Dokumentacja projektowa

Podstawy Teleinformatyki
Webscrapper

Piotr Sędrowski Filip Sochal

17 czerwca 2016

Spis treści

1 Wstęp **3**
- 1.1 Opis projektu . . . 3
- 1.2 Używane technologie . . . 3

2 Wymagania **3**
- 2.1 Funkcjonalne . . . 3
- 2.2 Pozafunkcjonalne . . . 4

3 Przypadki użycia **4**
- 3.1 Wyszukiwanie adresów URL oraz artykułów . . . 4
- 3.2 Eksport wyników do pliku tekstowego . . . 4
- 3.3 Eksport wyników do pliku HTML . . . 5
- 3.4 Zmiana głębokości wyszukiwania . . . 5
- 3.5 Wyczyszczenie okna głównego . . . 6
- 3.6 Wczytanie listy słów kluczowych z pliku(plików) . . . 6
- 3.7 Wczytanie listy adresów URL z pliku (plików) . . . 6

4 Diagramy UML **7**
- 4.1 Diagram przypadków użycia . . . 7
- 4.2 Diagramy aktywności . . . 8
- 4.3 Diagram Klas . . . 12

5 Interfejs graficzny **13**
- 5.1 Opis technologii . . . 13
- 5.2 Wyglad interfejsu . . . 13
- 5.3 Opis klas interfejsu . . . 17
  - 5.3.1 Klasa WebscrapperGUI . . . 17
  - 5.3.2 Klasa FileSelector . . . 20

6 Logika aplikacji **20**
- 6.1 Klasa Exporter . . . 20
- 6.2 TextualContent . . . 21
- 6.3 BasicSearchEngine . . . 22
- 6.4 JsoupParser . . . 23
- 6.5 SearchBar . . . 24
- 6.6 SearchEnginePresenter . . . 25

7 Użyte technologie **25**
- 7.1 JSOUP . . . 25

8 Teoria **26**
8.1 Web scraping 26
8.1.1 Wstęp 26
8.1.2 Techniki 27

1 Wstęp

1.1 Opis projektu

Aplikacja Webscrapper umożliwia przeszukiwanie stron internetowych w poszukiwaniu artykułów zawierających określone słowa kluczowe. Posiada obsługę wielu stron i dowolnej liczby słów kluczowych. Dodatkowo umożliwia eksport tych danych do plików HTML oraz tekstowych. Działa na każdym systemie, który posiada środowisko uruchomieniowe Javy w wersji.

1.2 Używane technologie

- Java w wersji 8,
- JavaFX do wykonania interfejsu użytkownika,
- Scene Builder
- Jsoup jako parser stron,
- środowisko programistyczne InteliJ Idea 2016.1.2,
- kontrola wersji - repozytorium Git.

2 Wymagania

2.1 Funkcjonalne

Klient umożliwia:

- Wyszukiwanie linków i artykułów zawierających podane słowa kluczowe na podanych stronach
- Możliwość wyszukiwania w plikach HTML znajdujących się na dysku twardym
- Eksport wyników do pliku tekstowego
- Eksport wyników do pliku HTML(kodowanie Unicode)
- Zmianę głębokości wyszukiwania
- Wczytywanie słów kluczowych z pliku,

- Wczytywanie adresów stron z pliku
- Wczytywanie ścieżek plików z pliku tekstowego
- Czyszczenie zawartości okna

2.2 Pozafunkcjonalne

- graficzny interfejs użytkownika
- aplikacja działa na systemach operacyjnych pozwalających na uruchomienie JRE (Java Runtime Enviroment) w wersji 8
- w przypadku przeszukiwania stron internetowych wymagane połączenie z siecią Internet

3 Przypadki użycia

3.1 Wyszukiwanie adresów URL oraz artykułów

1. Aktor: użytkownik.
2. Scenariusz główny:
   (a) podanie listy adresów URL do przeszukania
   (b) podanie listy słów kluczowych
   (c) opcjonalna zmiana głębokości wyszukiwania
   (d) naciśnięcie przycisku szukaj
   (e) wyświetlenie listy wyników
3. Scenariusz alternatywny:
   (a) wyświetlenie powiadomienia o wprowadzeniu błędnych danych,
   (b) wyświetlenie informacji o braku połączenia z siecią Internet

3.2 Eksport wyników do pliku tekstowego

1. Aktor: użytkownik.
2. Scenariusz główny:
   (a) Wybranie opcji z menu *Edit ->Export to txt*

(b) Podanie nazwy pliku do którego użytkownik chce zapisać wyniki

(c) Plik o podanej nazwie jest tworzony w podanej lokalizacji.

3. Scenariusz alternatywny:

   (a) wyświetlenie powiadomienia o braku praw do zapisu w podanej lokalizacji

.

3.3 Eksport wyników do pliku HTML

1. Aktor: użytkownik.

2. Scenariusz główny:

   (a) Wybranie opcji z menu *Edit ->Export to HTML*

   (b) Podanie nazwy pliku do którego użytkownik chce zapisać wyniki

   (c) Plik o podanej nazwie jest tworzony w podanej lokalizacji.

3. Scenariusz alternatywny:

   (a) wyświetlenie powiadomienia o braku praw do zapisu w podanej lokalizacji

.

3.4 Zmiana głębokości wyszukiwania

1. Aktor: użytkownik.

2. Scenariusz główny:

   (a) wybranie opcji z menu *Edit ->Change depth*

   (b) wpisanie poziomu w nowym oknie

   (c) naciśnięcie przyciku OK

3. Scenariusz alternatywny:
   brak scenariusza alternatywnego

3.5 Wyczyszczenie okna głównego

1. Aktor: użytkownik.
2. Scenariusz główny:
   (a) Wybranie opcji z menu *Edit - >Clear*
   (b) Pola słów kluczowych, adresów URL oraz wyników wyszukiwania zostają wyczyszczone
3. Scenariusz alternatywny:
   brak scenariusza alternatywnego

3.6 Wczytanie listy słów kluczowych z pliku(plików)

1. Aktor: użytkownik.
2. Scenariusz główny:
   (a) Wybranie opcji z menu *File ->Load keywords*
   (b) Naciśnięcie przycisku Open File (Open Files)
   (c) Odnalezienie i zaznaczenie pliku (plików) do wczytania
   (d) Słowa kluczowe zostają wczytane z pliku (plików)
3. Scenariusz alternatywny:
   brak scenariusza alternatywnego

