

Becchi, M., Crowley, P. (2008) Efficient regular expression evaluation: theory to practice. In Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS '08). Association for Computing Machinery, New York, NY, USA, 50–59. DOI:https://doi.org/10.1145/1477942.1477950

Bhushan, B. i Kumar, N. (2012). Surfacing deep web behind scripting languages. Int. J. Comput. Sci. Netw. Secur, 12, 55-58. Pobrane z: https://scholar.google.com/scholar_lookup?title=Surfacing+deep+web+behind+scripting+languages&author=Bhushan,+B.&author=Kumar,+N.&publication_year=2012&journal=Int.+J.+Comput.+Sci.+Netw.+Secur.&volume=12&pages=55%E2%80%9358. (dostęp: 08.01.2020)

Biehl, M. (2015). Architectural Styles for APIs: SOAP, REST and RPC. Pobrane z https://medium.com/api-university/architectural-styles-for-apis-soap-rest-and-rpc-9f040aa270fa (27.02.2020)

Boehmke, B. (2016). Data Wrangling with R. Dayton: Springer

Bradley, A. i James, R. (2019). Web Scraping Using R. Advances in Methods and Practices in Psychological Science, 2(3), 264-270. doi: 10.1177/2515245919859535 (dostęp: 06.02.2020)

BrainyQuote. (2020). Scrape Quotes. Pobrane z: https://www.brainyquote.com/topics/scrape-quotes (dostęp: 06.03.2020)

Buna, S. (2016). Learning GraphQL and relay. Packt Publishing Ltd.

Chatterjee, S. & Nath, A. (2017). Auto-Explore the Web – Web Crawler. International Journal of Innovative Research in Computer and Communication Engineering, 5, 6607-6618. Doi: 10.15680/IJIRCCE.2017.0504006

Clarke, L. (2018) What is the difference between the dark web and the deep web? Opublikowane online, Techworld by IDG, https://www.techworld.com/data/what-is-difference-between-dark-web-deep-web-3680293/ (dostęp: 10.03.2020)

Cui, Y., Sparkman, C., Tsang Lee, H., Loguinov, D. (2018) Unsupervised Domain Ranking in Large-Scale Web Crawls. ACM Trans. Web 12, 4, Article 26 (September 2018), 29 pages. DOI:https://doi.org/10.1145/3182180

Cunha, T., Soares, C., de Carvalho, A. (2018) A label ranking approach for selecting rankings of collaborative filtering algorithms. In Proceedings of the 33rd Annual ACM Symposium on Applied Computing (SAC '18). Association for Computing Machinery, New York, NY, USA, 1393–1395. DOI:https://doi.org/10.1145/3167132.3167418

Esichaikul, V., Phumdontree, C. (2018). Sentiment Analysis of Thai Financial News. In Proceedings of the 2018 2nd International Conference on Software and e-Business (ICSEB '18). Association for Computing Machinery, New York, NY, USA, 39–43. DOI:https://doi.org/10.1145/3301761.3301773

European Banking Authority. (2020). Payment Institutions Register. Pobrane z: https://euclid.eba.europa.eu/register/pir/search/agents (dostęp: 02.03.2020)

EY i ZPF (2019) Nadużycia na rynku finansowym. Edycja 2019. Opublikowany online: https://www.ey.com/pl/pl/newsroom/news-releases/news-ey-20191024-naduzycia-na-rynku-finansowym (dostęp 21.03.2020)

Flower, M. (2018). Refactoring. Pobrane z: https://refactoring.com/ (dostęp: 02.03.2020)

Frigó, E., Kocsis, L. (2019) Online ranking combination. In Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19). Association for Computing Machinery, New York, NY, USA, 12–19. DOI:https://doi.org/10.1145/3298689.3346993

Fu, T., Abbasi, A., Chen, H. (2010). A focused crawler for Dark Web forums. Journal of the American Society for Information Science and Technology, 61(6), 1213-1231. Pobrane z: https://www.researchgate.net/publication/220432724_A_Focused_Crawler_for_Dark_Web_Forums (dostęp: 06.03.2020)

Gojare, S., Joshi, R. i Gaigaware, D. (2015). Analysis and design of selenium webdriver automation testing framework. Procedia Computer Science, 50(1), 341-346. Doi: 10.1016/j.procs.2015.04.038 (dostęp: 06.03.2020)

Goldfein, S., Keyte, J. (2017). Big Data, Web 'Scraping' and Competition Law: The Debate Continues. New York Law Journal, 258(49), 1. Pobrane z: https://www.skadden.com/insights/publications/2017/09/big-data-web-scraping-and-competition-law-the (dostęp: 20.02.2020)

