UNIWERSYTET EKONOMICZNY W POZNANIU

Wydział Informatyki
i Gospodarki Elektronicznej

Wojciech Dopieralski

Wykorzystanie *web scrapingu* do pozyskania danych na temat wydarzeń kulturalnych

The utilize of *web scraping* to gather data about cultural events

Praca licencjacka

Promotor: dr Łukasz Wawrowski

Data przyjęcia:

Podpis promotora

Kierunek: Informatyka i ekonometria

Specjalizacja: Analityka gospodarcza

Poznań 2019

Spis treści

Wstęp

Analiza danych od wieków stanowi zadanie badaczy na całym świecie. głównym problemem związanym z tym tematem jest dostęp do przydatnych i aktualnych informacji. Wraz z nadejściem i rozpowszechnieniem się Internetu na urządzeniach stacjonarnych jak i mobilnych ilość ogólnodostępnych informacji stała się zatrważająca. Konieczne stało się wynalezienie rozwiązań, które pozwolą w sposób efektywny pobierać, kolekcjonować i przetwarzać gigantyczne zbiory danych. Warto zaznaczyć, że procesy związane ze zbieraniem danych ciągle się rozwijają i z roku no rok wchodzą na następne szczeble. Jest to silnie powiązane z rozwojem rozwiązań *smart*. W związku z tym rozwiązaniem coraz to większa liczba urządzeń codziennego użytku jest w stanie zbierać i przechowywać przydatne do analizy informacje.

Uniwersalną metodą, która pozwala nam pozyskać dane z wszelkiego rodzaju stron internetowych (w tym portali społecznościowych) jest *web scraping*. Pomimo szerokiego zastosowania praktycznego, które przekłada się na możliwości wykorzystania opracowanej wiedzy teoretycznej z zakresu analiz statystycznych jak i uczenia maszynowego w literaturze znajduję się mało pozycji, które w sposób kompleksowy opisywałyby to zjawisko. Celem głównym pracy jest wypełnienie powstałej luki i ustrukturyzowane informacji teoretycznych o *web scrapingu*, jak i przedstawienie narzędzi wykorzystywanych do *scrapowania*. Szczególna uwaga została poświęcona na narzędzia stosowane w *data science*, ze względu na znaczny rozwój tego obszaru na rynku IT. Dodatkowo celem pobocznym pracy jest praktyczne zastosowanie opisanych metod w środowisku z trudnym dostępem do danych. Temat badawczy został wybrany nie tylko ze względu na trudności związane z dostępem do ustrukturyzowanych danych, ale również z uwzględnieniem personalnych zainteresowań autora pracy. Praca składa się z wstępu, trzech rozdziałów i zakończenia.

W pierwszym rozdziale zostały poruszone kwestie związane z określeniem czym jest

web scraping i jak go definiujemy. Poruszony został problem analizy sentymentu, która w ostatnich latach zyskuje znacząco na popularności i w swoich pierwszych etapach silnie korzysta z ekstrakcji danych. Wyjaśnione zostały różnicy pomiędzy *scraperami* a *crawlerami*. Dodatkowo opisano wybrane algorytmy używane w pająkach sieciowych. Na zakończenie rozdziału poruszone zostały aspekty prawne związane z *web scrapingiem* z uwzględnieniem tego w jaki sposób administratorzy stron zabezpieczają swoje zasoby przed potencjalnymi *scraperami*.

W drugim rozdziale poruszone zostały kwestie dotyczące najpopularniejszych narzędzi używanych do *web scrapingu* w ramach dziedziny nauki jaką jest *data science*. Szczególną uwagę przywiązano do dwóch języków programowania: R i Python. Na początku rozdziału omówiono przydatne narzędzie, które pozwala ominąć wiele zabezpieczeń *anty scrapingowych* – Selenium. Jest ono dostępne w ramach pakietów/bibliotek dla obu języków. W ramach Pythona omówione zostały 2 pakiety: *Beutiful Soup* i *Scrapy*. R posiada jeden wiodący pakiet do *web scrapingu - rvest*.

Trzeci rozdział rozpoczyna się od zdefiniowania i przedstawienia pojęcia imprez fanowskich. W dalszej części rozdziału zaprezentowane zostało wykonanie ekstrakcji danych przy użyciu języka programowania R i opisanego w rozdziale drugim pakietu *rvest*. Wybrane strony internetowe o tematyce imprez fanowskich zawierają dane, które nie mają struktury pozwalającej na ich bezpośrednią analizę.

Podstawowymi źródłami informacji w pracy są artykuły naukowe związane z informatyką, statystyką i ekonomią. Równie istotnym źródłem informacji okazała się dokumentacja opisywanych pakietów jak i portale internetowe, które zawierają kursy związane z praktycznym zastosowaniem omawianego tematu.

Rozdział 1

Teoretyczne aspekty i zastosowanie *web scrapingu*

1.1 Sposoby na pozyskanie danych i definicja *web scrapingu*

Dane są podstawą każdego badania związanego z szeroko pojętą statystyką. Bez swobodnego dostępu do potrzebnych i użytecznych informacji badacz nie jest wstanie wyciągnąć żadnych konstruktywnych wniosków. W związku z rewolucją internetową i ustabilizowaniem się tzw. gospodarki internetowej (Talar & Kos-Łabędowicz, 2014) ludzie zaczęli generować olbrzymie ilości informacji, dzięki którym naukowcy mogą dokonywać wcześniej niemożliwych do wykonania analiz. Dane są generowane przez indywidualnych użytkowników sieci, prywatne firmy i podmioty publiczne. Wraz z napływem nowych źródeł informacji pojawiły się też nieznane wcześniej problemy. Dane pojawiają się na przeróżnych stronach internetowych, każda z nich posiada swoją własną strukturę i wymaga odpowiedniego podejścia do problemu. Część instytucji udostępniana badaczom dane w przyjaznych formatach, takich jak csv (*comma-separated values*) bądź xls. Jednakże znaczna większość danych, które można wykorzystać w innowacyjnych i nieprzeprowadzanych wcześniej analizach jest generowana przez użytkowników sieci. W efekcie dane te pojawiają się na stronach bądź portalach w nieustrukturyzowanej bądź trudnej do dalszej obróbki formie.

Istnieję wiele sposobów na pozyskanie informacji z sieci i w zależności od potrzeb

osoba dokonująca ekstrakcji powinna wybrać optymalny model. Jednym z nich jest korzystanie z API (*Application Programming Interface*), które pozwalają na komunikację z serwerem i bazą danych w celu pobierania danych w przyjaznych do przetworzenia formatach. Zasadnicza różnica między API, a standardowym *back-endem* strony jest taka, że w przypadku korzystania z API nie istnieje konieczność projektowania *front-endu* (Eising, 2017). Dzięki temu firmy są wstanie przyśpieszać proces komunikacji, co wpływa na redukcję kosztów. Innym sposobem na zdobycie potrzebnych danych jest wyciągniecie ich z formatu pdf (*portable document format*). Zgodnie z informacjami przytoczonymi przez Castrillo-Fernándeza (Castrillo-Fernández, 2015) 70% informacji indeksowanych przez Google jest udostępnianych w Internecie w formacie pdf. Warto zaznaczyć, że nie jest on przyjazny i dostosowany do pozyskiwania informacji. Mimo to ze względu na wysoką użyteczność danych zawartych w plikach udostępnianych w formacie pdf istnieją narzędzia, które umożliwiają skuteczną ekstrakcję danych (np. pakiet dla języka R o nazwie *pdftools* (Ooms, 2019)). Dane można też pobierać ręcznie, kopiując je. Sposób ten jest wysoce nieefektywny z punktu widzenia czasu jak i kosztów. Poza tym charakteryzuję się dużym prawdopodobieństwem popełnienia błędu, a więc jest ryzykowny. Wprowadzenie nieprawidłowych danych do modelu może być katastrofalne w skutkach. Alternatywą w stosunku do pobierania danych jest wysłanie prośby o udostępnienie. W celu uzyskania potrzebnych informacji należy zwrócić się do instytucji, która je posiada i ma możliwość udostępnienia ich w dogodnej do analizy formie (Lindenberg, 2011).

Poza wyżej wymienionymi metodami badacz, może zastosować szereg rozwiązań, które określamy jako *web scraping*. Inne spotykane w praktyce określenia na tą metodę to *screen scraping*, *web data extraction* i *web harvesting* (WebHarvey, 2018). W polskiej nomenklaturze zjawisko to nazywane jest ekstrakcją i agregacją danych. W niniejszej pracy określenia te będą używane naprzemiennie. Metoda ta dynamicznie zyskuję na popularności, wiąże się to ze znacznym wzrostem zainteresowanie dziedziną nauki jaką jest *data science* (Matusiak, 2012). Dodatkowo warto zauważyć, że administracja publiczna w wielu krajach przyjęła model promujący transparentność danych publicznych, tym samym wiele informacji jest udostępniany na stronach rządowych za darmo. Podczas przeprowadzania analizy dotyczącej rożnych problemów wewnątrz firmy zawsze warto rozważyć wsparcie modelu o dostępne publicznie dane, które mogą popra-

wić jego efektywność (Castrillo-Fernández, 2015).

Web scraping polega na pobieraniu danych z www (*World Wide Web*) i zapisywaniu ich w pliku bądź bazie danych w celu dalszego przetworzenia. Zazwyczaj dane są pobieranie z sieci przy wykorzystaniu protokołu HTTP (*Hypertext Transfer Protocol*) bądź przeglądarki internetowej. Użytkownik może tego dokonać samemu poprzez odpowiednie oprogramowanie bądź przy użyciu specjalnego pająka (*web crawler*). *Web crawlery* nazywane też botami indeksującymi, robotami internetowymi bądź pełzaczami zostaną szerzej opisane w podrozdziale 1.3. Technologia ekstrakcji idealnie wpasowuje się w paradygmat olbrzymich ilości danych (*big data*), ze względu na możliwość pobierania informacji publicznie dostępnych (Mooney, Westreich & El-Sayed, 2015). Ze względu na konieczność dopasowania się do różnych scenariuszy, współczesna technologia ekstrakcji danych została również dostosowana do pobierania niewielkich zbiorów. Część systemów jest wstanie od razu przetworzyć to co pobierze w ustrukturyzowane formy. Istotną kwestią, związaną z problemem *web scrapingu* jest też możliwość przetwarzania obrazów, które są masowo udostępniane w sieci za pośrednictwem portali społecznościowych. Przydatne dane znajdują się też w gotowych wizualizacjach, przygotowanych przez ludzi tworzących określone artykuły. Kolejnym problemem, który należy rozwiązać podczas ekstrakcji jest przetwarzania języka naturalnego. Dzięki dostępowi do wszystkich wymienionych wcześniej informacji badacze mogą stwierdzić w jaki sposób użytkownicy poruszają się po stronach internetowych, jak kształtują się trendy i na czym należałoby skupić przyszłe inwestycje (Zhao, 2017).

Proces pobierania danych z Internetu poprzez techniki *web scrapingowe* można podzielić na dwa sekwencyjne kroki:

- pozyskanie zasobów z sieci,
- wyróżnienie potrzebnych informacji spośród tego co udało się pobrać.

Proces ekstrakcji danych zaczyna się od skomponowania zapytania *HTTP* w celu pozyskania zasobów ze wskazanej w programie strony. Zapytanie może być sformatowane w *URL* i zawierać metodę *GET* bądź fragment *HTTP* zawierający metodę *POST*. Zapytanie zostaje przetworzone i zaakceptowane przez stronę, z której odpowiedni materiał zostanie pobrany i przesłany do programu odpowiedzialnego za scraping. Pobrane dane mogą przyjmować różny format, najpopularniejsze to: *JSON*, strony *HTML* bądź *XML*.