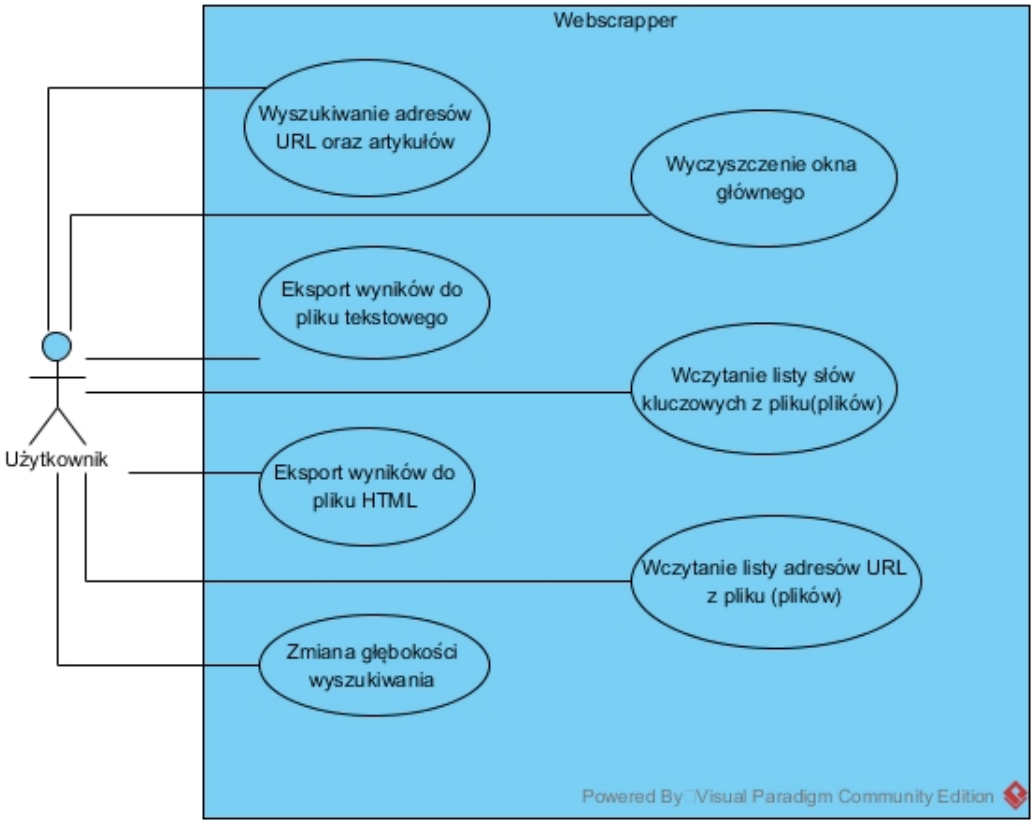
3.7 Wczytanie listy adresów URL z pliku (plików)

1. Aktor: użytkownik.
2. Scenariusz główny:
   (a) Wybranie opcji z menu *File ->Load urls*
   (b) Naciśnięcie przycisku Open File (Open Files)
   (c) Odnalezienie i zaznaczenie pliku (plików) do wczytania
   (d) Adresy URL zostają wczytane z pliku (plików)
3. Scenariusz alternatywny:
   brak scenariusza alternatywnego

4 Diagramy UML

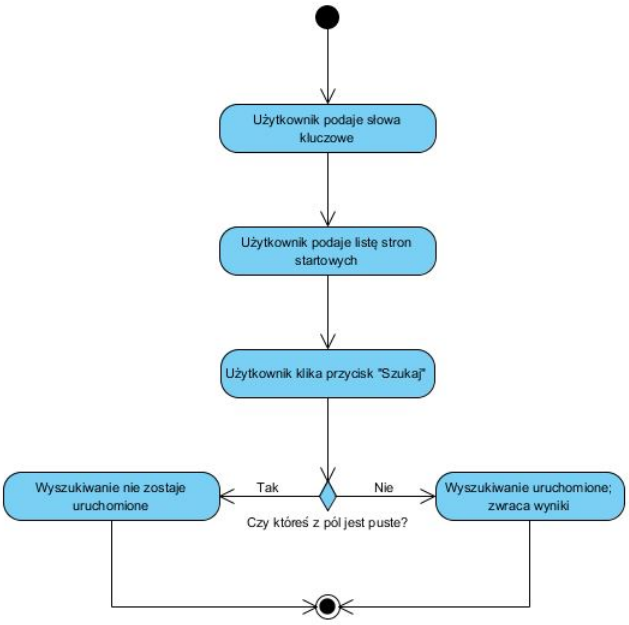
4.1 Diagram przypadków użycia

Diagram przypadków użycia obrazuje połączenia między aktorami i przypadkami uzycia. W tym projekcie wyodrębnilismy siedem przypadków użycia, aktorem we wszystkich z nich jest użytkownik