Goodman, B. i Flaxman, S. (2017). European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation". AI Magazine, 38(3), 50-57. Pobrane z: https://doi.org/10.1609/aimag.v38i3.2741 (dostęp: 06.03.2020)

Greenwich Associates (2018) Future of Investment Research Study. Raport na zlecenie Thomson Reuters. Pobrane z http://www.digitaltrendanalytics.com/wp-content/uploads/2018/11/thomson-reuters-and-greenwich-associates.pdf (dostęp: 10.03.2020)

Hagen, E., Eltringham, S., Marshall, H. J. i Michael, W. B. (2015). Prosecuting Computer Crimes: Computer Crime and Intellectual Property Section Criminal Division. Opublikowane przez: Office of Legal Education Executive Office for United States Attorneys. 18 U.S.C. §1030(a)(1) — § 1030(j), 4-56. Pobrane z: https://www.justice.gov/sites/default/files/criminal-ccips/legacy/2015/01/14/ccmanual.pdf (dostęp: 06.03.2020)

Hand, D., J. (2013). Data Mining Based in part on the article 'Data mining' by David Hand, which appeared in the Encyclopedia of Environmetrics. American Cancer Society. DOI: https://doi.org/10.1002/9780470057339.vad002.pub2 (dostęp: 06.03.2020)

Handschuh, S. i Staab, S. (2003). Annotation of the shallow and the deep web. Annotation for the semantic web, 96, 25-45.

Haque, A., Singh, S. (2015). Anti-scraping application development. 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 869-874. IEEE. Pobrane z: https://www.researchgate.net/publication/281098321_Anti-Scraping_Application_Development (dostęp: 06.03.2020)

Haster, J. (2019). xml2 package. Pobrane z: https://www.rdocumentation.org/packages/xml2/versions/1.2.2 (dostęp: 20.02.2020)

Jasim Hadi, H., Hameed Shnain, A., Hadishaheed, S. i Haji Ahmad, A. (2015). Big Data and Five V'S Characteristics. International Journal of Advances in Electronics and Computer Science, 2, 2393-2835. Pobrane z: https://www.researchgate.net/publication/332230305_BIG_DATA_AND_FIVE_V'S_CHARACTERISTICS (dostęp: 06.03.2020)

Jeffery, R. (2014). Advantages of exposing your API. Pobrane z: https://passionateaboutoss.com/advantages-of-exposing-your-api/ (dostęp: 27.02.2020).

Jiang, L., Wu, Z., Feng, Q., Liu, J., Zheng, Q. (2010). Efficient Deep Web Crawling Using Reinforcement Learning. 428-439. 10.1007/978-3-642-13657-3_46.

Kaczmarek, P. (2014). Wybrane metody efektywnej integracji komponentów w systemach rozproszonych. Zeszyty Naukowe Wydziału Elektrotechniki i Automatyki Politechniki Gdańskiej, 40, 53-56. Pobrane z: https://eia.pg.edu.pl/documents/10623/32925502/ZN_WEiA_PG_40.pdf (dostęp: 12.03.2020).

Kaczmarek, T. (2006) Integracja danych z głębokiego Internetu dla potrzeb analizy otoczenia przedsiębiorstwa, praca doktorska, Akademia Ekonomiczna w Poznaniu

Klimontowicz, M. i Pyka, A. (2016). Modele biznesowe banków – wyzwania XXI wieku. Studia i Prace WNEiZ US, 44, 153-166. DOI:10.18276/sip.2016.44/2-12

Kowalkiewicz, M. (2006) Ekstrakcja i agregacja informacji dla potrzeb podmiotów gospodarczych, praca doktorska, Akademia Ekonomiczna w Poznaniu

Krotov, V. i Silva, L. (2018). Legality and Ethics of Web Scraping.Twenty-fourth Americas Conference on Information Systems, New Orleans. Pobrane z: https://www.researchgate.net/publication/324907302_Legality_and_Ethics_of_Web_Scraping (dostęp: 15.02.2020)

Landers, R. N., Brusso, R. C., Cavanaugh, K. J. i Collmus, A. B. (2016). A primer on theory-driven web scraping: Automatic extraction of big data from the Internet for use in psychological research. Doi: 10.1037/met0000081