Istnieją dwa rodzaje modułów do *web harvestingu*, jeden skupia się na komponowaniu zapytań *HTTP*, a drugi na *parsowaniu* i wyodrębnianiu określonych informacji z surowego *HTML*. *Web scraping* jest stosowany do rozwiązania wielu problemów natury komercyjnej jak i hobbistycznej, za przykład mogą służyć:

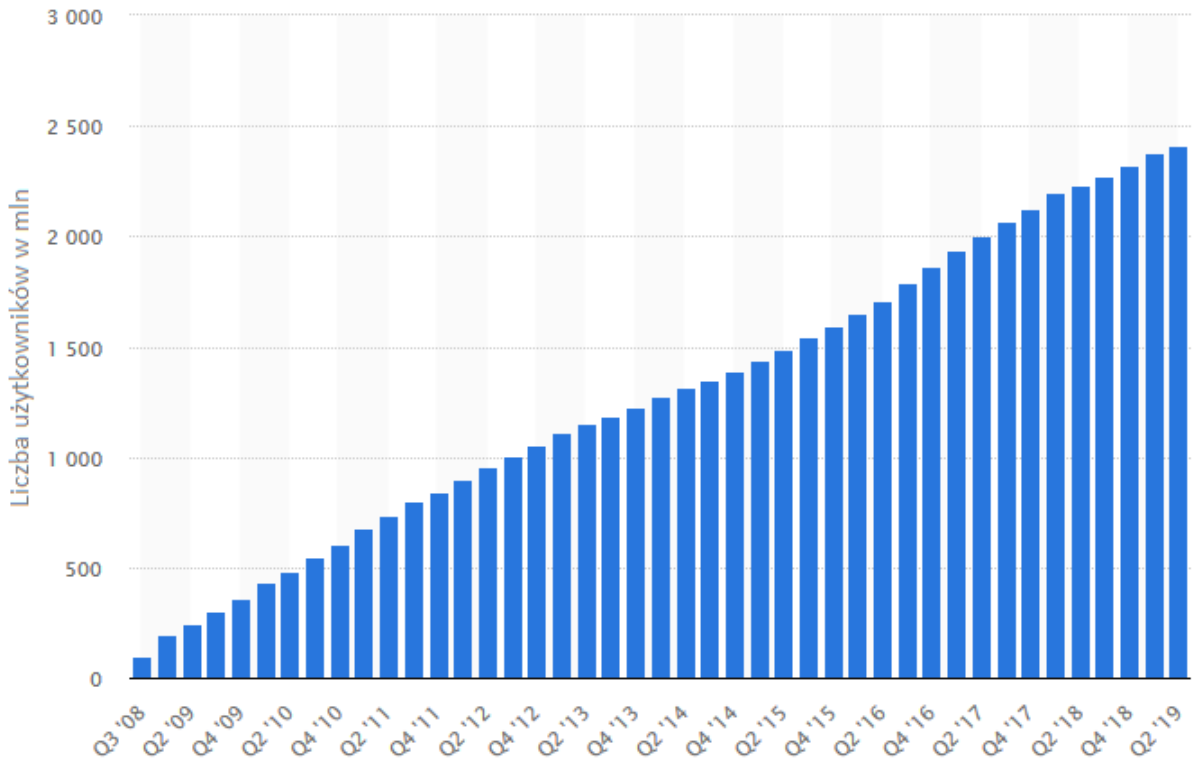
- monitorowanie i porównywanie cen,
- zbieranie recenzji produktów,
- zbieranie danych o nieruchomościach,
- monitorowanie pogody,
- wykrywanie zmian na stronach,
- zbieranie kontaktów numery telefonów, adresy e-mail, etc.) (Zhao, 2017).

W związku z rewolucją internetową na ogromną skalę zaczęto używać portali społecznościowych, które są idealnym materiałem do *scrapowania* i przeprowadzania analiz. Ze względu na wpływ i możliwości tych badań, zostaną one omówione w następnym podrozdziale.

1.2 *Web Scraping* jako składowy element analizy sentymentu.

Wyjątkowo istotną kwestię w XXI wieku odgrywają *social media*. Portale i strony tego typu mają znaczący wpływ na postrzeganie świata przez rzeszę ludzi. Warto podkreślić, że najczęstszymi użytkownikami są osoby młode, których poglądy dopiero się kształtują. W erze demokracji, możliwość wpływania, a wręcz sterowania wyborcami wydaje się kluczowa. Nie trudno wyciągnąć wnioski, że *social media* mają ogromny wpływ na życie społeczno gospodarcze całych narodów. Są one świetnym źródłem informacji dla polityków na temat opinii społecznych. Główne zainteresowanie ludzi rządzących, bądź aspirujących do władzy może dotyczyć opinii na ich własny temat, jak i zdania o opozycji. Dane *scrapowane* z takich źródeł stanowią potężne narzędzie, które może służyć do przeprowadzenia wielu analiz.

Przykładem obszaru nauki, który idealne wpasowuje się w omawiane zagadnienie jest analiza wydźwięku (sentymentu). Wykorzystuje ona opinie i oceny w celu wykrycia emocjonalnego nastawienia autorów określonych treści do produktów, usług bądź innych ludzi (Abramowicz i in., 2015). Zgodnie ze zmianami, które można zaobserwować w *Google Trends*, w ostatnich latach zyskuje ona na znacznej popularności (Czech, 2016). Na podstawie tej analizy można wyciągnąć wnioski na temat stosunku autorów do jakiejś osób, stwierdzić czy jest on pozytywny czy negatywny. Metoda ta zyskuję wraz z rozwojem *social mediów*, które są źródłem ogromnego zbioru opinii społecznych. Dodatkowo wszystkie dane są w postaci cyfrowej co umożliwia skuteczną ekstrakcję. Warto zaznaczyć, że *social media* rozwijają się cały czas, co pokazuje wzrost liczby użytkowników poszczególnych portali na przestrzeni lat. Rysunek 1.1 przedstawia liczbę użytkowników najpopularniejszego portalu społecznościowego *Facebook* w latach 2008–2019.



Rysunek 1.1. Liczba użytkowników portalu Facebook w latach 2008–2019 w ujęciu kwartalnym

Źródło: (Statista, 2019)

Analiza sentymentu charakteryzuje się sześcioma podstawowymi etapami. Pierw-

szy z nich jest kluczowy z punktu widzenia tematu pracy, gdyż polega na dostarczeniu i czyszczeniu danych. W tej dziedzinie idealne wydaje się zastosowanie technologii opartych na *web scrapingu*. Kluczowe jest by dostarczyć dane z jak największej liczby źródeł. Na szczęście dzięki rewolucji internetowej badacze mają swobodny dostęp do wielu stron publikujących odpowiednie informacje. Ze względu na różne źródła danych konieczne jest ich oczyszczenie i pozostawienie w formie czystego tekstu. Drugi etap wymaga ustalenia czy dany tekst jest odpowiedni pod względem kryterium emocjonalności. W analizie wydźwięku istotne są tylko te teksty, które pozwalają stwierdzić jakie jest nastawienie odbiorcy do badanej sprawy. Trzeci etap polega na wyróżnieniu pojęć z przekształconego tekstu. Dokładniej chodzi o wydobycie słów kluczowych, które będą podlegać dogłębniejszej analizie. Wyróżnia się trzy podejścia:

- Słownikowe - tworzymy słownik, który zawiera zbiór nazw, ze wszystkimi możliwymi wersjami językowymi.
- Na podstawie reguł - identyfikujemy ciągi znaków, przykładem może być wyszukiwanie słów zaczynających się z dużej litery.
- Metody uczenia maszynowego - algorytmy w procesie uczenia wyodrębniają określone słowa automatycznie.

W czwartym etapie przeprowadza się wyszukiwanie słów nacechowanych. Procedurę tą przeprowadzamy w obrębie danej jednostki, którą poddajemy badaniu. W ramach tego etapu korzysta się z tzw. korpusu słów, który zawiera przypisane emocje (postrzegamy coś negatywnie, neutralnie bądź pozytywnie). Etapem piątym jest stworzenie podsumowania, w ramach którego następuje grupowanie słów zgodnie z ich wydźwiękiem. Zestawia się określenia pozytywne, negatywne oraz neutralne i dokonuje sumowania. Ostatni, szósty etap to wygenerowanie raportu, który będzie możliwy do zrozumienia dla badacza (Abramowicz i in., 2015).

Analiza sentymentu znajduję zastosowanie w wielu obszarach, jednym z nich jest branża *E-commerce* i związany z nią marketing. Firmy mogą śledzi jakie informacje przesyłają im klienci na temat produktów bądź usług i w efekcie mogą planować efektywne strategie obronne danej marki (Czech, 2016). Trzeba pamiętać, by analizie poddawać nie tylko opinie na temat produktów pochodzących ze stron wewnętrznych. Dzięki możliwościom ekstrakcji danych osoba dokonująca analizy ma sposobność

pozyskania informacji ze strony konkurencji jak i sprzedawców pośrednich. Dodatkowe źródła tematyczne to fora internetowe, bądź grupy na portalach społecznościowych.

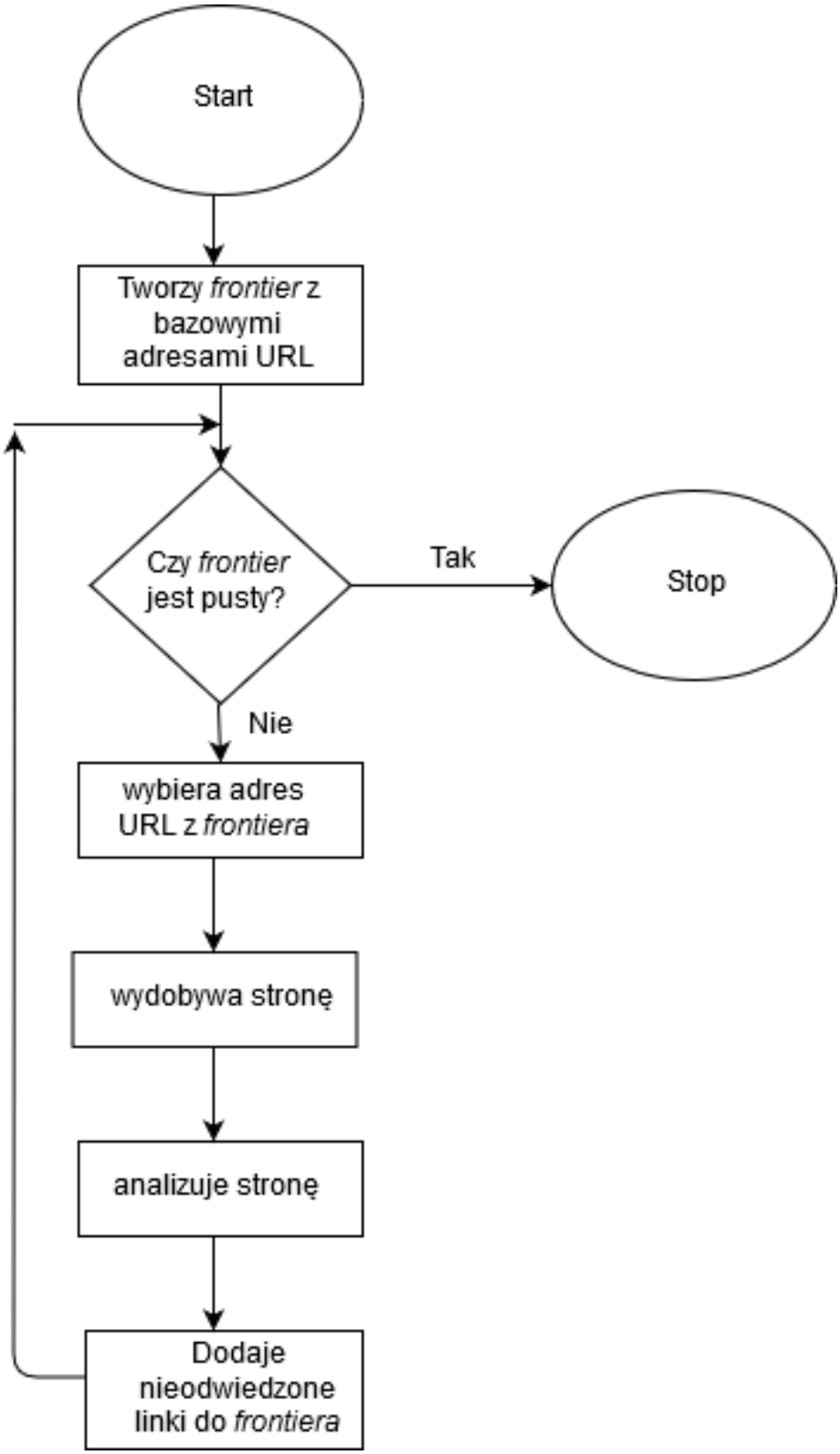
1.3 *Web crawler* jako metoda automatycznego pobierania danych.