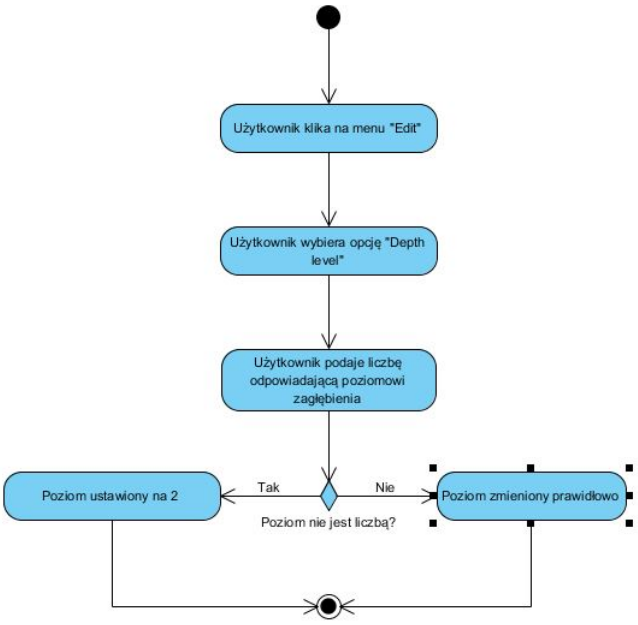


Rysunek 1: Diagram przypadków użycia

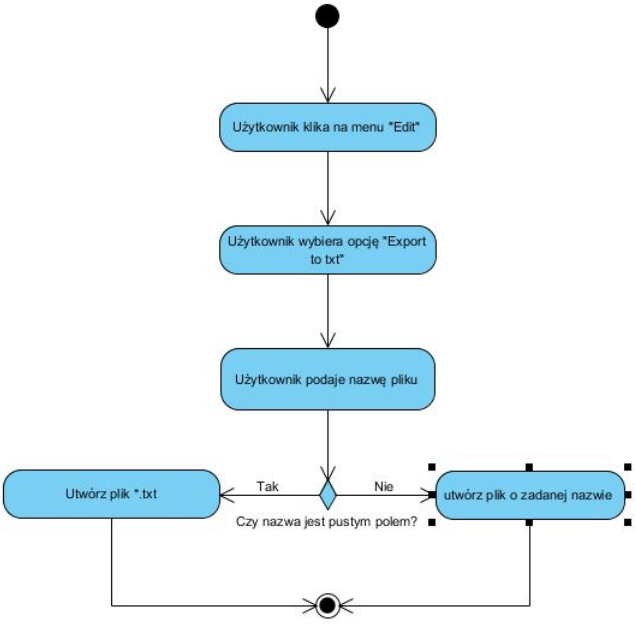
4.2 Diagramy aktywności



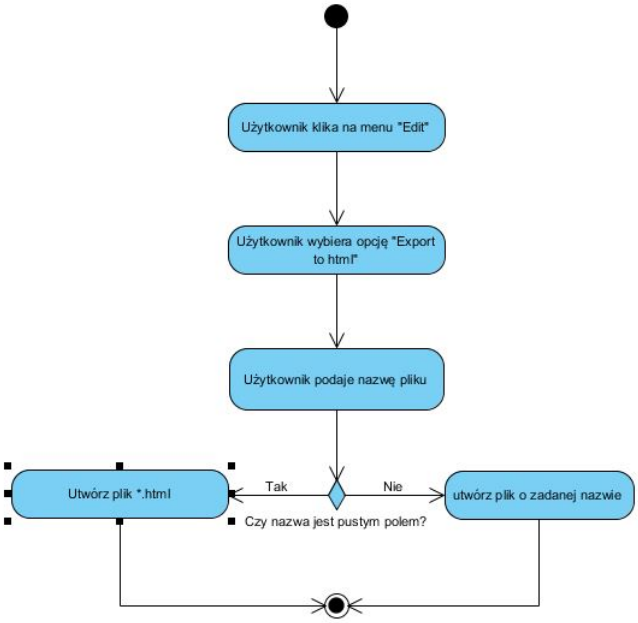
Rysunek 2: Wyszukiwanie artykułów



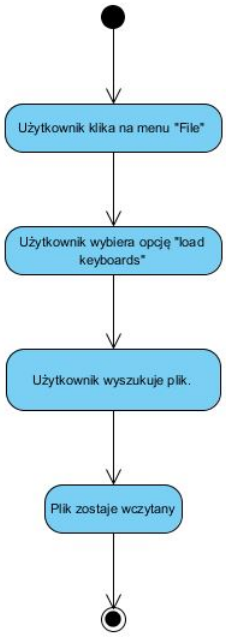
Rysunek 3: Zmiana poziomu głębokości wyszukiwania



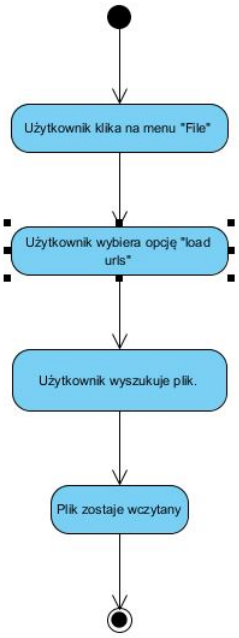
Rysunek 4: Eksport wyników do pliku tekstowego



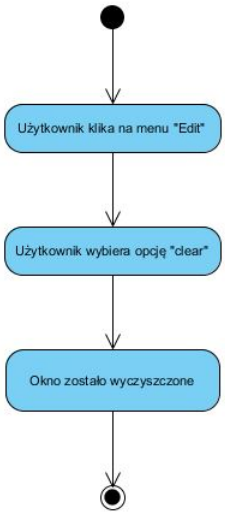
Rysunek 5: Eksport wyników do dokumentu HTML



Rysunek 6: Załadowanie słów kluczowych z pliku

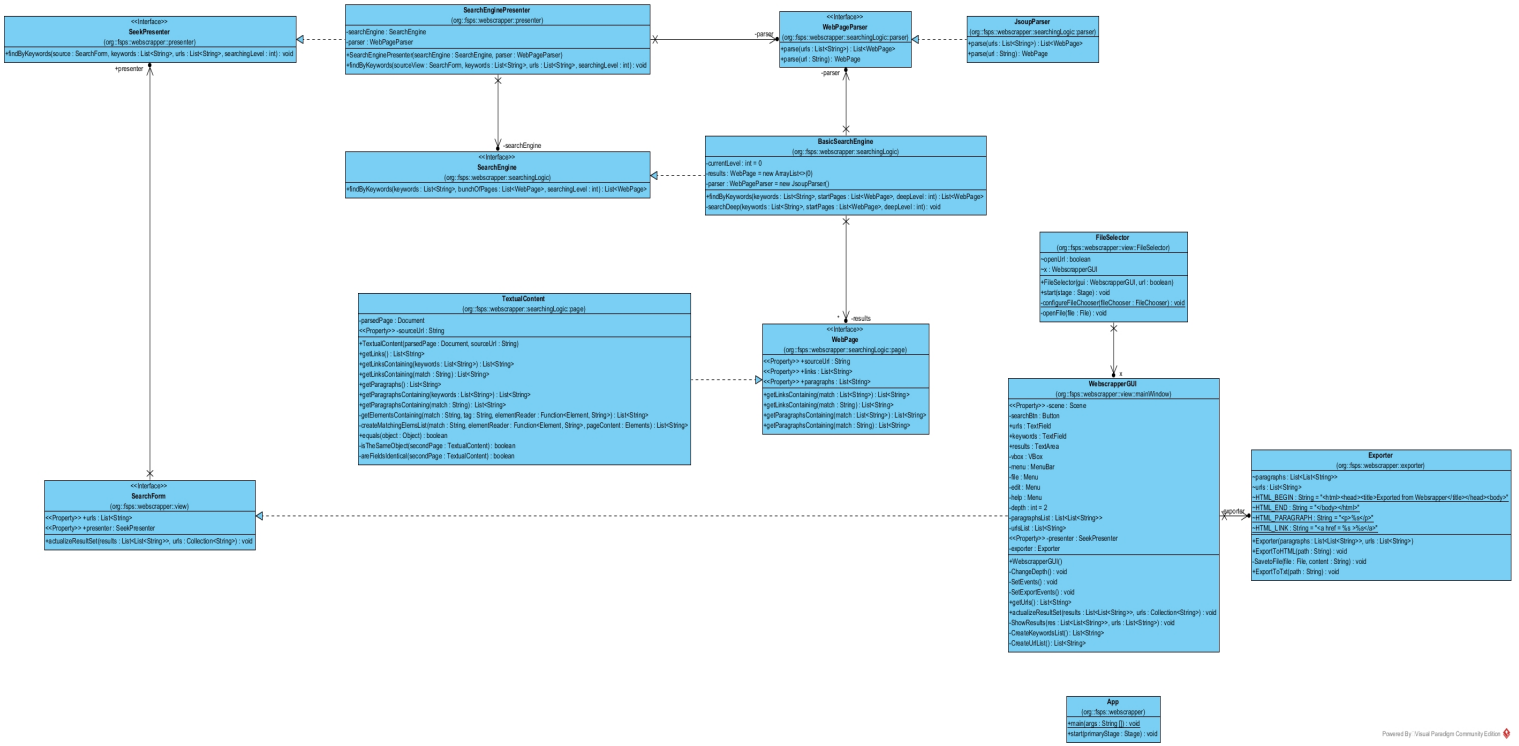


Rysunek 7: Załadowanie hiperłączy z plików



Rysunek 8: Czyszczenie okna aplikacji

4.3 Diagram Klas



Rysunek 9: Diagram klas

5 Interfejs graficzny

Interfejs jest zbudowany w technologii JavaFX, do projektowania wyglądu został użyty program SceneBuilder.