Lewańska, E. (2017). Towards Automatic Business Networks Identification. W W. Abramowicz, R. Alt, & B. Franczyk, W. Abramowicz, R. Alt, & B. Franczyk (Red.), *Business Information Systems Workshops BIS 2016 International Workshops, Leipzig, Germany, July 6-8, 2016, Revised Papers* (T. 263, ss. 389–398). http://doi.org/10.1007/978-3-319-52464-1_36

Maślankowski, J. (2019). Pozyskiwanie i analiza danych na temat ofert pracy z wykorzystaniem big data. Wiadomości Statystyczne, 64 (9), 60-74. Pobrane z: https://repozytorium.bg.ug.edu.pl/info/article/UOG4f30168ef8bd49f48b95430390aafc13/?tab=&lang=en#.Xb8iXZpKhPY (dostęp: 15.02.2020)

McDowell, M. (2009). Security Tip (st04-015) Understanding Denial-of-Service Attacks. Pobrane z: https://www.us-cert.gov/ncas/tips/ST04-015 (dostęp: 06.03.2020)

McKenna, S. F. (2016). Detection and classification of Web robots with honeypots. Naval Postgraduate School Monterey United States.11-14. Pobrany z: https://apps.dtic.mil/dtic/tr/fulltext/u2/1027491.pdf (dostęp: 06.03.2020)

Meegahapola, L., Mallawaarachchi, V., Alwis, R., Nimalarathna, E., Meedeniya, D., Jayarathna, S. (2018) Random Forest Classifier based Scheduler Optimization for Search Engine Web Crawlers. In Proceedings of the 2018 7th International Conference on Software and Computer Applications (ICSCA 2018). Association for Computing Machinery, New York, NY, USA, 285–289. DOI:https://doi.org/10.1145/3185089.3185103

Ministerstwo Cyfryzacji. (2018). Standard API (interfejsu programistycznego aplikacji). Pobrane z: https://dane.gov.pl/media/ckeditor/2018/10/04/standard-api_N6KVZvb.pdf (dostęp: 27.02.2020).

Mokube, I. i Adams, M. (2007). Honeypots: concepts, approaches, and challenges. In Proceedings of the 45th annual southeast regional conference, 321-326. Doi: 10.1145/1233341.1233399

Moore, A. W. i Zuev, D. (2005). Internet traffic classification using bayesian analysis techniques. In Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems — SIGMETRICS '05, 50-60. Doi: 10.1145/1064212.1064220

Muenchen, R. (2019). The Popularity of Data Science Software. Pobrane z https://r4stats.com/articles/popularity/ (dostęp: 03.03.2020)

Mulik, S., Shinde, S. i Kapase. S (2011). Comparison of Parsing Techniques For Formal Languages. International Journal on Computer Science and Engineering, 3(4), 1611 – 1615. Pobrane z: https://www.researchgate.net/publication/268275609_Comparison_of_Parsing_Techniques_For_Formal_Languages (dostęp: 03.03.2020)

Najork, M. i Heydon, A. (2002). High-performance web crawling. Kluwer Academic Publishers. Handbook of massive data sets, 25-45. Pobrane z: https://dl.acm.org/citation.cfm?id=779235 (dostęp: 06.03.2020)

von Nordheim, G., Boczek, K., Koppers, L. (2018) Sourcing the Sources, Digital Journalism, 6:7, 807-828, DOI: 10.1080/21670811.2018.1490658

Oleński, J. (2001) Ekonomika informacji. Podstawy. PWE: Warszawa

Palakollu, S. M. (2019). Scrapy Vs Selenium Vs Beautiful Soup for Web Scraping. Pobrane z: https://towardsdatascience.com/scrapy-vs-selenium-vs-beautiful-soup-for-web-scraping-24008b6c87b8 (dostęp: 02.03.2020)

Platt, A., Soens, S. (2018) Toward an automatic warning system for reputation damaging social media posts. J. Comput. Sci. Coll. 33, 5 (May 2018), 33–38.

Polski Instytut Cyberbezpieczeństwa (2019) Raport #PolskiBarometrCyberbezpieczeństwaSpołecznego2019. Opublikowany online. Pobrane z: http://picb.pl/gfx/raport/PolskiBarometrCyber2019.pdf (dostęp 21.03.2020)

ProWebScraping (2020) Web Scraping Vs Web Crawling. Pobrane z: http://prowebscraping.

com/web-scraping-vs-web-crawling/ (dostęp: 10.03.2020)

Ramler, R. i Wolfmaier, K. (2006). Economic perspectives in test automation: balancing automated and manual testing with opportunity cost. In Proceedings of the 2006 international workshop on Automation of software test, 85-91. Doi: 10.1145/1138929.1138946