Z punktu widzenia pobierania danych i dostępu do informacji zawartych na różnych stronach istotnym zagadnieniem jest tzw. *web crawler* inaczej robot internetowy, bot indeksujący bądź pająk lub pełzacz (Wikipedia, 2019b). Pierwszy program tego typu został zaprogramowany przez naukowców z MIT w 1993 roku, w celu zbadania wielkości *World Wide Web* (Chatterjee & Nath, 2017). Narzędzie to pobiera informacje ze strony na podstawie adresu startowego i przyjętych założeń. Użytkownik może narzucić mu na ile linków do przodu ma się poruszyć, jaki typ plików ignorować itd. Pająk pobiera wszystko co jest podlinkowane od punktu startowego. Sporo stron reguluje działanie pająków i umieszcza w swojej strukturze plik o nazwie robots.txt, który pozwala określić jak narzędzie powinno działać na danej stronie. W praktyce plik ten zawiera listę stron, na którą robot nie powinien się dostawać (Coffin, 2010). Bazowe kroki wykonywane przez *web crawlera* przedstawiają się następująco:

1. Dodanie jednego bądź paru adresów URL do *frontiera*.
2. Wybranie adresu z *frontiera* zgodnie z przyjętymi zasadami.
3. Wydobycie odpowiedniej strony internetowej.
4. Przeanalizowanie strony w celu wydobycia adresów URL .
5. Dodanie wszystkich nieodwiedzonych linków do *frontiera*.

   Algorytm powtarza swoje działanie od kroku 2, do momentu w którym cokolwiek znajduję się we *frontierze*.

Proces ten został również przedstawiony na schemacie blokowym:



Rysunek 1.2. Schemat blokowy działania podstawowego algoytmu *web crawlera*

Źródło: Opracowanie własne na podstawie (Chatterjee & Nath, 2017)

Roboty internetowe możemy podzielić na dwie grupy: użyteczne dla użytkowników sieci i takie, które utrudniają przeglądanie Internetu. Warto zauważyć, że część techniczna *crawlerów* złych i dobrych potrafi być do siebie zbliżona, kwintesencja podziału leży w sposobie wykorzystania technologii. Roboty, które wykonują pożyteczną pracę możemy podzielić na:

- Roboty wykrywające plagiat - poszukują treści, które zostały skopiowane bez zgody autora. Potrafią one wykryć niewłaściwie pliki, jak i zdjęcia umieszczone nielegalnie. Przykładem takiego rozwiązania jest System Content ID.
- Roboty gromadzące aktualne informacje - skupiają się na poszukiwaniu nowości, między innymi danych pogodowych, kursów bądź ważnych informacji. Do grupy tych robotów należy np. *Google home* bądź *Amazon Echo*.
- Roboty zbierające informacje o strukturach i treściach stron internetowych - za pomocą tego typu robotów wyszukiwarki zbierają informację na temat nowo powstałych stron. Są to najpopularniejsze *crawlery* w sieci, jako przykład może posłużyć *Googlebot*.
- Roboty pobierające dane o cenach produktów - ich zadaniem jest znalezienie najlepszej z punktu widzenia konsumenta oferty. *Honey* to robot, który automatycznie znajduję kupon rabatowy i dobierze go do naszego koszyka, wyjątkowo użyteczny w razie zakupów w *Amazonie*.

Nie wszystkie programy, które działają w roli pająków służą dobrym celom. Podział robotów wyrządzających szkody prezentuje się następująco:

- Roboty "klikające” (*clickbot*) — ich zadaniem jest klikanie w reklamy, w ten sposób imitują one ruch prawdziwego użytkownika po danej stronie. Boty te utrudniają analizę, ze względu na zaburzenie realnych wyników, dodatkowo mogą one powodować straty finansowe, ze względu na sztuczne zwiększanie kliknięć w płatne reklamy.
- Roboty "pobierające” — ich szkodliwe działanie nie polega na samym pobieraniu, jednak na zaburzaniu realnego zainteresowania danym produktem.

- Roboty "maskujące” — Podają się one za prawdziwych użytkowników, ich celem jest ominięcie zabezpieczeń i wykonanie ataku np. typu DDoS.
- Roboty plagiatujące - są przeciwieństwem robotów antyplagiatowych. Pobierają one treści, które nie są udostępnione celowo. Dodatkowo pająki te upowszechniają treści pobrane bez zgody autorów i dokonują niewielkich zmian by utrudnić usunięcie plagiatu z sieci.
- Roboty spamujące - są na tyle "inteligentne", że możemy mieć problem z odróżnieniem ich od człowieka. Wyjątkowo często zalewają media społecznościowe.
- Roboty szpiegujące - wyszukują informacji na temat ludzi z publicznie dostępnych danych. Na tej podstawie sporządzają listy mailingowe (Ozorowski, 2017).

W ramach swojego działania roboty internetowe wykorzystują rożne algorytmy, część z nich przestawia tabela 1.1.

Tabela 1.1. Najpopularniejsze algorytmy wykorzystywane w *crawlerach*

Nazwa algorytmu	**Opis działania**
Poszukiwanie wszerz (breadth-first search)	Najprostsza forma algorytmów używanych w pająkach. Zaczyna od jakiegoś linku i przemieszcza się pomiędzy linkami powiązanymi, nie zastanawiając się jakiego tematu dotyczą. Nie skupia się na zależności w przeszukiwanych stronach.
Najlepsze pierwsze wyszukanie (Best First Search)	Jest algorytmem heurystycznym. Trafność dopasowania jest wyliczana dla każdego linku i linku najbardziej z nim powiązanego.
A*	Korzysta z algorytmu Best First Search, kalkuluje trafność każdego linku i różnice między trafnością strony docelowej i aktualnym linkiem. Suma służy jako miara wyboru najlepszej ścieżki.
Dostosowujący się A*	Działa na świadomej heurystyce w celu skupienia się na wyszukiwaniu. W każdej iteracji ulepsza wartość trafności dopasowania i używa jej w następnym przejściu.
Fish Search	Dynamiczny heurystyczny algorytm. Działa na zasadzie intuicji - dopasowane linki, mają dopasowanych sąsiadów. W tym algorytmie kluczowa jest kolejność URL.

Źródło: Opracowanie własne na podstawie (Chatterjee & Nath, 2017)

1.4 Uwarunkowania prawne związane z *web scrapingiem*

Problemy techniczne nie są jedynymi, które warto rozważyć podczas próby ekstrakcji danych z sieci. Paradoksalnie dużo większe znaczenie i wpływ na możliwości wykorzystania danych pobranych z Internetu ma obowiązujące prawo. Bardzo często strony

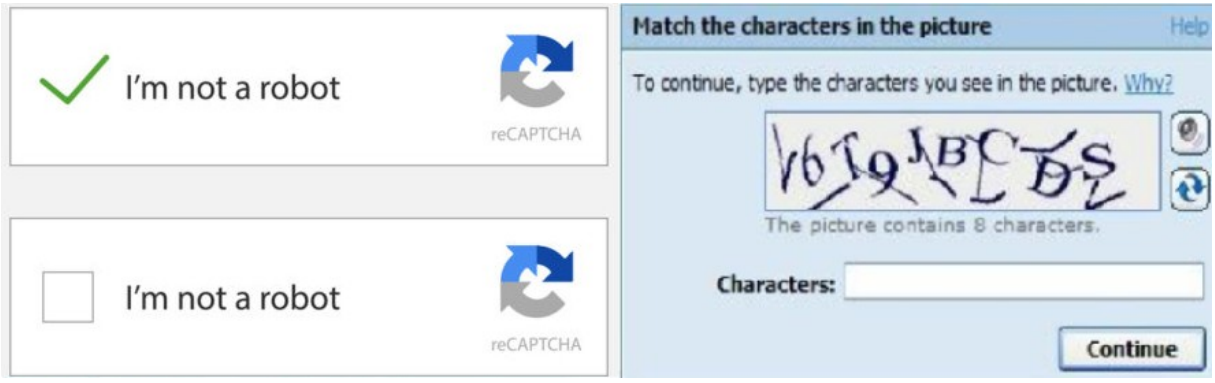
w swoich regulaminach zgodnie z obowiązującymi kodeksami zastrzegają, że określone informację nie mogą być wykorzystywane do użytku komercyjnego. Oznacza to , że firma, która chciałaby stworzyć produkt opierający się na tych danych (np. ranking bądź katalog) będzie musiała najpierw uzyskać zgodę na ich wykorzystanie, co najczęściej wiąże się z dodatkowymi kosztami. W przypadku użytku niekomercyjnego (np. do prowadzenia badań) regulaminy są zdecydowanie przychylniejsze *scrapowaniu*.

Przed rozpoczęciem ekstrakcji programista powinien zapoznać się z regulaminem strony, z której ma zamiar pozyskać dane. Pośród zapisów regulaminu najpewniej znajdzie się informacja dotycząca tego czy taki proceder jest pożądany przez właściciela strony. Nawet jeżeli w regulaminie znajduję się zapis, który zabrania wykonywania czynności *scrapingowych* to należy przestudiować obowiązujące w danym państwie prawo. Może się okazać, że zapisy znajdujące się w regulaminie nie mają żadnej podstawy prawnej i są tylko i wyłącznie wymysłem właściciela strony. Dobrą praktyką jest też przestudiowanie działu nazywanego FAQ (*frequently asked questions*), w którym to znajdują się często zadawane pytania, na które administratorzy portalu musieli odpowiadać. Jeżeli właściciele strony regularnie otrzymywali prośby dotyczące możliwości skorzystania z ich danych, to najpewniej w tej sekcji znajdzie się odpowiedź dotycząca tego, czy istnieje możliwość ich legalnego *scrapowania*. Na stronie można też uzyskać informacje o możliwości pozyskania danych za pomocą udostępnionego przez daną organizację bądź firmę narzędzia (np. API). W takiej sytuacji proces *scrapowania* nie ma żadnego sensu, gdyż udostępnione rozwiązanie będzie najpewniej wygodniejsze, efektywniejsze i w pełni zgodne z obowiązującym prawem. Istnieją też organizację i firmy, które na swoich stronach będą od razu umieszczały oferty związane z pozyskiwaniem posiadanych przez nich danych. Dane są postrzegane wtedy jako towary i będą posiadały odpowiednią cenę, którą należy zapłacić by uzyskać dostęp do informacji. Najczęściej w ramach oferty będziemy mogli wybrać formę, w jakiej to dane mają do nas trafić.

Kluczową ustawą, do której mogą się odnosić właściciele stron, podlegających *scrapowaniu* jest Ustawa z dnia 27 lipca 2001 roku o ochronie baz danych (Ustawa o ochronie baz danych, 2001). Zgodnie z ustawą zbiory danych zawarte w bazie są inaczej traktowane niż reszta utworów, która podlega pod ustawę o prawie autorskim i prawach pokrewnych. Baza danych definiowana jest jako zbiór informacji, lub jakichkol-

wiek innych materiałów zgromadzonych zgodnie z określonymi regułami, uwzględniając zbiory elektroniczne. Wymaga ona nakładów inwestycyjnych w celu prezentacji lub weryfikacji jej zawartości. Ustawa definiuje producenta baz danych jako osobę fizyczną, prawną bądź jednostkę bez osobowości prawnej, która ponosi nakłady inwestycyjne, umożliwiające istnieje bazy. Warto zaznaczyć, że ustawie nie podlegają programy wykorzystywane do tworzenia baz danych, które w większości przypadków działają w ramach otwartych licencji (np. MySQL, postgreSQL). Z ochrony korzystają producenci działający na terenie Rzeczypospolitej Polskiej, ale również obywatele Unii Europejskiej.