5.1 Opis technologii

JavaFX została opracowana jako następca Swinga, i jest obecnie rekomendowaną technologią tworzenia GUI przez Oracle, domyślnie dołączaną do JDK od wersji 7. JavaFX pozwala na stosunkowo łatwe tworzenie wizualnie efektownych aplikacji, które mogą być uruchamiane jako samodzielne aplikacje na komputerach, jako aplety na stronach internetowych, aplikacje uruchamiane z przeglądarek a także jako programy na urządzeniach mobilnych. Wprowadzono też szereg udogodnień dla programistów i nowych możliwości w tworzeniu GUI, między innymi dotyczących wsparcia materiałów multimedialnych, animacji itd. Pozwala na tworzenie GUI na dwa sposoby - poprzez tworzenie kontrolek od razu w kodzie aplikacji i umieszczaniu ich na panelach, albo poprzez pliki fxml. Pliki FXML mają składnię bardzo podobną do XMLa i zawierają cały opis layoutu aplikacji, który później nalezy wczytać w klasie kontrolera.

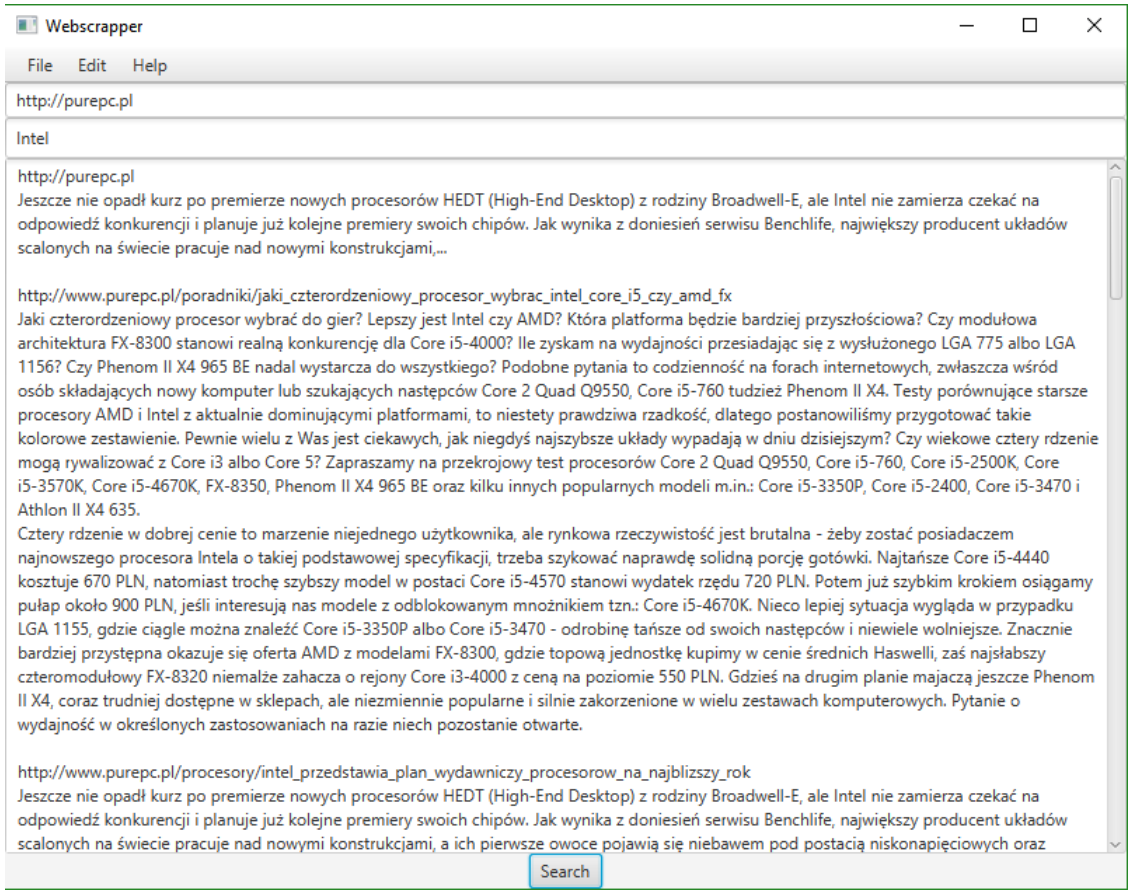
5.2 Wyglad interfejsu

W interfejsie postawiliśmy na przejrzystość oraz prostotę. Na górze okna znajduje się menu, poniżej pole do wpisania adresów url, które mają zostać przeszukane oraz pole do wpisania poszukiwanych słów kluczowych. Największe pole słuzy do wyświetlania rezultatów wyszukiwania. Na dole okna znajduje się przycisk *Search*, który powoduje rozpoczęcie wyszukiwania.

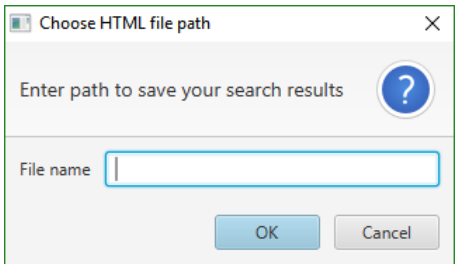
Struktura menu:

- **File**
  - *Load keywords*
    Pozwala na wczytanie listy poszukiwanych słów kluczowych z pliku/plików. Każde słowo kluczowe musi być w nowej lini pliku.
  - *Load urls*
    Pozwala na wczytanie listy adresów URL z pliku/plików. Każdy adres musi być w nowej lini pliku.
  - *Close*
    Zamyka program.

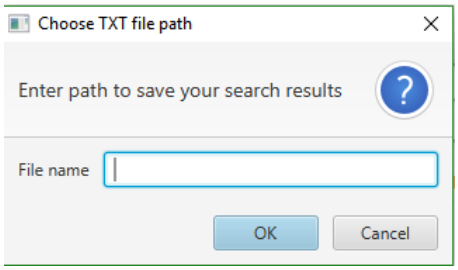
- **Edit**
  - *Clear*
    Czyści listę słów kluczowych, listę adresów URL oraz listę wyników
  - *Depth Level*
    Pozwala na ustawienie, w nowym oknie dialogowym, głębokości przeszukiwania parsera.
  - *Export to txt*
    Pozwala na eksport listy wyników wyszukiwania do pliku tekstowego o podanej nazwie.
  - *Export to HTML*
    Pozwala na eksport listy wyników wyszukiwania do pliku HTML(Kodowanie UTF-8) o podanej nazwie.
- **Help**
  - *About*
    Wyświetla okno z informacjami o programie.