Richardson, L. (2007). Beautiful soup documentation. Pobrane z: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (dostęp: 06.03.2020)

RODO (2016) Rozporządzenie Parlamentu Europejskiego i Rady (UE) 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (ogólne rozporządzenie o ochronie danych), https://eur-lex.europa.eu/eli/reg/2016/679/oj

Rozprawa sądowa pomiędzy American Airlines, Inc. oraz FareChase, Inc. (2003). 67th District Court, Tarrant County, Texas. Cause NO. 067-194022-02. Pobrane z: http://www.internetlibrary.com/pdf/american%20airlines%20farechase.pdf (dostęp: 06.03.2020)

Ramotowski, J. (2016) Bank powinien zbierać dane o kliencie z wielu źródeł. Wywiad z Adamem Kępą, opublikowany na portalu obserwatorfinansowy.pl. Pobrane z https://www.obserwatorfinansowy.pl/tematyka/rynki-finansowe/bankowosc/bank-powinien-zbierac-dane-o-kliencie-z-wielu-zrodel/ (dostęp 22.03.2020)

Sądel. E. (2018). Selektory w Selenium – najprostsze metody lokalizowania elementów na stronie. Pobrane z: https://testelka.pl/selektory-w-selenium-najprostsze-metody/. (dostęp: 06.03.2020)

ScienceDaily. (2019). Web crawler. Pobrane z: https://www.sciencedaily.com/terms/web_crawler.htm (dostęp: 06.03.2020)

Scrapy. (2019). Architecture overview — Scrapy 1.8.0 documentation. Pobrane z: https://docs.scrapy.org/en/latest/topics/architecture.html (dostęp: 06.03.2020)

Scrapy. (2019b). Dokumentacja Scrapy. Pobrane z: https://docs.scrapy.org/en/latest/(dostęp:06.03.2020)

Selenium. (2019). Getting started. Pobrane z: https://selenium.dev/documentation/en/getting_started/. (dostęp: 15.02.2020)

Selenium. (2019b). The Selenium project and tools. Pobrane z: https://selenium.dev/documentation/en/introduction/the_selenium_project_and_tools/. (dostęp: 06.03.2020)

Selenium. (2019c). Browsers. Pobrane z: https://selenium.dev/documentation/en/getting_started_with_webdriver/browsers/. (dostęp: 06.03.2020)

Singhal. G. (2020). Crawling the Web with Python and Scrapy. Pobrane z: https://www.pluralsight.com/guides/crawling-web-python-scrapy (dostęp: 06.03.2020)

Sivakorn, S., Polakis, J. i Keromytis, A. D. (2016). I'm not a human: Breaking the Google reCAPTCHA. Pobrane z: https://media.kasperskycontenthub.com/wp-content/uploads/sites/63/2017/11/21031220/asia-16-Sivakorn-Im-Not-a-Human-Breaking-the-Google-reCAPTCHA-wp.pdf (dostęp: 06.03.2020)

Southwest Airlines Co. (2004) Southwest Airlines Co., Plaintiff, V. Farechase, Inc. and Outtask, Inc., Defendants. Pobrane z: https://law.justia.com/cases/federal/district-courts/FSupp2/318/435/2472845/ (dostęp: 06.02.2020)

Spina, D., Meij, E., de Rijke, M., Oghina, A., Thuong Bui, M., Breuss, M. (2012) Identifying entity aspects in microblog posts. In Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval (SIGIR '12). Association for Computing Machinery, New York, NY, USA, 1089–1090. DOI:https://doi.org/10.1145/2348283.2348483

Spitzner, L. (red.) (2003). Honeypots: tracking hackers. Boston: Addison Wesley.

Steven, J., Z. (2017). Integrated Development Environments. Pobrane z: https://www.cs.odu.edu/~zeil/cs350/f17/Public/IDEs/index.html (dostęp: 06.03.2020)

Sun, Y., Councill, I. G., Lee Giles, C. (2010) The Ethicality of Web Crawlers. In Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology — Volume 01 (WI-IAT '10). IEEE Computer Society, USA, 668–675. DOI:https://doi.org/10.1109/WI-IAT.2010.316

Sun, Y., Zhuang, Z., Lee Giles, C. (2007) A large-scale study of robots.txt. In Proceedings of the 16th international conference on World Wide Web (WWW '07). Association for Computing Machinery, New York, NY, USA, 1123–1124. DOI:https://doi.org/10.1145/1242572.1242726

Szczepaniak, D. (2018). Wstęp do REST API. Pobrane z: https://devszczepaniak.pl/wstep-do-rest-api/ (dostęp: 27.02.2020).