Część właścicieli stron wykorzystuje dodatkowe zabezpieczenia związane z ograniczeniem możliwości *scrapowania*. Najpopularniejszym tego typu rozwiązaniem, które występuje w sieci jest *CAPTCHA* i jej następca *reCAPTCHA*. Celem istnienia tego mechanizmu jest sprawdzenie za pomocą prostego zadania czy dany odbiorca strony jest robotem czy człowiekiem (Kosiński, 2017). Różnica pomiędzy wersją re, a zwykłą polega na samym zadaniu — w przypadku zwykłej *CAPTCHY* trzeba przepisać jakiś fragment tekstu, bądź zaznaczyć odpowiednie obrazki, zgodnie z zadanym opisem. *reCAPTCHA* charakteryzuje się tym, że jedyne co musi zrobić użytkownik, to kliknąć w kwadracik, obok tekstu, mówiącego o tym, że nie jest on robotem. Różnice można zobaczyć na rysunku 1.3.



Rysunek 1.3. Test reCAPTCHA i CAPTCHA

Źródło: opracowanie własne na podstawie (Amlen, 2018) i (Sachdeva, 2018)

Z samej definicji mechanizmu CAPTCHA można wywnioskować, że jest on wysoce skuteczną obroną przed ekstrakcją danych. Oczywiście warto zaznaczyć, że często wyskakujące testy są irytujące dla odbiorców strony, stąd też administrator musi zna-

leźć optimum pomiędzy zabezpieczeniem swojej strony, a odpowiednim *User experience* związanym z jej użytkowaniem. Problem ten został częściowo rozwiązany przez *reCAPTCHA*, jednakże klikanie w nią nadal bywa uciążliwe, w szczególności, gdy zależy nam na czasie.

Nawet jeżeli wydaje nam się, że strona nie posiada żadnego zabezpieczenia dotyczącego ekstrakcji danych to nie powinniśmy wykonywać tej czynności nieostrożnie. Częste odwiedzanie strony i zapychanie serwera może zostać uznane przez administratora za próbę ataku. W razie wystąpienia takowej sytuacji skontaktuje się on z dostawcą Internetu osoby *scrapującej* w celu wyjaśnienia problemu. Symptomami świadczącymi o tym, że ktoś może nie życzyć sobie tak intensywnego ruchu na swojej stronie jest wyskakiwanie różnych błędów.

Aspekty prawne związane z ekstrakcją danych nie są w pełni unormowane, ze względu na ciągły rozwój Internetu i zasobów elektronicznych. Niemniej jednak realizując duże projekty komercyjne należy zawsze zasięgnąć opinii specjalisty z dziedziny prawa, w celu uniknięcia problemów. Istotną stroną są narzędzia, najczęściej *open source* wykorzystywane do procesu *web scrapingu*. Temat ten został rozwinięty w następnym rozdziale.

Rozdział 2

Współczesne narzędzia web scrapingu

2.1 Selenium jako narzędzie do ekstrakcji danych

Jednym z najpopularniejszych narzędzi używanych do ekstrakcji danych jest Selenium stworzone w 2004 roku przez Jasona Hugginsa (Wikipedia, 2019c).

Rysunek 2.1. Logo Selenium

Źródło: (Brands of the World, 2017)

Zgodnie z informacją podaną na stronie wydawcy Selenium możemy zdefiniować jako narzędzie, które umożliwia automatyzację korzystania z przeglądarek internetowych. Jego bazową funkcją jest testowanie oprogramowania i jest elementarnym narzędziem w pracy dla testera. Ze względu na swoją budowę, nie ogranicza się tylko i wyłącznie do tego celu — to od użytkownika zależy do czego wykorzysta narzędzie (SeleniumHQ, 2019b). Kluczowym aspektem Selenium jest jego dostępność, narzędzia można uży-

wać we współpracy z najpopularniejszymi przeglądarkami (SeleniumHQ, 2019a), takimi jak:

- Firefox,
- Internet Explorer,
- Opera,
- Safari,
- Chrome.

Twórcy Selenium dostosowują swój produkt do ciągłego rozwoju przeglądarek, w celu dotarcia do jak najszerszego grona odbiorców. Warto zaznaczyć, że oprogramowanie jest udostępnione zgodnie z licencją *Apache License 2.0*, co oznacza, że korzystanie z niego jest w pełni darmowe. Narzędzie to nie tylko dostosowuje się do większości używanych przez użytkowników przeglądarek, ale jest również przystosowane do pracy z wieloma językami programowania, na 3 najpopularniejszych systemach operacyjnych (*Microsoft Windows*, *Apple OS X* i *Linux*). Zgodnie z informacjami zawartymi na stronie wydawcy(SeleniumHQ, 2019a), języki programowania, na których można pracować przy użyciu Selenium to:

- C sharp (*framework*: NUnit),
- Haskell,
- Java (*framework*: JUnit, TestNG),
- JavaScript (*framework*: WebdriverJS, WebdriverIO, NightwatchJS, NemoJS),
- Objective-C,
- Perl,
- PHP (*framework*: Behat + Mink),
- Python (*framework*: unittest, pyunit, py.test, robot framework),
- R,

- Ruby (*framework*: RSpec, Test::Unit).

Selenium w *web scrapingu* używane jest przede wszystkim jako narzędzie, które umożliwia naśladowanie ruchu użytkownika po stronie. Programista jest w stanie automatycznie kliknąć w jakiś przycisk, skorzystać z zawartej na stronie wyszukiwarki w celu wpisania określonego hasła bądź dostać się do informacji ukrytej w formularzu (Taracha, 2017). Dzięki swoim możliwościom Selenium umożliwia poruszanie się po stronie, bez konieczności zmieniania linku. Oznacza to, że programista nie jest ograniczony do pobierania danych ze stron posiadających ściśle określoną strukturę linku. Warto pamiętać, że w ramach naśladowania ruchu użytkownika, nie należy zbyt szybko poruszać się pomiędzy podstronami. Systemy antybotowe są wyczulone na zbyt szybkie, wręcz nieludzkie zmiany podstron. Rozwiązaniem tego problemu jest ustawienie tak zwanego stopu, który spowalnia proces jednocześnie zwiększając jego bezpieczeństwo i zmniejsza obciążenie serwera strony, po której się poruszamy. Oprócz możliwości swobodnego i automatycznego poruszania się po stronie, Selenium daje możliwości ekstrakcji elementów ze strony na podstawie wielu selektorów. Od preferencji użytkownika narzędzia zależy na które rozwiązanie się zdecyduje. Poznanie wszystkich metod nie jest konieczne by wykonać pojedyncze zadanie, jednak bardzo często jedna z nich jest szybsza i prostsza w stosunku do innych. Pisząc regularnie *scrapery* należy zapoznać się z jak największą liczbą możliwych rozwiązań problemu, by uniknąć żmudnej i niepotrzebnej pracy. Spis selektorów dostępnych w Selenium prezentuje się następująco:

- ClassName — wykorzystuję nazwę klasy elementu, jeżeli jest on opisany przez parę klas to Selenium można znaleźć więcej niż jeden element. Trzeba dostosować wyszukiwanie do potrzeb.
- Id — funkcjonuje na tej samej zasadzie co ClassName, z tym że najczęściej jest unikalne.
- Name — funkcjonuje na tej samej zasadzie co ClassName.
- TagName — lokalizujemy element po jego tagu html, np. możemy wyszukać listę wybieraną.
- LinkText — metoda zwracająca link, musimy podać cały tekst by zadziałała.

- PartialLinkText — różni sie od poprzedniczki tym, że można wprowadzić jedynie fragment tekstu (Sądel, 2019b).
- XPath — to inaczej *XML Path Languag*, czyli język do opisu ścieżek. W celu wybrania danego elementu możemy podać ścieżkę absolutną, albo relatywną. Metoda ta traktuje dokument HTML jak drzewo (Sądel, 2019c).
- CSS — kaskadowe arkusze stylów, pozwalają na zdefiniowanie styli, czyli odpowiadają za sposób wyświetlania elementów HTML na stronie. Znak "#" w CSS oznacza id, za to "." klasę (Sądel, 2019a).

Najpopularniejszym pakietem do używania Selenium w R jest RSelenium (Harrison, 2019). *Scrapowanie* z tym pakietem jest znacznie prostsze jeżeli używamy jakiejś nakładki do przeglądarki, która pozwala nam na prostsze wyszukiwanie odpowiednich selektorów (Brooks, 2014). Współczesne przeglądarki posiadają wbudowane opcje umożliwiające zajrzenie w kod strony, jednakże wydobycie selektora z takiego narzędzia jest najczęściej trudniejsze i bardziej czasochłonne, z drugiej strony zmniejsza zależności w projekcie. W pythonie najczęściej używanym pakietem do *web scrapingu* przy użyciu Selenium jest po prostu "Selenium". Bardzo często używa się go wraz z pakietem "*Beautiful Soup*"(Gray, 2018), który zostanie szerzej omówiony w następnym podrozdziale.

2.2 Biblioteki *Beautiful Soup* i *Scrapy* - sposób na ekstrakcję w Pythonie

Jednym z najpopularniejszych języków używanych podczas wykonywania analiz danych i szeroko pojętego *data science* jest Python. Jego zaletą jest relatywnie prosta konstrukcja i duża ilość bibliotek dostosowana do rozwiązywania najpopularniejszych problemów pojawiających się podczas prowadzenia badań jak i tworzenia rozwiązań komercyjnych. Python jest językiem darmowym i można w nim pisać przy użyciu różnych IDE (*Integrated Development Environment*). Użytkownik musi sam dokonać wyboru w zależności od specyfiki projektu i prywatnych preferencji. W kwestii *web scrapingu* warto pochylić się nad dwoma bibliotekami, które powstały by ułatwić ekstrakcję danych: *Beautiful Soup* i *Scrapy*.

2.2.1 Beautiful Soup

Beautiful Soup został napisany przez Leonarda Richardsona (Wikipedia, 2019a) w 2009 roku (Richardson, 2019a). Najnowsza wersja pakietu została wydana w 2019 roku. Nazwa biblioteki pochodzi od piosenki, która pojawiła się w powieści Lewis Carroll pt. ”Alicja w Krainie Czarów”. Biblioteka nie posiada jednolitego logo, jednak na oficjalnej stronie autora i w różnych wersjach dokumentacji zawsze powiązana jest z obrazkiem pochodzących z powieści.

Rysunek 2.2. Obrazek znajdujący się w obecnej wersji dokumentacji *Beautiful Soup*

Źródło: (Richardson, 2019b)

Mechanizm ekstrakcji danych przy użyciu *Beautiful Soup* jest dość prosty. Pierwszym krokiem, powinno być wybranie strony, którą chcemy *scrapować*. Po dokonaniu wyboru przypisujemy adres URL strony do zmiennej korzystają z funkcji: *requests.get(url).text*, która pochodzi z biblioteki *requests*. Następnie tworzymy obiekt korzystając z funkcji *BeautifulSoup*, ważne aby najpierw pobrać i załadować bibliotekę *BeautifulSoup*. Argumentami tejże funkcji są wcześniej przypisany URL (możemy wprowadzić nazwę zmiennej) i typ wybranego *parsera*. Pod postacią *parsera* rozumiemy metodę, przy pomocy której pobierzemy dane ze strony. Szybsze wersje są najczęściej zależne od C, pełny

spis dostępny jest w dokumentacji (Richardson, 2019b). Jeżeli chcemy zobaczyć w jaki sposób prezentuje się drzewo wewnątrz obiektu powinniśmy skorzystać z funkcji *Prettify*, którą łączymy ze zwykłym *printem* (Chaudhary, 2018). Dalsze użytkowanie pakietu jest silnie zależne od potrzeb i konstrukcji strony, którą chcemy *scrapować*. Bardzo często struktury danych będą przybierać podobne formy, stąd też warto znać funkcje, które wraz z odpowiednimi zmiennymi pozwolą dostać się do większości danych:

- soup.find - pozwala na znalezienie jakiejś klasy wewnątrz skryptu HTML. Może być to np. tabela. Najlepszym rozwiązaniem jest zapisanie znaleziska do osobnego obiektu.
- findAll - pozwala na znalezienie wszystkich elementów. Najlepiej najpierw przypisać do zmiennej odpowiedni fragment strony przy pomocy funkcji "find", a następnie wyszukać wszystkie elementy potrzebne do badań.