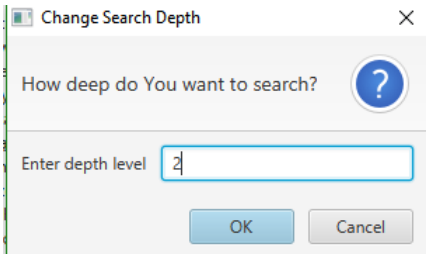
Rysunek 10: Wyglad okna glownego aplikacji



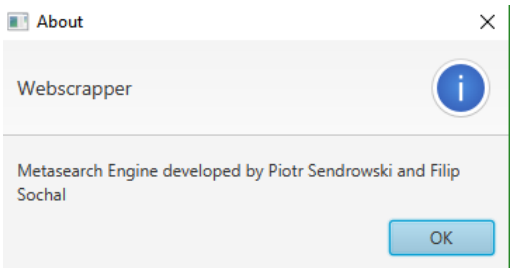
Rysunek 11: Okno eksportu do pliku HTML



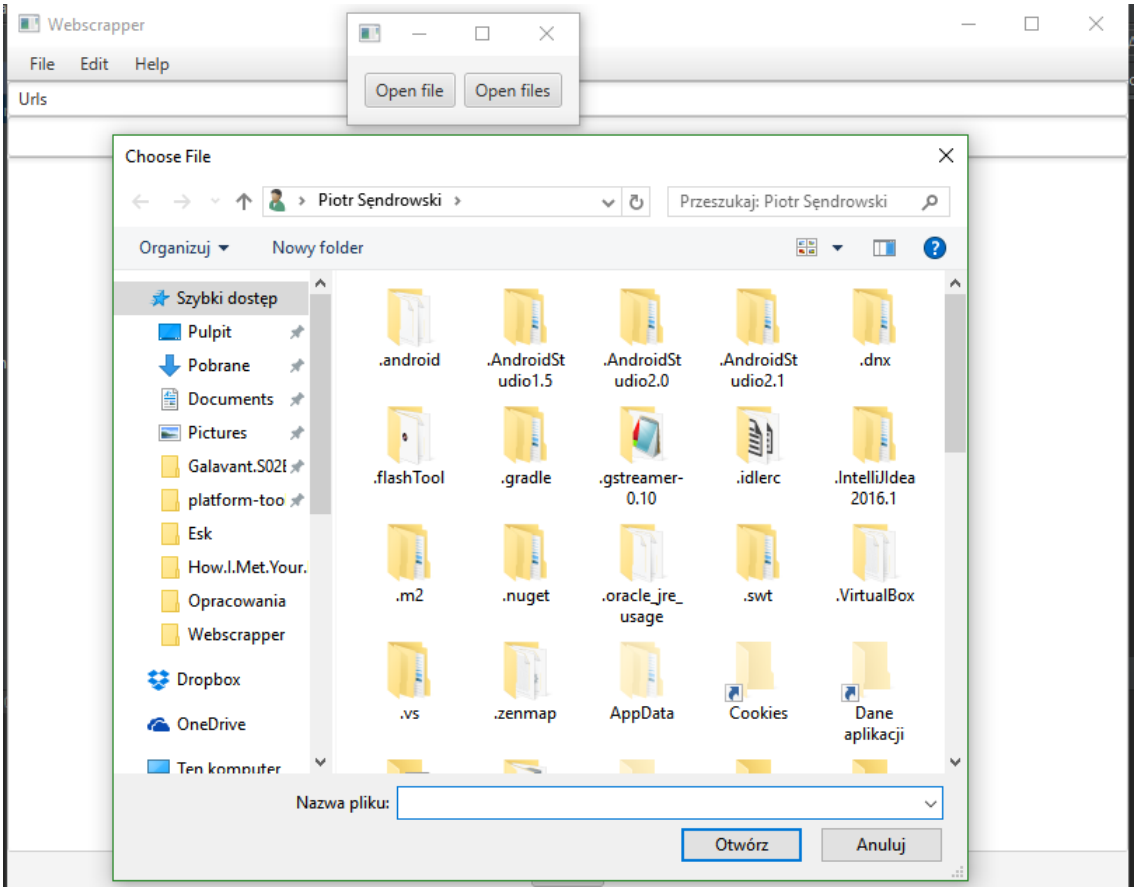
Rysunek 12: Okno eksportu do pliku tekstowego



Rysunek 13: Okno ustawiania głębokości wyszukiwania



Rysunek 14: Okno z informacjami o programie



Rysunek 15: Okno wyboru plików do wczytania

5.3 Opis klas interfejsu

5.3.1 Klasa WebscrapperGUI

Podstawowa klasa aplikacji *WebscrapperGUI* jest kontrolerem głównego okna aplikacji. Odpowiada za wszystkie operacje wykonywane z głównego okna i inicjuje proces wyszukiwania. Klasa posiada pola, które przechowują wszystkie widoczne na ekranie kontrolki, oraz zasoby:

- *searchBtn*
  Pole klasy *Buttton*, przechowuje kontrolkę przycisku *Search*

- *urls*
  Pole klasy reprezentujące obiekt typu *TextField*, kontrolka do której

należy wpisać adresy internetowe do przeszukania w poszukiwaniu słów kluczowych.

- *keywords*
  Pole klasy reprezentujące obiekt typu *TextField*, kontrolka do wpisywania słów kluczowych

- *results*
  Pole klasy reprezentujące obiekt typu *TextArea*, kontrolka do wyświetlania wyników wyszukiwania

- *vbox*
  Pole klasy reprezentujące obiekt typu *VBox*, przechowuje kontrolki *urls* i *keywords*

- *menu*
  Pole klasy reprezentujące obiekt typu *MenuBar*, kontrolka paska górnego menu aplikacji, przechowuje obiekty *file*, *edit* oraz *help*.

- *file*
  Pole klasy reprezentujące obiekt typu *Menu*, kontrolka rozwijanego menu File

- *edit*
  Pole klasy reprezentujące obiekt typu *Menu*, kontrolka rozwijanego menu Edit

- *help*
  Pole klasy reprezentujące obiekt typu *Menu*, kontrolka rozwijanego menu Help

- *presenter*
  Pole klasy reprezentujące obiekt typu *SeekPresenter*, głównej klasy obsługującej wyszukiwanie i prezentację danych

- *depth*
  Pole klasy typu *int*, przechowuje informacje o aktualnym poziomie głębokości wyszukiwania, domyślnie **2**

- *paragraphsList*
  Pole klasy reprezentujące listę list obiektów typu *String*, przechowuje znalezione paragrafy do wyświetlenia

- *urlsList*
  Pole klasy reprezentujące listę obiektów typu *String*, przechowuje adresy url do znalezionych paragrafów

- *exporter*
  Pole klasy reprezentujące obiekt typu *Exporter*, który obsługuje eksport rezultatów wyszukiwania do pliku.

Ponadto, klasa *WebscrapperGUI* jest wyposażona w następujące metody:

- *WebscrapperGUI()*
  Konstruktor klasy służy do załadowania pliku FXML oraz przypisania polom klasy odpowiednich obiektów z ekranu. Na końcu wywołuje metodę *SetEvents()*

- *SetEvents()*
  Metoda pomocnicza, służy do ustawienia zdarzeń dla wszystkich obiektów, które tego wymagają - w tym wypadku wszystkie obiekty menu(z wyłączeniem opcji eksportu) oraz przycisk *Search*

- *SetExportEvents()*
  Kolejna metoda pomocnicza, służąca do ustawienia zdarzeń dla opcji eksportu wyników wyszukiwania do plików HTML oraz tekstowych. Jest wywoływana dopiero po zakończeniu wyszukiwania.