Technopedia. (2019) Web Scraping. Pobrane z: https://www.techopedia.com/definition/5212/web-scraping (dostęp: 02.03.2020)

Thelwall, M. (2001). A web crawler design for data mining. Journal of Information Science, 27(5), 319–325. Pobrane z: https://doi.org/10.1177/016555150102700503 (dostęp: 06.03.2020)

Tidyverser. (2020). Tidyverse packages. Pobrane z: https://www.tidyverse.org/packages/ (dostęp: 06.03.2020)

Tiru, A., Toma I. I Necula, M. (2017). RTempo – an R package to access the TEMPO-Online database. Romanian Statistical Review, 4, 57-66. Pobrane z: http://www.revistadestatistica.ro/index.php/rtempo-an-r-package-to-access-the-tempo-online-database/ (dostęp: 19.02.2020)

Ustawa z dnia 27 lipca 2001 r. — O ochronie baz danych (Dz.U. 2001, Nr 128, poz. 1402 z późn. zm.). Pobrane z: http://prawo.sejm.gov.pl/isap.nsf/DocDetails.xsp?id=WDU20011281402 (dostęp: 06.03.2020)

Von Ahn, L., Maurer, B., McMillen, C., Abraham, D. i Blum, M. (2008). recaptcha: Human-based character recognition via web security measures. Science, 321(5895), 1465-1468. Doi: 10.1126/science.1160379 (dostęp: 06.03.2020)

W3C. (2017). XML Path Language (XPath) 3.1. Pobrane z: https://www.w3.org/TR/2017/REC-xpath-31-20170321 (dostęp: 06.03.2020)

W3schools. (2020). CSS Selector Reference. Pobrane z: https://www.w3schools.com/cssref/css_selectors.asp (dostęp: 19.02.2020)

Wanas, N., El-Saban, M., Ashour, H., Ammar, W. (2008) Automatic scoring of online discussion posts. In Proceedings of the 2nd ACM workshop on Information credibility on the web (WICOW '08). Association for Computing Machinery, New York, NY, USA, 19–26. DOI:https://doi.org/10.1145/1458527.1458534

Wang, J., Yu, C.T., Yu, P.S., Liu, B., Meng, W. (2015) Diversionary Comments under Blog Posts. ACM Trans. Web 9, 4, Article 18 (September 2015), 34 pages. DOI:https://doi.org/10.1145/2789211

Webber, W., Moffat,A., Zobel, J. (2010). A similarity measure for indefinite rankings. ACM Trans. Inf. Syst. 28, 4, Article 20 (November 2010), 38 pages. DOI:https://doi.org/10.1145/1852102.1852106

Wickham, H. (2019). Rvest. Pobrane z: https://cran.r-project.org/web/packages/rvest/rvest.pdf (dostęp: 06.03.2020)

Wickham, H. (2019b). httr package. Pobrane z: https://www.rdocumentation.org/packages/httr/versions/1.4.1 (dostęp: 06.03.2020)

Wikibooks. (2020). CSS/Selektory — Wikibooks, biblioteka wolnych podręczników. Pobrane z: https://pl.wikibooks.org/wiki/CSS/Selektory (dostęp: 06.03.2020)

Wiliams, J. (2018). 13 Reasons Why Web Scraping is Getting More Popular Since the Past Decade. Pobrane z: https://www.promptcloud.com/blog/13-reasons-why-web-scraping-is-getting-popular/ (dostęp: 06.03.2020)

World Health Organization. (2020). The global health observatory. Pobrane z: https://www.who.int/data/gho

Yang, C., Liao, H. J. (2010). Using the Robots. txt and Robots Meta tags to implement online copyright and a related amendment. Journal: Library Hi Tech, 28, 94-106.

YIT. (2020). Polityka prywatności danych i warunki użytkowania. Pobrane z: https://www.yit.pl/polityka-prywatnosci-danych-i-warunki-uzytkowania (dostęp: 06.03.2020)

You, G., Hwang, S. (2007) Personalized ranking: a contextual ranking approach. In Proceedings of the 2007 ACM symposium on Applied computing (SAC '07). Association for Computing Machinery, New York, NY, USA, 506–510. DOI:https://doi.org/10.1145/1244002.1244119

Zelent, M. (2017). Sieci równorzędne oraz klient-serwer. Pobrane z: https://miroslawzelent.pl/informatyka/sieci-rownorzedne-oraz-klient-serwer/ (dostęp: 27.02.2020).