2.2.2 Scrapy

Scrapy został napisany przez grupę deweloperów w 2008 roku (Scrapy, 2019a), projekt jest wykonywany zgodnie ze standardami *open source*, tak więc korzystanie z produktu jest darmowe. Obecny kod jest dostępny za pośrednictwem repozytorium GitHub nazwanego *scrapy* (Scrapy repository, 2019).

Rysunek 2.3. Logo Scrapy

Źródło: (Scrapy, 2019b)

Scrapy to Pythonowa biblioteka, która dostarcza wszystkich niezbędnych narzędzi do dokonania procesu *web scrapingu*. Umożliwia ekstrakcje danych, przetwarzanie ich i przechowywanie w wybranym przez programiście formacie. Pierwszym krokiem koniecznym do użycia biblioteki jest jej zainstalowanie. Instalacja tak jak w przypadku

wszystkich rozwinięć Pythona jest zależna od posiadanej przez nas wersji i IDE. Charakterystyczną rzeczą dla tego narzędzia jest działania na zasadzie *crawlera*. Podstawową funkcją, która pozwala dostać się programowi na stronę jest *fetch*. By zobaczyć gdzie znajduję się nasz *crawler* należy skorzystać z funkcji *view*, bazowo obiekt, który należy przeglądać nosi nazwę *"response"*. Po użyciu funkcji *view* z odpowiednim argumentem otworzy się strona, na którą dostał się nasz pająk. Tak samo jak pozostałe pakiety *Scrapy* korzysta z selektorów css w celu wydobycia informacji ze strony internetowej. Ponownie przydatne stają się rozszerzenia dla przeglądarek, która ułatwiają przeglądanie kodu strony. Chcąc znaleźć odpowiednie elementy, korzystamy z funkcji *css* odwołując się do elementu "response". By je wydobyć musimy na sam koniec polecenia dodać funkcję *extract*, która wydobywa wszystkie elementy zgodne z zadanym schematem, bądź *extract_first*, która wydobywa tylko pierwszy element, na który natrafi. Poprzez kontrolę selektorów css i odpowiedni dobór funkcji jesteśmy wstanie zapisywać różne informacje w dogodnym dla nas formacie.

Większe projekty przy użyciu *Scrapy* polegają na tworzeniu pająków sieciowych, którym przydziela się różne zadania. Funkcją odpowiedzialną za to zadanie jest *genspider*. Za jej pomocą można stworzyć *crawlera* i nadać mu nazwę. Bazowo crawel jest przypisany do jakiejś domeny i tylko w ramach niej może pobierać dane, jego startowy URL znajduję się w tejże domenie. Dodatkowo wewnątrz klasy *crawlera* pojawia się funkcja *parse*, w której to musimy zawszeć wszystkie dane dotyczące *scrapowanego* materiału. Tak jak w przypadku ręcznego *scrapowania* zapisujemy je poprzez określone selektory css. By uruchomić pająka korzystamy z funkcji *crawl* (Sanad Zaki Rizvi, 2017).

2.3 Pakiet *Rvest* - sposób na ekstrakcję w R

Drugim dominującym językiem w dziedzinie *data science* jest R. Do mocnych stron R w kwestii analizy danych należą liczne pakiety, które ułatwiają rozwiązanie popularnych problemów i znajdują się na repozytorium o nazwie CRAN. Zaletą R jest jeden stabilny i wspierany IDE - R Studio. Dodatkowym atutem w kwestii *data science* jest fakt iż język ten został napisany przez statystyków, a nie informatyków. R jak i R Studio są w pełni darmowe. Najpopularniejszym pakietem służącym do *web scrapingu* przy użyciu R jest napisany w 2014 roku przez Hadleya Wickhama rvest (Wickham, 2019).

Rysunek 2.4. Logo rvest

Źródło: (tidyverse, 2019)

Podstawowym krokiem podczas korzystania z rvest jest zainstalowanie pakietu i załadowanie go za pomocą funkcji *library*. Dalej należy zapisać adres URL, do strony którą mamy zamiar scrapować w postaci obiektu. Ważne by zapisywany link umieścić w nawiasach. Następnym krokiem tak jak w przypadku większości scraperów jest zapoznanie się z kodem strony, korzystając z wbudowanych narzędzi przeglądarkowych bądź zainstalowanych dodatków. W celu odczytania kodu HTML ze strony korzystamy z funkcji *read_html*, której argumentem jest wcześniej zapisany adres URL. W celu wydobycia czegoś po określonym selektorze css korzystamy z funkcji *html_nodes*. Dalej możemy sprecyzować co dokładnie ma być wynikiem, jeżeli to tekst, to korzystamy z funckcji *html_text* (Kaushik, 2017). Całą operację wydobycia danego elementu najlepiej zapisać w ramach jednej linijki kodu z użyciem przetwarzania potokowego. Dodatkowo korzystając z operatora ”%>%” możemy od razu dokonać podstawowych przekształceń na wydobytych danych, np. zmienić ich typ z tekstowych na numeryczne. W celu uzyskania atrybutów z tagów korzystamy z funkcji *html_attr*, przykładowo jeżeli chcemy pozyskać linki to argumentem tejże funkcji będzie wyrażenie "*href*", a jeżeli naszym celem jest pozyskanie ID to argumentem będzie "*id*". Wyjątkowo użyteczną i przyśpieszającą pracę funkcją dostępną w ramach pakietu rvest jest *html_table*. Jak sama nazwa wskazuje funkcja ta pozwala na ekstrakcję tabel wraz z zawartością. Jeżeli na stronie dodane są tabele html to ich wydobycie i zapisanie w dogodnym formacie zajmuję tylko parę linijek kodu. Dodatkowo rvest oferuje symulację wyszukiwania przy użyciu funkcji *html_session* (Treadway, 2019).

Rozdział 3

Analiza rynku imprez fanowskich na terenie Polski — zastosowanie web scrapingu

3.1 Opis rynku imprez fanowskich w polsce

Imprezy fanowskie zwane też konwentami (od angielskiego *convention*) zdefiniować można jako kilku dniowe, okresowe zjazdy fanów określonej dziedziny kultury. W trakcie trwania zjazdu organizowane są spotkaniami ze specjalistami i twórcami, prezentacje twórczości artystów związanych z tematem imprezy, wystawy i konkursy *cosplay* (Lisowska-Magdziarz, 2017). Konkursy w wewnętrznej nomenklaturze często są określane mianem "wiedzówek", a spotkania z różnego rodzaju interesującymi osobami, które są specjalistami w danej dziedzinie nazywane są "panelami". Panele możemy podzielić zgodnie z kryterium partycypacji uczestników na:

- Wykładowe — uczestnicy przez większość wystąpienia biernie słuchają prelegenta, który oddaje im głos w sekcji pytań. Czas przeznaczony na pytania nie powinien trwać dłużej niż 10 do 20 procent całości wystąpienia.
- Dyskusyjne — uczestnicy biorą czynny udział w panelu, odpowiadają na pytania zadane przez prelegenta i rozwiązują swoje własne wątpliwości poprzez wymianę zdań z innymi uczestnikami jak i prelegentem. Część osób biernie przysłuchuję się dyskusji w celu rozwinięcia swojej wiedzy na dany temat.

- Mieszane — charakteryzują się pośrednim udziałem uczestnika. W najczęściej występującym schemacie prelegent krótko przybliża omawiany temat, a potem przechodzi do żywej dyskusji. W tej formie istotne jest dobre rozłożenie czasu — przez 50% prelekcji należy prowadzić panel w formie wykładowej, a przez pozostałą połowę w formie dyskusyjnej.

Czas trwania konwentu jest różny, jednak najczęściej jest to okres od jednego do trzech dni. Ze względu na chęć przyciągnięcia jak najszerszej rzeczy fanów imprezy odbywają się najczęściej w weekendy. Część wydarzeń w celu wydłużenia czasu trwania imprezy do czterech dni wykorzystuje ogólnonarodowe święta, które w sposób naturalny wydłużają weekend. Wyjątkiem i pewnym fenomenem są zjazdy trwające tydzień i dłużej. Najczęściej konwenty takie związane z grami fabularnymi i rekonstrukcją. Zjazdy tego typu najczęściej odbywają się na dużej powierzchni, często otwartej (np. Orkon (Orkon, 2019)). Umożliwia to uczestnikom odpowiednie odgrywanie różnego rodzaju bitew i wcielanie się w przeróżne postaci, które są częścią określonych sesji.

W przypadku konwentów organizowanych na terenie polski znaczna większość imprez od lat odbywa się na terenie szkół. Od paru lat można zauważyć trend, w którym to organizatorzy większych zjazdów decydują się na wynajem obiektów targowych (np. Poznańskie MTP) lub uczelni wyższych (np. Uniwersytet Przyrodniczy we Wrocławiu). Cechą charakterystyczną polskich imprez konwentowych jest konieczność zapewnienia uczestnikom noclegu, bardzo często znajdującego się na terenie samej imprezy. Z tego powodu część sal szkolnych (bądź hal targowych) przekształcane jest w tzw. *sleep roomy*, w których nie odbywają się atrakcje, a jedynie organizowane jest miejsce do spania. Z biegiem czasu, ze względu na swoją popularność część atrakcji znalazła stałe miejsce na konwentach. Niezależnie od typu imprezy i jej wielkości bardzo często można się natknąć na przynajmniej jedną z wymienionych poniżej rozrywek:

- Karaoke — najczęściej realizowane przy użyciu komputera poprzez darmowe oprogramowanie *UltraStar*, z udziałem mikrofonów z popularnej gry na konsolę *Play Station – Sing Star*.

- Maty do tańczenia — w środowisku określane skrótem DDR (od ang. *Dance Dance Revolution*). Są podłączone do komputera i zsynchronizowane ze specjal-

nym oprogramowaniem

- Sala z konsolami — często występuje w dwóch wersjach: współczesnej i retro. Warto zauważyć, że wraz z upływem czasu coraz więcej sprzętu zalicza się do drugiej kategorii.

- Gry planszowe i karciane — na imprezach organizowane jest pomieszczenie, w którym można zagrać w rozmaite gry. Najczęściej określane jest mianem *games roomu*, poza samą dostępnością różnych tytułów na terenie pomieszczenia znajduję się osoba, która zna zasady wielu gier i pomoże nowym osobom się wdrożyć.

- Konkurs *cosplay* — polega na wykonaniu jak najdokładniejszego stroju, zgodnego z załączonym obrazkiem referencyjnym i odegraniu krótkiego przedstawienia, mającego odzwierciedlać charakter danej postaci. Odbywa się na scenie, najczęściej ulokowanej w sali gimnastycznej. Organizacja konkursu wiążę się z zaproszeniem niezależnego jury, które wyłoni zwycięzców w wybranych przez organizatorów kategoriach.

Konwenty można podzielić zgodnie z grupą docelową fanów, do których mają dotrzeć. Podział ten był wyjątkowo widoczny na początku XXI wieku, obecnie zaczyna zanikać, ze względu na ciągle rosnące wymagania uczestników. Wraz z rewolucją internetową dostęp do wielu elementów popkultury stał się znacznie ułatwiony, w efekcie obecni fani poszerzali swoje zainteresowania, a nowi od razu nie zamykali się w określonych ramach. Do tematów, które na ten moment zyskały znaczącą popularność i posiadają liczne imprezy zaliczamy:

- szeroko pojętą fantastykę — spotkania fanów związanych z takimi uniwersami jak LOTR (*Lord of the Rings*) czy SW (*Star Wars*). Konwenty fantastyczne bardzo często są prekursorami rozwiązań organizacyjnych stosowanych na innych imprezach tego typu.