- *actualizeResultSet(List<List<String>>results, Collection<String>urls)*
  Metoda pozwalająca na pobranie rezultatów zakończonego wyszukiwania. Parametr *results* jest listą list znalezionych paragrafów , a parametr *urls* jest po prostu kolekcją adresów url, prowadzących do znalezionych paragrafów.

- *ShowResults(List<List<String>>res, List<String>urls)*
  Metoda wyświetlająca wyniki wyszukiwania, fomatuje otrzymane dane i wpisuje je do odpowiedniego okna.

- *CreateUrlList()*
  Metoda zwracająca listę adresów url do przeszukania, pobiera dane z pola urls i wpisuje je do listy.

- *CreateKeywordsList()*
  Metoda zwracająca listę słów kluczowych, pobiera dane z pola keywords i wpisuje je do listy

5.3.2 Klasa FileSelector

Klasa *FileSelector* jest kontrolerem okna do wyboru wczytywanych plików zawierających słowa kluczowe. Korzysta głównie z dobrodziejstw elementu *FileChooser*. W przeciwieństwie do okna głownego, ta klasa nie posiada pliku FXML, cały interfejs jest zdefiniowany w kodzie klasy. Wyposażona jest w następujące metody:

- *configureFileChooser(final FileChooser fileChooser)*
  Metoda służy do nadania tytułu oknu wyboru plików oraz ustawienia początkowego katalogu.

- *openFile(File file)*
  Metoda obsługuje otwarcie pliku i wyciągnięcie z nich potrzebnych danych(słowa kluczowe/adresy URL)

6 Logika aplikacji

6.1 Klasa Exporter

Klasa *Exporter* służy do eksportowania wyników wyszukiwania do plików HTML oraz TXT. Do tworenia poprawnych plików HTML są wykorzystywane cztery statyczne obiekty typu *String*:

- HTML_BEGIN = "<html><head><title>Exported from Websrapper</title><meta charset=ÜTF-8Ẅ></head><body>",
  Ten łańcuch znaków jest dodawany na początku każdego pliku HTML, tag meta charset gwarantuje poprawne wyswietlanie znaków w przeglądarce

- HTML_END = "</body></html>"
  Ten łańcuch znaków konczy każdy plik HTML.

- HTM_PARAGRAPH = " %s "
  Znaczniki paragrafów, każdy akapit znaleziony w trakcie wyszukiwania jest formatowany za pomocą tej zmiennej, przy użyciu *String.format()*

- HTML_LINK = "%s"
  Znacznik hiperłącza, każdy adres url znaleziony w trakcie wyszukiwania jest formatowany za pomocą tej zmiennej, przy użyciu *String.format()*

Ponadto klasa jest wyposażona w następujące metody:

- *ExportToHTML(String path)*
  Metoda zapisująca wyniki wyszukiwania do pliku HTML. Jako argument przyjmuje ściezkę do pliku, do którego ma wyeksportować dane. Dokonuje odpowiedniego formatowania wyników przed wywołaniem metody *SavetoFile*.

- *ExportToTxt(String path)*
  Metoda zapisująca wyniki wyszukiwania do pliku tekstowego. Jako argument przyjmuje ściezkę do pliku, do którego ma wyeksportować dane. Tutaj również odbywa się wstępne formatowanie przed zapisem, ale mniej złożone niż w metodzie wyżej.

- *SavetoFile(File file, String content)*
  Metoda pomocnicza, realizująca sam zapis do pliku.

6.2 TextualContent

Klasa *TextualContent* jest reprezentacją wyników działania parsera. Zawiera wszystkie dane, jakie znajdują się na stronie, jednakże zawiera metody pozwalające na dostęp jedynie do zawartości tekstowej, tj. hiperłączy oraz akapitów tekstu.

W jej skład wchodzą następujące metody:

- *getSourceUrl()* - zwraca adres domenowy przeanalizowanej pod kątem składniowym strony w postaci łańcucha znaków,

- *getLinks()* - zwraca listę wszystkich odnośników do innych stron (w tym podstron) w postaci listy łańcuchów znaków,

- *getLinksContaining(List<String >keywords*, int urlsLimit) - metoda podobna do powyższej, jednakże z tą różnicą, iż pozwala wprowadzić limity zarówno co do interesujących użytkownika słów kluczowych (argument w postaci listy łańcuców znaków), jak również limit liczby wyników, jakie metoda zwraca. Metoda ta umożliwia wyszukiwanie interesującej Nas zawartości już na poziomie hiperłączy,

- *getLinksContaining(List<String >keywords)* - metoda ta umożliwia wyszukiwanie na stronie hiperłączy zawieraących określone słowa kluczowe (podane w postaci listy łańcuchów znaków), jednakże bez nakładania limitów co do liczby wyników, która to potrafiła przekraczać 1200 rekordów na liście. Zwraca listę łańcuchów znaków z potencjalnie interesującymi odnośnikami,

- *getLinksContaining(String match)* - jest to metoda, która wyszukuje hiperłącza zawierające jedno, konkretne słowo kluczowe podane w postaci łańcucha znaków. Może być używana, jako samodzielna metoda, jednakże jest głównie wykorzystywana przez inną wersję tej metody, wykorzystującą listę łańcuchów znaków ze zbiorem słów kluczowych. Zwraca listę łańcuchów znaków,

- *getParagraphs()* - pozwala na pobranie danych ze wszystkich akapitów właściwego tekstu, jaki znajduje się na stronie. Zwraca listę łańcuchów znaków z całą tekstową zawartością strony,

- *getParagraphsContaining(List <String >keywords)* - jest to metoda, która wyszukuje akapity tekstu zawierające wiele słów kluczowych, podanych jako argument w postaci listy łańcuchów znaków. Jest to właściwa metoda dla wyszukiwania tekstu na stronie internetowej, zwraca listę łańcuchów znaków z potencjalnie interesującą użytkownika treścią,

- *getParagraphsContaining(String match)* - jest to metoda, która wyszukuje konkretne akapity tekstu zawierające jedno, konkretne słowo kluczowe podane w postaci łańcucha znaków. Może być używana, jako samodzielna metoda, jednakże jest głównie wykorzystywana przez inną wersję tej metody, wykorzystującą listę łańcuchów znaków ze zbiorem słów kluczowych. Zwraca listę łańcuchów znaków.

6.3 BasicSearchEngine

Jest to klasa, która odpowiada za wyszukiwanie tekstów na podstawie interesujących użytkownika słów kluczowych i zbioru stron startowych, na podstawie których zaczyna proces szukania.

W jej skład wchodzą następujące metody:

- *findByKeywords(List<String >keywords, List<SearchBar>searchBars)* - jako argumenty przyjmuje kolejno:

  - listę łańcuchów znaków zawierających interesujące użytkownika słowa kluczowe,
  - listę obiektów typu SearchBar reprezentującą zbiór przeanalizowanych pod kątem składniowym stron internetowych, od których użytkownik chciałby zacząć wyszukiwanie.

  Metoda rozpoczyna działanie od znalezienia w zestawie stron startowych wyszukiwarek. Następnie za ich pomocą generuje zapytania (me-

tody typu GET) z zadanymi słowami kluczowymi. Odnośniki z powyższymi zapytaniami są pobierane i analizowane pod kątem składniowym. Następnie tworzony jest zestaw odnośników do stron podanych w wynikach wyszukiwania. Tutaj sprawdzana jest unikalność odnośników. Jeśli któryś wystąpił w liście wynikowej, nie zostaje do niej ponownie dodany. Końcowym etapem jest analiza składniowa listy wyników, tak, aby było możliwe wyciągnięcie odpowiednich informacji.

6.4 JsoupParser

Jest to klasa będąca implementacją interfejsu WebPageParser, która umożliwia wczytywanie i analizę składniową wszystkich stron internetowych. Wykorzystuje w tym celu darmową bibliotekę Jsoup napisaną w języku Java. Do analizy wykorzystuje zarówno podejście w stylu DOM, jak również wyszukiwanie elementów stylizowane na kaskadowe arkusze stylów.

Zawiera ona następujące metody:

- *SearchBar parseSearchBar(String url) throws IOException* - jest metodą umożliwiającą analizę składniową strony pod kątem znalezienia na niej wyszukiwarki internetowej i stworzenie URL pozwalającego na wyszukanie danych za pomocą wyszukiwarki.

- *List<SearchBar>parseSearchBar(List<String>urls)* - pozwala na analizę składniową wielu stron z wykorzystaniem metody *SearchBar parseSearchBar(String url) throws IOException.*

- *List<TextualContent>parseResult(List<String>urls)* - metoda ta umożliwia wczytanie i analizę składniową zbioru stron internetowych reprezentowanych poprzez hiperłącza. Jako argument przyjmuje listę łańcuchów znaków reprezentujących wspomniane wyżej odnośniki. Wywołuje drugą wersję metody, która zajmuje się wczytywaniem oraz analizą składniową pojedynczych stron internetowych. Zwraca listę obiektów typu WebPage zawierających przeanalizowane i ułożone w spójną strukturę strony internetowe wraz z API potrzebnym do wyszukiwania tekstu i odnośników do innych stron,

- *TextualContent parseResult(String url) throws IOException* - jest to metoda umożliwiająca wczytanie strony internetowej oraz jej dogłębną analizę składniową prowadzącą do otrzymania w pełni ustrukturyzowanego obiektu, do którego dostęp został obudowany klasą TextualContent w celu ułatwienia, ale też ograniczenia dostępu. Metoda ta przyjmuje za argument łańcuch znaków reprezentujący hiperłącze,

które użytkownik zamierza przeanalizować pod kątem składniowym. W jej pracy jest wykorzystywane API biblioteki Jsoup, do której odnośnik znajduje się tutaj. Ze względu na możliwe utrudnienia, takie jak m. in.:

- brak dostępu do sieci Internet,
- przerwanie dostępu do sieci w trakcie pobierania dokumentu z sieci Internet,
- błąd podczas wczytywania dokumentu do procesu analizy składniowej,

metoda ta może wyrzuć wyjątek typu IOException, a więc ogólny wyjątek dotyczący operacji wejścia/wyjścia - będący będący typem nadrzędznym względem wszystkich operacji wejścia/wyjścia w tym tych związanych z pobieraniem danych z Internetu, wczytywaniem, zapisywaniem i tym podobne. Jej wydajność jest zależna od wydajności samej biblioteki Jsoup. Na komputerze z czterowątkowym procesorem firmy Intel (seria i5, dla laptopów) operacja wczytania dużej strony (poprzez dużą stronę rozumiem w tym wypadku np. portal www.onet.pl, albo wp.pl i tak dalej) może zająć nawet do dwóch sekund (łącze Internetowe ma prędkość praktyczną na poziomie ok. 70 Mb/s przy pobieraniu danych).

6.5 SearchBar

Jest to klasa opakowująca wyniki analizy składniowej danej strony internetowej pod znalezienia na danym serwisie wyszukiwarki. Pozwala na skonstruowanie (na podstawie danych zawartych w kodzie strony) zapytania (metoda GET) zwracającego wyniki działania wyszukiwarki.

W skład tej klasy wchodzą następujące metody:

- *String getSearchURL()* - zwraca zapytanie do wyszukiwarki internetowej w postaci łańcucha znaków. Aby go skonstruować, należy znaleźć formularz wyszukiwania na danej stronie stronie internetowej (charakterystyczne cechy:
  - formularz może zawierać atrybut method="get"
  - sam znacznik formularza, lub znaczniki zewnętrzne zawierają słowa związane z wyszukiwaniem pośród atrybutów (albo angielskie „search", albo w języku ojczystym - w przypadku polskiego jest to temat słowotwórczy „szuka"))

Zazwyczaj w atrybucie action znajduje się relatywny/absolutny odnośnik do wyszukiwarki, dodaje się do niego symbol pytajnika, nazwę na zapytanie (atrybut name w znaczniku typu input, często „szukaj", „q", „qt"), znak równości. Tak skonstruowany łańcuch znaków zostaje zwrócony przez tę metodę. Wystarczy do niego dodać słowa kluczowe oddzielone symbolem dodawania, co jest wykonywane w procesie wyszukiwania danych przez aplikację.

6.6 SearchEnginePresenter

Jest to klasa będąca prezenterem, a więc warstwą pośredniczącą pomiędzy widokiem (GUI), a logiką (wyszukiwarka). Zostaje wywoływany przez GUI w momencie kliknięcia przycisku „search", wywołuje wyszukiwarkę, a gdy ta zwraca wyniki przekazuje wyniki klasie obsługującej GUI, w celu ich wyświetlenia.

W skład tej klasy wchodzą następujące metody:

– *void findByKeywords(SearchForm source, List<String >keywords, List<String >urls, int searchingLevel) throws IOException* - uruchamia wyszukiwarkę, filtruje wyniki pod kątem zawartości (usuwa odnośniki, które nie zawierają tekstu lecz wyłącznie obrazki, filmiki), a następnie przekazuje wyniki do GUI reprezentowanemu przez obiekt typu SearchForm.

7 Użyte technologie

7.1 JSOUP

JSOUP jest biblioteką napisaną w języku Java umożliwiającą pracę z dokumentami HTML. JSOUP dostarcza spójne API dla manipulowania, jak również pozyskiwania danych ze stron internetowych. Implementuje standard WHATWG HTML 5, umożliwia parsowanie plików HTML do modelu DOM. JSOUP umożliwia parsowanie stron o zróżnicowanej strukturze. Podstawowe funkcjonalności:

- webscrapping oraz parsowanie dokumentów HTML na podstaiwe URL, lokalnego pliku lub łańcucha znaków zawierającego kod HTML,
- dostęp do danych za pomocą składni przypominającej przechodzenie po drzewie DOM/ selektory CSS,

- umożliwia manipulowanie oraz modyfikację poszczególnych elementów, atrybutów oraz samego tekstu,
- czyszczenie HTML z niebezpiecznych elementów z wykorzystaniem białej listy.