- kulturę japońską — zjazdy najczęściej nawiązują do japońskiej kultury popularnej i tworzonych przez Japończyków komiksów (*mangi*) i animacji (*anime*).

- gry planszowe, karciane i fabularne — zjazdy związane z wszelaką formą papierowej rozrywki.

- gry komputerowe — imprezy skupiające wokół siebie fanów elektronicznej rozgrywki, często łączone z turniejami.

Część imprez określa się jako mieszane. Oznacza to, że impreza będzie starała się dotrzeć do paru grup docelowych. Należy jednak mieć na uwadze, że nawet jeżeli cała impreza jest definiowana jako fantastyczna to bardzo często znajdują się na jej terenie pojedyncze sale związane z mangą oraz anime i na odwrót. Nie oznacza to, że imprezę od razu należy traktować jako mieszaną, bardzo często organizatorzy w nazwach bądź regulaminach swoich wydarzeń określają z jaką grupą docelową chcą być kojarzeni i kto jest ich zdaniem głównym odbiorcą.

Konwenty są najczęściej tworzone przez społeczności związane z daną subkulturą, które nazywane są fandomami. Fandom definiujemy na dwa sposoby, po pierwsze jako społeczność fanów, związaną z określonymi elementami popkultury (np. fantastyką, mangą i anime, kreskówkami itd.). Po drugie fandom to twórcza i partycypacyjna konsumpcja upodobanego przez siebie tekstu medialnego (Wikipedia, 2019b). Od strony prawnej zloty są najczęściej organizowane przez stowarzyszenia (np. Klub Fantastyki „Druga Era"), posiadają one prezesa, zarząd, a członkowie muszą uczestniczyć w zebraniach walnych. Część organizatorów decyduję się na założenie fundacji, a część korzysta z jednoosobowej działalności gospodarczej. Od paru lat na rynku pojawiają się wydarzenia organizowane przez większych graczy, najczęściej są to spółki odpowiedzialne za obiekty targowe (np. PTAK WARSAW EXPO Sp. z o.o.). Mimo ich obecności na rynku organizowane przez nich imprezy bardzo często nie są postrzegane przez społeczność fanowską jako konwenty. Wpływ na taki stan rzeczy ma czysto komercyjny charakter części imprez organizowanych na terenie targów. Charakterystyczną rzeczą tzw. *Expo* jest zrezygnowanie z przestrzeni do spania na terenie samych targów, *sleep roomy* są oddalone od terenu właściwego imprezy. Część nowoczesnych imprez na wzór konwentów zachodnich w ogóle odsuwa się od zapewnienia noclegu swoim uczestnikom.

W związku z rozwojem branży konwentowej na terenie polski pojawiły się strony internetowe i redakcje specjalizujące się w medialnej obsłudze tego typu zdarzeń. W obrębie zadań tych podmiotów mieści się:

- tworzenie relacji pisemnych z imprez,

- tworzenie relacji zdjęciowych z imprez,
- nagrywanie kluczowych punktów programu danej imprezy,
- prowadzenie i uzupełniania kalendarza konwentów.

Ze względu na ostatnią z wymienionych funkcji portale tego typu są świetnym źródłem niedostępnych nigdzie indziej danych, które umożliwią wykonanie podstawowej analizy rynku imprez fanowskich w Polsce.

3.2 Opis źródeł danych wybranych do ekstrakcji.

W celu dokonania ekstrakcji danych należy wybrać strony, które posiadają odpowiednie informacje, mogące się przydać w prowadzonym badaniu bądź projekcie komercyjnym. W celu znalezienia odpowiedniego źródła konieczne jest skorzystanie z wiedzy eksperckiej, bądź odkrycie jakiegoś rankingu, który klasyfikowałby portale. Po wybraniu odpowiednich stron niezbędne jest zapoznanie się z ich strukturą. Kluczową kwestią na samej stronie wydaję się znalezienie podstron, zawierających potrzebne dane. Branża imprez fanowskich rozwijała się wraz z Internetem toteż od początku jej istnienia możemy natrafić na strony, które prowadzą spisy dotyczące wydarzeń. Obecnie najpopularniejszą formą takiego spisu są kalendarze, w ramach których osoby odpowiedzialne za dane portale regularnie umieszczają informacje na temat konwentów. Obecnie rynek jest zdominowany przez dwie strony tego typu: Konwenty Południowe (Konwety Południowe, 2019) i Informator Konwentowy (Informator Konwentowy, 2019). Sama nazwa portali wskazuje na ich silne powiązanie z branżą imprez fanowskich. Pierwsza ze stron posiada dane na temat imprez odbywających się na terenie polski od 2013 roku. Istotne jest to, że nie można mieć całkowitej pewności, że dany portal dokonał pełnego spisu, stąd też warto korzystać z minimum dwóch uzupełniających się źródeł. W ramach kalendarza prowadzonego przez redakcję konwentów południowych możemy uzyskać informację na temat:

- data – uwzględniająca dzień rozpoczęcia i dzień zakończenia, w przypadku imprez jednodniowych podany jest dany dzień,
- nazwa,

- zakładana wielkość wydarzenia – udostępniona w postaci grafiki, podawana przedziałowo,
- miasto,
- kategoria – z jakim fandomem związany jest dany konwent, strona wyróżnia następujące kategorię:
  - fantastykę,
  - mange&anime,
  - gry planszowe,
  - mieszane,
  - inne,
  - gry/e–sport,
  - larp,
  - komiks,
  - serial,
  - książka.

W przypadku Informatora konwentowego dane są gromadzone od 2002 roku, a informacje które można uzyskać to:

- nazwa,
- miejscowość,
- data – uwzględniająca dzień rozpoczęcia i dzień zakończenia, w przypadku imprez jednodniowych podany jest dany dzień,
- dni – na ile dni planowane jest dane wydarzenie,
- cena – bardzo czesto zawiera braki danych, dodatkowo brakuje jednakowej struktury,
- strona – link do strony wydarzenia,

- tematyka – informacja z jakim fandomem związany jest dany konwent, strona wyróżnia następujące kategorię:
  - fantastykę,
  - mange&anime,
  - larp,
  - planszówki,
  - rpg,
  - gaming,
  - retro,
  - demoscena,
  - muzyka,
  - turnieje.

Poza informacjami zawartymi w kalendarzach strony te posiadają wiele innych przydatnych danych, które mogłyby być częścią następnego badania. Już po kliknięciu na nazwę określonej imprezy jesteśmy wstanie uzyskać dodatkowe dane na jej temat. Dodatkowo z wielu imprez prowadzone są foto–relacje, które mogłyby posłużyć do bardziej zaawansowanych analiz (np. próby określenia przeciętnego wieku uczestników w danej grupie). Ze względu na ograniczony zakres pracy w niniejszym opracowaniu kluczową sprawą będzie ekstrakcja samych kalendarzy i wyciągnięcia podstawowych wniosków na temat branży konwentów w Polsce.

3.3 Ekstrakcja danych ze stron specjalizujących się w branży imprez fanowskich

Proces ekstrakcji należy rozpocząć od zapoznania się z kodem strony. Analizując oba portale można zauważyć, że nie będzie koniecznie zastosowanie narzędzi pomagających omijać zabezpieczenia napisane w *JavaScript*, tak więc całość procesu zostanie dokonana przy użyciu języka programowania R za pomocą pakietu rvest. W pierwszej

kolejności należy zainstalować potrzebne pakiety, operację tę wykonujemy raz na danym urządzeniu. Możemy skorzystać z funkcji *install.packages*, bądź użyć wbudowanego w R Studio narzędzia. Wszystkie pakiety używane podczas ekstrakcji są dostępne na CRAN. Poza pakietem rvest użyte zostały następujące narzędzia:

- tidyverse — pakiet używany do przetwarzania danych, wyjątkowo użyteczny ze względu na możliwość przetwarzania potokowegoi zwiększenie przejrzystości kodu.
- stringr — pakiet, który ułatwia przetwarzanie poszczególnych zmiennych tekstowych.
- lubridate — pakiet używany do przetwarzania dat, będzie potrzebny by ustalić długość trwania imprez z kalendarza Konwentów Południowych.
- openxlsx — pakiet, który pozwoli w prosty sposób weksportować dane do excela i umożliwić korzystanie z nich osobom nieznających środowiska R.

Po instalacji pakietów w celu ich załadowania należy zastosować funkcję *library*, operację załadowania trzeba powtarzać zawsze na początku pracy z kodem w języku R.

W celu rozpoczęcia ekstrakcji potrzebujemy gotowej listy z linkami. Korzystając z załączonego kodu (”Stworzenie listy url”) jesteśmy wstanie uzyskać jednolitą listę. W pierwszej kolejności w zmiennej *url* zapisujemy fragment linku, który powtarza się w latach 2017-2019. Warto zauważyć, że pobranie dotyczy tylko i wyłącznie archiwum konwentów, w celu pozyskania wydarzeń, które są zapowiedziane na przyszłość konieczne byłoby dodanie jeszcze jednego linku. W następnym kroku tworzymy listę linków uzupełniając każdy o odpowiednią końcówkę zawierającą rok. Następnie powtarzamy proces dla lat 2014-2016, podział na dwie listy jest konieczny ze względu na zmianę, która pojawiła się w linkach (przed rokiem 2017 w linku nie występował ciąg znaków "konwentow-"). Na samym końcu łączymy obie listy w celu uzyskania jednolitego zbioru zawierającego potrzebne linki. Uzyskaną listę przedstawia Rysunek 3.1.

Name	Type	Value
lista_url	list [6]	List of length 6
[[1]]	character [1]	'https://konwenty-poludniowe.pl/konwenty/kalendarz/archiwum-2014'
[[2]]	character [1]	'https://konwenty-poludniowe.pl/konwenty/kalendarz/archiwum-2015'
[[3]]	character [1]	'https://konwenty-poludniowe.pl/konwenty/kalendarz/archiwum-2016'
[[4]]	character [1]	'https://konwenty-poludniowe.pl/konwenty/kalendarz/archiwum-konwentow-2017'
[[5]]	character [1]	'https://konwenty-poludniowe.pl/konwenty/kalendarz/archiwum-konwentow-2018'
[[6]]	character [1]	'https://konwenty-poludniowe.pl/konwenty/kalendarz/archiwum-konwentow-2019'

Rysunek 3.1. Lista linków ze strony Konwenty Południowe

Źródło: opracowanie własne.

Do ekstrakcji danych z wybranego linku stworzono specjalną funkcję o nazwie "wczytaj_tabele_kon_pol". Zastosowana metoda polega na załadowaniu odpowiedniego linku, po czym znalezieniu na stronie elementu #kalendarz i pobraniu całej tabeli html przy użyciu funkcji *html_table*. Dalsza część funkcji istnieje w celu uporządkowania danych. Po pierwsze usuwamy pustą kolumnę, która nie jest nazwana w tabeli na stronie, znajdują się w niej obrazki z przewidywaną ilością ludzi. Po drugie dzielimy kolumnę *Data* na dwie, jedna zawiera datę początku imprezy, a druga końca. Następnie przekształcamy nowo utworzone kolumny *Początek* i *Koniec* do formatu daty. Na samym końcu za pomocą funkcji *filter* usuwamy imprezy, które zostały odwołane. Fragment tabeli będącej wynikiem zastosowania funkcji przestawia Rysunek 3.2

	Poczatek	Koniec	Nazwa.wydarzenia.konwentu	Miasto	Kategoria
1	2019-01-12	NA	Yukicon VII	Opole	Manga & Anime
2	2019-02-01	2019-02-03	Zjava - Zimowa Jazda Avangardowa 10	Warszawa	Gry planszowe
3	2019-02-02	2019-02-03	Aishiteru 2019	Wrocław	Manga & Anime
4	2019-02-08	2019-02-10	10. Konwent Fantastyki Fantasmagoria	Gniezno	Fantastyka
5	2019-02-09	NA	Gostkon 2019	Gostyń	Fantastyka
6	2019-02-09	2019-02-10	Gakkon 5 Edycja Zimowa	Łódź	Manga & Anime
7	2019-02-09	NA	Cosplayowy Bal Walentynkowy	Łódź	Inne
8	2019-02-15	2019-02-17	KOLA - Konferencja Larpowa 2019	Wrocław	LARP
9	2019-02-22	2019-02-24	ESL One Katowice 2019 - Dota 2 (+ IEM Expo)	Katowice	Gry/E-Sport
10	2019-02-23	NA	Fun Fest 2019	Poznań	Fantastyka

Rysunek 3.2. Dziesięć pierwszych obserwacji z kalendarza Konwentów Południowych

Źródło: opracowanie własne.