8 Teoria

8.1 Web scraping

8.1.1 Wstęp

Web scraping jest techniką wyciągania informacji ze stron internetowych za pomocą wyspecjalizowanego oprogramowania. Jest to osiągane albo za pomocą bezpośredniej implementacji protokołu HTTP (na którym sieć Internet bazuje), albo poprzez wbudowanie przeglądarki internetowej.

Web scraping jest ściśle związany z indeksowaniem stron internetowych, które to indeksuje informacje w Internecie za pomocą bota lub Web Crawlera i jest uniwersalną techniką przyjęta przez większość wyszukiwarek internetowych. W przeciwieństwie do tego, web scraping skupia się bardziej na przekształceniu nieuporządkowanych danych w sieci, zwykle w formacie HTML w strukturę danych, która może być przechowywana i analizowana w centralnej, lokalnej bazie danych lub arkuszu kalkulacyjnym. Web scraping jest również związany pojęciem web automation, które to jest procesem symulującym surfowanie w sieci przez człowieka za pomocą wyspecjalizowanego programu komputerowego. Zastosowania web scrapingu obejmują porównanie cen online, monitorowanie danych o pogodzie, wykrywanie zmian w określonej witrynie internetowej, wszelakie badania oraz integrację danych internetowych.

Web scrapping to proces automatycznego zbierania informacji z sieci WWW. Jest to aktywnie rozwijająca się dziedzina, dzieląca wspólny cel, którym jest wizja semantycznej sieci; ambitna inicjatywa, która wciąż wymaga przełomu w przetwarzaniu tekstu, lepszego rozumienia semantyk, sztucznej inteligencji i interakcji na linii człowiek-komputer. Obecne rozwiązania web scrapingu wahają się od rozwiązań *„ad-hoc”* (wymagających udziału człowieka), do w pełni zautomatyzowanych systemów, które są w stanie dokonać konwersji całych witryn internetowych w informacje o określonej strukturze, oczywiście z pewnymi ograniczeniami.

8.1.2 Techniki

Istnieje wiele technik analizy składniowej i wydobywania danych ze stron internetowych. Część z nich stawia na pełną automatyzację procesu (tutaj też istnieje rozróżnienie na aplikacje dokonujące analizy według pewnych gotowych schematów, a te które idą w niezwykle zaawansowane rozwiązania związane ze sztuczną inteligencją i uczeniem maszynowym), część stawia na kooperację z człowiekiem. Poniżej znajduje się lista najpopularniejszych technik analizy składniowej (nazwy angielskie):

- „Human copy-and-paste": Czasami nawet najlepsze technologie nie są wstanie zastąpić możliwości, jakie daje analiza danych przez człowieka. W niektórych sytuacjach może być to jedyne działające rozwiązanie, zwłasza wtedy, gdy strony internetowe zawierają zabezpieczenia uniemożliwiające działanie zautomatyzowanemu oprogramowaniu.

- „Text grepping and regular expression matching": proste, ale potężne podejście do wydobywania informacji ze stron internetowych; może być oparte o polecenia grep UNIX-a lub wyrażenia regularne z języków programowania takich jak np. Perl, Python.

- „HTTP programming": statyczne i dynamiczne strony internetowe mogą być pobierane poprzez umieszczenie żądania HTTP do serwera sieci Web przy użyciu gniazdek programowych.

- „HTML parsers": Wiele portali internetowych posiada duże zbiory stron generowanych dynamicznie ze źródeł, takich jak bazy danych. Dane z tej samej kategorii są zazwyczaj kodowane w podobny sposób za pomocą wspólnego skryptu lub szablonu. W eksploracji danych, program, który wykrywa takie szablony w danym źródle informacji, wydobywa ich treść i tłumaczy je do relacyjnej postaci, nazywany jest wrapperem. Algorytm obudowywania zakłada, że strony wejściowe, otrzymywane przez wrapper, mają wspólny schemat i mogą one być łatwo zidentyfikowane w kategorii wspólnego schematu URL. Ponadto, niektóre semistrukturalnych języki zapytań, takie jak XQuery oraz HTQL mogą być używane do analizowania stron HTML oraz ich pobierania wraz przekształcaniem zawartości danej strony.

- „DOM parsing": Poprzez wbudowanie w pełnoprawnej przeglądarki internetowej, takiej jak Internet Explorer, Google Chrome, Mozilla Firefox, aplikacje mogą otrzymywać dynamiczną zawartość stron generowaną przez skrypty po stronie klienta (w tym wypadku: przeglądarki).

Te aplikacje kontrolujące przeglądarki umożliwiają także analizę składniową stron internetowych wraz z konwersją do drzew DOM, bazując na tym, które programy mogą otrzymać części danej strony.

- „Web-scraping software": Na rynku jest dostępnych wiele narzędzi, które mogą zostać wykorzystane do dostosowania rozwiązań z dziedziny web-scrapingu. To oprogramowanie może być w stanie automatycznie rozpoznawać strukturę strony, tworzyć interfejsy, które usuwają potrzebę samodzielnego pisania kodu dokonującego web-scrapingu, skrypty, które mogą być zdolne do pozyskiwania i przetwarzania zawartości lub interfejsy bazodanowe, które mogą przechowywać przechwycone dane w lokalnych bazach danych.

- „Vertical aggregation platforms": Istnieje kilka firm, które zaprojektowały i wdrożyły platformy zbierające dane dla konretnych branży. Platformy te tworzą i monitorują dużą ilość botów dla konkretnej branży bez bezpośredniego udziału człowieka, oraz bez pracy związanej z konkretną stroną docelową. Przygotowania do działania platformy zawierają ugruntowanie bazy wiedzy o całej branży, co pozwala oprogramowaniu na automatyczne tworzenie kolejnych botów. Solidność całej platformy jest mierzona jakością otrzymywanych informacji (zwykle liczba pól) oraz skalowalnością (czyli jak szybko jest w stanie zwiększyć skalę do setek, a nawet tysięcy obsługiwanych przez boty stron). Ta ostatnia cecha jest głównie wykorzystywana do znajdowania elementów stron, które platformy te mogą uznać za zbyt skomplikowane, albo wymagające zbyt wiele pracy w celu wydobycia z nich danych.

- „Semantic annotation recognizing": Parsowane strony mogą zawierać metadane albo znaczniki semantyczne wraz z adnotacjami, które mogą być wykorzystane do lokalizowania określonych fragmentów danych. Jeśli adnotacje są wbudowane w stronę, tak jak w przypadko Microformat, to ta technika może być pojmowana jako specyficzny przypadek analizy składniowej dokumentu DOM. W innym wypadku, adnotacje zebrane w warstwę semantyki są przechowywane i zarządzanie niezależnie od stron internetowej, więc scrapery mogą otrzymywać schemat danych wraz z instrukcjami właśnie z tej warstwy przed właściwą analizą strony.

- „Computer vision web-page analyzers": Istnieją techniki wykorzystujące uczenie maszynowe, które pozwala na próby identyfikacji i wyciągania ifnormacji ze stron internetowych poprzez interpetorwanie stron wizualnie, w przybliżony sposób do tego, w jaki robią to ludzie.