W celu pobrania informacji z Informatora Konwentowego skorzystamy z tych samych pakietów. Początkowy etap ekstrakcji jest identyczny, tworzymy listę adresów url, dzięki której swobodnie będziemy wstanie pobrać dane z wybranego roku. Dodatkowo w przypadku tego portalu nie zachodzi żadna zmiana w linku, dlatego kod potrzebny do wykonania tej operacji jest krótszy. Wynik tej operacji przedstawia Rysunek 3.3

Name	Type	Value
url_ik_2014_2019	list [6]	List of length 6
[[1]]	character [1]	'https://konwenty.info/konwenty/?rok=2014'
[[2]]	character [1]	'https://konwenty.info/konwenty/?rok=2015'
[[3]]	character [1]	'https://konwenty.info/konwenty/?rok=2016'
[[4]]	character [1]	'https://konwenty.info/konwenty/?rok=2017'
[[5]]	character [1]	'https://konwenty.info/konwenty/?rok=2018'
[[6]]	character [1]	'https://konwenty.info/konwenty/?rok=2019'

Rysunek 3.3. Lista linków ze strony Informator Konwentowy

Źródło: opracowanie własne.

Ekstrakcja danych ze strony Informator Konwentowy wymaga większego wysiłku, ze względu na brak tabeli w formacie html. Każda kolumna importowana jest do R w postaci tekstu. Potem z zebranych wektorów zawierających zmienne w postaci tekstu tworzony jest *data frame*. Podczas procesu tworzenia nadawane są nazwy, dodatkowo upewniamy się, że kolumny nie będą przyjmowały typu *factor*. Utworzona tabela zawiera wiele elementów, które przeszkadzają w jej dalszym przetwarzaniu. By umożliwić dalsze prowadzenie badań należy uporządkować zbiór, w pierwszym kroku usunięte zostają frazy, które powtarzają się w wybranych kolumnach i nie są przydatne w dalszym badaniu. Na samym końcu usunięte zostają dwa pierwsze wiersze, które zawierają nieprzydatne informacje.

W celu zaprezentowania użyteczności pobranych danych, wykonana została analiza konwentów odbywających się w 2019 roku. Analiza obejmuje imprezy fanowskie, które miały już miejsce i takie, które są planowane. Źródłem jest kalendarz publikowany przez Konwenty Południowe. Pierwszym krokiem, umożliwiającym dokonanie takowej analizy jest złączenie zbiorów imprez, które już się odbyły i takich, które dopiero się odbędą. Dokonano tego za pomocą fragmentu kodu nazwanego ”połączenie

imprez odbywających się w 2019 roku". Wynikiem operacji jest *data frame*, posiadający 5 kolumn i 151 wierszy, oznacza to, że w 2019 roku strona Konwenty Południowe umieściła w swoim kalendarzu 151 imprez. Istotną kwestią jest sprawdzenie, w którym mieście odbyło się najwięcej imprez fanowskich. Pierwsze 10 obserwacji zostało przedstawionych na Rysunku 3.4.

	Miasto	n
1	Kraków	14
2	Wrocław	10
3	Warszawa	9
4	Łódź	7
5	Poznań	7
6	Katowice	5
7	Opole	4
8	Rybnik	4
9	Toruń	4
10	Częstochowa	3

Rysunek 3.4. Liczba imprez z podziałem na miasta w 2019 roku

Źródło: opracowanie własne.

Najpopularniejszym miastem pod względem liczby imprez w 2019 roku jest Kraków. Pierwsze 6 pozycji zajmują miasta wojewódzkie, zaliczające się do największych miast w kraju. Ponad 44% imprez organizowanych jest na terenie 10 najpopularniejszych miast. Ciekawą obserwacją jest Rybnik, znajdujący się na 7 pozycji wraz z Opolem i Toruniem. Po dokładniejszym zbadaniu tej miejscowości okazuję się, że odbywają się tam trzy imprezy związane z manga oraz anime, które organizuje jedna firma. Konwenty odbywające się w Rybniku przedstawia Rysunek 3.5.

	Poczatek	Koniec	Nazwa.wydarzenia.konwentu	Miasto	Kategoria
22	2019-03-16	2019-03-17	Aicon 2019	Rybnik	Manga & Anime
63	2019-06-12	2019-06-13	E-sportowe Mistrzostwa Szkół Rybnik	Rybnik	Gry/E-Sport
84	2019-07-13	2019-07-14	Mochicon 2019	Rybnik	Manga & Anime
147	2019-11-23	2019-11-24	Tsuru Japan Festival 2019	Rybnik	Manga & Anime

Rysunek 3.5. Imprezy odbywające się w Rybniku w 2019 roku

Źródło: opracowanie własne.

Następną rzeczą, którą warto przeanalizować jest ilość konwentów znajdująca się w danej kategorii. Pozwoli to ustalić, jaki fandom organizuje najwięcej imprez fanowskich w 2019 roku. Dokonano tego za pomocą kodu zaprezentowanego w ”kategorie konwentów”. Pierwsze 5 obserwacji znajduje się w tabeli reprezentującej wynik zastosowanego kodu, którą przedstawia Rysunek 3.6:

	Kategoria	n
1	Fantastyka	47
2	Manga & Anime	23
3	Gry planszowe	18
4	Gry/E-Sport	15
5	Inne	13

Rysunek 3.6. Liczba konwentów z podziałem na określone kategorie w 2019 roku

Źródło: opracowanie własne.

Zdecydowanie najwięcej imprez związanych jest z najstarszym istniejącym na terenie Polski fandomem – fantastyką. Prawie 30% wszystkich imprez organizowanych w kraju jest związanych z tą tematyką. Drugie miejsce zajmują imprezy związane z mangą oraz anime. Warto zaznaczyć, że imprez związanych z kulturą japońską jest o ponad 50% mniej niż tych związanych z fantastyką. W następnym kroku zbadano ile trwają konwenty na terenie Polski uwzględniając podział na poszczególne kategorię fandomowe. Dane dostępne na stronie Konwenty Południowe umożliwiają wyliczenie czasu trwania poszczególnej imprezy fanowskiej na podstawie daty rozpoczęcia i zakończenia. W tym celu do podstawowej tabeli dodano kolumnę ”dlugosc”, następnie dokonano operacji podliczenia i pogrupowania zgodnie z odpowiednimi zmiennymi. Wynik tychże operacji dla 5 najliczniejszych pod względem długości typów imprez przedstawia Rysunek 3.7

	dlugosc	Kategoria	n
1	3	Fantastyka	23
2	2	Manga & Anime	14
3	2	Fantastyka	13
4	2	Gry planszowe	13
5	1	Fantastyka	8

Rysunek 3.7. Długość trwania imprez z podziałem na kategorie w 2019 roku

Źródło: opracowanie własne.

Najbardziej popularne okazały się 3 dniowe fantastyczne konwenty. Warto zaznaczyć, że w tej kategorii znajdują się największe pod względem liczby odwiedzających imprezy na terenie Polski (np. Pyrkon, Copernicon, Falkon). Taką formę imprezy przyjęło ponad 15% konwentów i 49% konwentów fantastycznych. Co ciekawe drugie miejsce na liście zajmują dwu dniowe konwenty związane z mangą oraz anime (trwające najczęściej od soboty do niedzieli). Ponad 60% imprez tego typu trwa dwa dni. Poza analizą długości imprez z podziałem na określone kategorię przeanalizowano jaka długość imprezy fanowskiej była najpopularniejsza w Polsce.

	dlugosc	n
1	2	56
2	3	52
3	1	27
4	4	8
5	7	3
6	5	1
7	6	1
8	9	1
9	10	1
10	15	1

Rysunek 3.8. Długość trwania imprez fanowskich na terenie Polski w 2019 roku

Źródło: opracowanie własne.

Zgodnie z informacjami zawartymi na Rysunku 3.8 najwięcej imprez, bo aż 56

trwało dwa dni (sobota–niedziela). Następne w kolejce są imprezy trzydniowe, tych w 2019 roku było o 4 mniej, a więc 52. Średni czas trwania konwentu to 2,6 dnia. Pierwsze dwa miejsca w tej kategorii to ponad 71% wszystkich konwentów. Najdłuższa impreza w Polsce trwała 15 dni i jest to wcześniej wspomniany Orkon, który należy do konwentów z kategorii LARP. Wszystkie trzy najdłużej trwające imprezy kwalifikują się do tej kategorii.

Podsumowanie

Web scraping jest prężnie rozwijającą się techniką, która jest powszechnie używana w nauce jak i biznesie. Rewolucja Internetowa i rozwój portali społecznościowych wpłynęły na udostępnianie ogromnej ilości nieustrukturyzowanych danych. W efekcie kluczowe jest zastosowanie metod, które umożliwią badaczom wykorzystanie tych danych. W pracy została ustrukturyzowana definicja *web scrapingu* i przedstawiono popularną metodę jej zastosowania, przy uwzględnieniu ram prawnych. Podjęto też zagadnienia związane z pająkami Internetowymi, które często są mylone ze *scraperami*.

Data science to prężnie rozwijająca się dziedzina biznesu i nauki, w której dominującymi narzędziami są R i Python. Stąd też w ramach narzędzi używanych do *web scrapingu* opisano pakiety napisane właśnie w tych językach. Dodatkowo uwzględniono użycie narzędzia do testowania - Selenium, jest ono niezwykle przydane w ekstrakcji stron, które zawierają zabezpieczenia napisane w języku *JavaScript*. W pracy przedstawiono zagadnienia, które umożliwią skuteczne zastosowanie wyżej wymienionych narzędzi w praktyce *web scrapingu*.

Rozdział trzeci zawiera nowatorskie definicje i opisy związane z tematyką imprez masowych, tzw. konwentów. Są to miejsca spotkań przeróżnych fandomów, często imprezy te są tworzone od fanów dla fanów i przyciągają tysiące odbiorców. Tematyka ta jest wyjątkowo rzadko poruszana w nauce, a zyskuję znaczną popularność. W wyniku pracy dokonano ekstrakcji danych pochodzących z dwóch portali zajmujących się tematyką imprez fanowskich. Strony tego typu to jedyne źródło z którego można uzyskać duże zbiory danych związanych z konwentami. W wyniku badania dane zostały pobrane ze strony i wstępnie przeczyszczone, zgodnie z przyjętymi zasadami. Format, w którym umieszczone zostały dane jest powszechnie stosowany w *data science*, umożliwia również eksport informacji do *Excela*. Dokonano wstępnej analizy, na podstawie, której ustalono, że w 2019 roku na stronie Konwenty Południowe dodano 151 imprez.

Najwięcej konwentów odbyło się w Krakowie, dominującą kategorią okazały się imprezy fantastyczne trwające trzy dni (od piątku do niedzieli).

Bibliografia

Abramowicz, W., Opałka, J., Skołowska, W., Kubaczyk, M., Fabisz, K. & Hossa, T. (2015). Automatyczna analiza wydźwięku opinii o operatorach energetycznych jako element wsparcia podejmowanych decyzji. *Studia Ekonomiczne. Zeszyty Naukowe Uniwersytetu Ekonomicznego w Katowicach*, *243*.

Amlen, D. (2018). *What the Heck Is That?: CAPTCHA*. https://www.nytimes.com/2018/02/26/crosswords/what-the-heck-is-that-captcha.html.

Brands of the World. (2017). *Selenium Test Automation Logo / Icon*. https://www.brandsoftheworld.com/logo/selenium-1.

Brooks, A. (2014). *Scraping with Selenium*. https://www.r-bloggers.com/scraping-with-selenium. R-bloggers.

Castrillo-Fernández, O. (2015). Web Scraping: Applications and Tools.

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

Chaudhary, S. (2018). *Web Scraping Wikipedia Tables using BeautifulSoup and Python*. https://medium.com/analytics-vidhya/web-scraping-wiki-tables-using-beautifulsoup-and-python-6b9ea26d8722. Medium.

Coffin, J. (2010). *crawler vs scraper*. https://stackoverflow.com/questions/3207418/crawler-vs-scraper.

Czech, M. (2016). *Co to jest analiza sentymentu oraz jak możesz ją wykorzystać?* https://blog.brand24.pl/co-to-jest-analiza-sentymentu-oraz-jak-mozesz-ja-wykorzystac/.

Eising, P. (2017). *What exactly IS an API?* https://medium.com/@perrysetgo/what-exactly-is-an-api-69f36968a41f.

Gray, D. (2018). *Better web scraping in Python with Selenium, Beautiful Soup, and pandas*. https://www.freecodecamp.org/news/better-web-scraping-in-python-with-selenium-beautiful-soup-and-pandas-d6390592e251. freecodecamp.

Harrison, J. (2019). *Package 'RSelenium'*. https://cran.r-project.org/web/packages/RSelenium/RSelenium.pdf.

Informator Konwentowy. (2019). *Informator Konwentowy*. https://konwenty.info/.

Kaushik, S. (2017). *Beginner's Guide on Web Scraping in R (using rvest) with hands-on example*. https://www.analyticsvidhya.com/blog/2017/03/beginners-guide-on-web-scraping-in-r-using-rvest-with-hands-on-knowledge. Analytics Vidhya.

Konwety Południowe. (2019). *Konwenty Południowe*. https://konwenty-poludniowe.pl/.

Kosiński, D. (2017). *Google nie zapyta cię już, czy jesteś robotem*. https://www.spidersweb.pl/2017/03/captcha-recaptcha-google.html.

Lindenberg, F. (2011). *Getting Data from the Web*. https://datajournalism.com/read/handbook/one/getting-data/getting-data-from-the-web.

Lisowska-Magdziarz, M. (2017). *Fandom dla początkujących. Część I Społeczeństwo i wiedza*. Instytut Dziennikarstwa, Mediów i Komunikacji SpołecznejUniwersytetu Jagiellońskiego.

Matusiak, J. (2012). Ekstrakcja i agregacja zawartości Stron internetowych na przykładzie portali pracy. *zeszyty naukowe uniwersytetu Szczecińskiego studia informatica*, *30*, 59–73. http://wneiz.pl/nauka_wneiz/studia_inf/30-2012/si-30-59.pdf.

Mooney, S., J, Westreich, J. D. & El-Sayed, A., M. (2015). Epidemiology in the Era of Big Data. *Epidemiology*. doi:10.1097/EDE.0000000000000274

Ooms, J. (2019). *Package 'pdftools'*. https://cran.r-project.org/web/packages/pdftools/pdftools.pdf.

Orkon. (2019). *Strona główna konwentu Orkon*. http://orkon.org/pl/.

Ozorowski, J. (2017). *Dobre i złe roboty internetowe – przykłady oraz ich lista*. https://mobiletry.com/blog/dobre-i-zle-roboty-internetowe.

Richardson, L. (2019a). *Beautiful Soup*. https://code.launchpad.net/beautifulsoup. launchpad.

Richardson, L. (2019b). *Beautiful Soup Documentation*. https://www.crummy.com/software/BeautifulSoup/bs4/doc/. crummy.

Sądel, E. (2019a). *Selektory w Selenium – CSS*. https://testelka.pl/selektory-w-selenium-css. testelka.

Sądel, E. (2019b). *Selektory w Selenium – najprostsze metody lokalizowania elementów na stronie*. https://testelka.pl/selektory-w-selenium-najprostsze-metody. testelka.

Sądel, E. (2019c). *Selektory w Selenium – XPath*. https://testelka.pl/selektory-w-selenium-xpath/?fbclid=IwAR2eTxlAUpYrqnPSNLxL8cy0gVtkcyeR-T3d2DLpC0fY8IcMfZAfWWZamMs. testelka.

Sachdeva, A. (2018). *Google Replaces reCAPTCHA Checkboxes With Behavioural Analysis*. https://fossbytes.com/google-replaces-recaptcha-with-behavioral-analysis/.

Sanad Zaki Rizvi, M. (2017). *Web Scraping in Python using Scrapy (with multiple examples)*. https://www.analyticsvidhya.com/blog/2017/07/web-scraping-in-python-using-scrapy/. Analytics Vidhya.

Scrapy. (2019a). *Release notes*. https://doc.scrapy.org/en/latest/news.html.

Scrapy. (2019b). *scrapy*. https://scrapy.org.

Scrapy repository. (2019). *scrapy*. https://github.com/scrapy/scrapy.

SeleniumHQ. (2019a). *Platforms Supported by Selenium*. https://www.seleniumhq.org/about/platforms.jsp.

SeleniumHQ. (2019b). *What is Selenium?* https://www.seleniumhq.org/.

Statista. (2019). *Number of monthly active Facebook users worldwide as of 2nd quarter 2019 (in millions)*.
https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/.

Talar, S. & Kos-Łabędowicz, J. (2014). Internacjonalizacja w warunkach gospodarki internetowej na przykładzie firm micromultinationals. *Biznes międzynarodowy w gospodarce globalnej 2014*, *85*, 571–582. doi:10.4467/23539496IB.13.042.2427

Taracha, R. (2017). *Introduction to Web Scraping using Selenium*.
https://medium.com/the-andela-way/introduction-to-web-scraping-using-selenium-7ec377a8cf72. Medium.

tidyverse. (2019). *rvest logo*.
https://rvest.tidyverse.org/logo.png.

Treadway, A. (2019). *BeautifulSoup vs. Rvest*.
https://www.r-bloggers.com/beautifulsoup-vs-rvest/. R-bloggers.

Ustawa o ochronie baz danych. (2001). Ustawa z dnia 27 lipca 2001 r. o ochronie baz danych (Dz.U. 2001 nr 128 poz. 1402).
http://prawo.sejm.gov.pl/isap.nsf/DocDetails.xsp?id=WDU20011281402.

WebHarvey. (2018). *What is Web Scraping?*
https://www.webharvy.com/articles/what-is-web-scraping.html.

Wickham, H. (2019). *Package 'rvest'*.
https://cran.r-project.org/web/packages/rvest/rvest.pdf.

Wikipedia. (2019a). *Beautiful Soup (HTML parser)*.
https://en.wikipedia.org/wiki/Beautiful_Soup_(HTML_parser).

Wikipedia. (2019b). *Fandom*.
https://pl.wikipedia.org/wiki/Fandom.

Wikipedia. (2019c). *Selenium (software)*.
https://en.wikipedia.org/wiki/Selenium_(software).

Zhao, B. (2017). Web Scraping. *Encyclopedia of Big Data*. doi:10.1007/978-3-319-32001-4_483-1

Spis tabel

Spis rysunków

Dodatek A

Kody języka R

Listing A.1. Wczytanie pakietów

```
library(rvest)
library(tidyverse)
library(stringr)
library(lubridate)
library(openxlsx)
```

Listing A.2. Stworzenie listy url

```
url <- ("https://konwenty-poludniowe.pl/konwenty/kalendarz/archiwum-
    konwentow-")

urls_od_2017 <- sapply(2017:2019, function(x) {
  paste0(url, x) }) %>%
  as.list()

url_2 <- ("https://konwenty-poludniowe.pl/konwenty/kalendarz/archiwum-")

urls_do_2017 <- sapply(2014:2016, function(x) {
  paste0(url_2 , x) }) %>%
  as.list()

lista_url <- append(urls_do_2017, urls_od_2017)
```

Listing A.3. Funkcja na pobieranie tabeli ze strony Konwenty Południowe

```
wczytaj_tabele_kon_pol <- function (adres) {

              read_html(adres) %>%
              html_nodes("#kalendarz") %>%
              html_table %>%
              as.data.frame()%>%
              select (-c(Nazwa.wydarzenia.konwentu.1)) %>%
              separate(Data, c("Poczatek", "Koniec"), " - ") %>%
              mutate(Poczatek=as.Date(Poczatek, "%d.%m.%Y"),
                     Koniec = as.Date(Koniec, "%d.%m.%Y")) %>%
              filter(!str_detect(Nazwa.wydarzenia.konwentu, "(odwolany)"))

}
```

Listing A.4. Stworzenie listy url ze strony Informator Konwentowy

```
url_ik <- ("https://konwenty.info/konwenty/?rok=")

url_ik_2014_2019 <- sapply(2014:2019, function(x) {
                    paste0(url_ik, x) }) %>%
                    as.list()
```

Listing A.5. Pobranie danych ze strony Informator Konwentowy jako *stringi*

```
kon_info_nazwa <- read_html(url_ik_2014_2019[[6]]) %>%
              html_nodes(".tabik_nazwa") %>%
              html_text()

kon_info_miejscowosc <- read_html(url_ik_2014_2019[[6]]) %>%
              html_nodes(".tabik_miejscowosc") %>%
              html_text()

kon_info_data <- read_html(url_ik_2014_2019[[6]]) %>%
              html_nodes(".tabik_data") %>%
              html_text()

kon_info_dni <- read_html(url_ik_2014_2019[[6]]) %>%
              html_nodes(".tabik_dni") %>%
              html_text()

kon_info_rodzaj <- read_html(url_ik_2014_2019[[6]]) %>%
              html_nodes(".tabik_rodzaj") %>%
              html_text()

kon_info_cena <- read_html(url_ik_2014_2019[[6]]) %>%
              html_nodes(".tabik_cena") %>%
              html_text()
```

Listing A.6. Stworzenie tabeli i uporządkowanie jej

```
informator_df <- data.frame(Nazwa = kon_info_nazwa,
                  Miejscowosc = kon_info_miejscowosc,
                  Data = kon_info_data,
                  Dni = kon_info_dni,
                  Tematyka = kon_info_rodzaj,
                  Cena = kon_info_cena,
                  stringsAsFactors = FALSE)

informator_df$Miejscowosc <-informator_df$Miejscowosc %>%
                            str_remove("Miejscowosc':")

informator_df$Data <- informator_df$Data %>%
                      str_remove("Termin:")

informator_df$Tematyka <- informator_df$Tematyka %>%
                          str_remove("Rodzaj:")

informator_df$Cena <- informator_df$Cena %>%
                      str_remove("Cena:")

informator_df <- informator_df[-c(1,2),]
```

Listing A.7. Połączenie imprez obywających się w 2019 roku

```
bylo_2k19 <- wczytaj_tabele_kon_pol(as.character(lista_url[6]))

aktualne_kony_url <- "https://konwenty-poludniowe.pl/konwenty/kalendarz"

bedze_2k19 <- wczytaj_tabele_kon_pol(aktualne_kony_url)
```

```
kon_2k19 <- merge(x = bylo_2k19, y = bedze_2k19, all = TRUE) %>%
                                  arrange(Poczatek) %>%
                                  subset(Poczatek < "2020-01-01")
```

Listing A.8. Konwenty a miasta

```
miasto_2019 <- kon_2k19 %>%
               group_by(Miasto) %>%
               count(Miasto, sort = T)
```

Listing A.9. Kategorie konwentów

```
kategorie_2019 <- kon_2k19 %>%
                  group_by(Kategoria) %>%
                  count(Kategoria,sort = T)
```

Listing A.10. Długość trwania imprez z podziałem na kategorie

```
konwenty_latami <- kon_2k19 %>%
                   mutate(dlugosc = coalesce(Koniec - Poczatek+1, 1)) %>%
                   group_by(dlugosc) %>%
                   count(Kategoria, sort = T) %>%
                   arrange (desc(n))
```

Listing A.11. Długość trwania imprez

```
konwenty_dlugosc<- kon_2k19 %>%
                   mutate(dlugosc = coalesce(Koniec - Poczatek+1, 1)) %>%
                   group_by(dlugosc) %>%
                   count(dlugosc, sort = T)
```