Scrapper ostrzeżeń pogodowych IMGW

Jakub Dakowski

11.01.2021

Opis teoretyczny

Motywacja

Od ponad stu lat ludzkość może obserwować skutki swojej industrialnej działalności na klimacie. Wśród skutków pojawiają się między innymi zmiany w temperaturze globalnej. Wspomniany wzrost temperatury stanowi sam w sobie zagrożenie zdrowotne dla ludzi. Oprócz tego, powoduje on wyższe szanse na takie anomalie jak fale ciepła, czy susze. Wraz z nasileniem się tych zjawisk, skutki zdrowotne będą także się nasilać, wymuszając na wielu osobach zmiany w trybie życia. W związku z tym pojawia się potrzeba uwzględniania w działaniu różnych aplikacji informacji o ostrzeżeniach klimatycznych. Niestety jednak IMGW - organ odpowiedzialny za tworzenie tych ostrzeżeń dla Polski - publikuje je w jednej z najtrudniejszych do automatyzacji form - plikach PDF.

Czym jest ten program?

Projekt ten jest przedsięwzięciem z zakresu inżynierii danych mając za zadanie sprowadzić dane PDF do jaśniejszego formatu JSON. Dokonywane jest to przy pomocy biblioteki *plumber*, która umożliwia utworzenie z kodu R działającego API.

API to posiada dwa endpointy. Jeden udostępnia wszystkie ostrzeżenia. Drugi natomiast pozwala na wyszukiwananie ostrzeżeń na podstawie nazw powiatu.

Dane zbierane przez scrapper

Największym problemem tej pracy jest mała ilość dostępnej dokumentacji na temat sposobu działania ostrzeżeń. Większość informacji na ich temat autor musiał wywnioskować na własną rękę.

Opublikowane ostrzeżenia publikowane są w zbiorowych komunikatach dla określonych województw. Trudno powiedzieć, czy jedno województwo może mieć kilka plików. W jednym może znajdować się jednak kilka ostrzeżeń.

Komunikaty mogą być ogłoszeniem, aktualizacją, lub odwołaniem ostrzeżenia. Autor jednak także w tym obszarze nie ma informacji na temat sposobu działania systemu. W każdym ogłoszeniu, bądź aktualizacji znajduje się informacja o typie i stopniu zjawiska, czasie ważności, prawdopodobieństwie, przebiegu, komunikatach SMS, RSO oraz uwagach. Odwołania podają natomiast nie podają czasu ważności, prawdopodobieństwa i przebiegu, a zamiast tego dają powód odwołania oraz moment odwołania. Każdy plik dodatkowo zawiera informację o dyżurnym synoptyki, kilka informacji o numerze ostrzeżenia (dla województwa i powiatów, ale niestety nie dla kraju). Można także tam znaleźć krótką informację prawną o możliwości udostępniania. Cytując:

> Wszelkie dalsze udostępnianie, rozpowszechnianie (przedruk, kopiowanie, wiadomość sms) jest dozwolone wyłącznie w formie dosłownej z bezwzględnym wskazaniem źródła informacji tj. IMGW-PIB.

Autor dokładał wszelkich starań, aby przekazywać ostrzeżenia w formie dosłownej, starając się jedynie przetransformować je do czytelnej dla komputera wersji.

Rozwiązanie problemu pobierania ostrzeżeń

Czyli inaczej opis działania algorytmu.

Pobieranie danych

Schemat działania systemu jest stosunkowo prosty. System rozpoczyna od pobrania listy aktualnych ostrzeżeń ze strony https://danepubliczne.imgw.pl/data/current/ost_meteo/. Dane są następnie transformowane do formy tabeli oraz czyszczone dla łatwiejszej obsługi.

```
webpage_url <- "https://danepubliczne.imgw.pl/data/current/ost_meteo/"
webpage <- xml2::read_html(webpage_url)

ost_files <- rvest::html_table(webpage)[[1]] %>%
    tibble::as_tibble(.name_repair = "unique") %>%
    filter(Name != "" & Name != "Parent Directory") %>%
    select(Name, `Last modified`) %>%
    rename_with(function(x) {
        "modified_at"
    }, "Last modified")
```

```
## New names:
## * `` -> ...1
```

```
ost_files
```

Program następnie pobiera kolejne ogłoszone ostrzeżenia i kompiluje je do ramki danych. Dokonuje tego funkcja `get_warns`, która odwołuje się do `read_warns`.

```
get_warns <- function(file) {
    webpage_url <- "https://danepubliczne.imgw.pl/data/current/ost_meteo/"
    saved.file <- paste("tmp", file, sep = "/")
    if (!file.exists(saved.file)) {
        download.file(paste(webpage_url, file, sep = ""), saved.file,
            mode = "wb")
    }
    return(read_warns(saved.file))
}

read_warns <- function(saved.file) {
    txt <- preprocess(pdf_text(saved.file))
    extracted <- extract(txt)

    extracted$file <- saved.file

    extracted$messtype[extracted["messtype"] == " ZMIANA"] <- "1"
    extracted$messtype[extracted["messtype"] == " WYCOFANIE"] <- "2"
    extracted$messtype[extracted["messtype"] == ""] <- "0"
    return(extracted)
}
```

`read_warns` odczytuje plik PDF o określonym adresie i przetwarza go do formy tekstowej z pomocą biblioteki *pdftools*. Te dane ulegają przetworzeniu przez funkcje `preprocess` oraz `extract`.

Ekstrakcja ostrzeżeń

Ponieważ przetwarzane pliki są oryginalnie przechowywane w formacie PDF, znaczna część zawartych w nich znaków nie należy do faktycznych komunikatów, a jest jedynie nagłówkami, czy stopkami dokumentu.

Zadaniem funkcji `preprocess` jest usunięcie tych segmentów. Przeprowadza ona kolejno:

1. Usunięcie nowych linii, tabulacji oraz nadmiaorwych spacji.
2. Usunięcie numerów stron.
3. Usunięcie nagłówka z danymi adresowymi IMGW.
4. Usunięcie "ostrzeżenia dla powiatu)" znajdującego się w niepoprawnie zescrapowanym miejscu.
5. Dodanie nowej linii na początku każdego nowego komunikatu.
6. Konkatenacja oraz ponowne usunięcie nadmiarowych spacji.

```
# Przy uruchamianiu programu funkcja jest uruchamiana wraz z setupem,
# nie trzeba jej uruchamiać drugi raz.
preprocess <- function(txt) {
   txt <- gsub("\n+| {2,}\t+", " ", txt)
   txt <- gsub("strona \\d+ z \\d+", "", txt, ignore.case=T)
   txt <- gsub(" Instytut Meteorologii i Gospodarki Wodnej .+  www: www\\.imgw\\.pl",
               "", txt, ignore.case=T)
   txt <- gsub("Zjawisko/Stopień", "\nZjawisko/Stopień",
               gsub("ostrzeżenia dla powiatu\\) ", "", txt))
   txt <- paste(txt, collapse=" ")
   txt <- gsub(" {2,}", " ", txt)

   return (txt)
   }
```

Funkcja `extract` ma o wiele trudniejsze zadanie. Przyjmuje ona przetworzony tekst i rozpoczyna od przeszukania go z pomocą wyrażenia regularnego `START_PATTERN`, które umożliwia wykrycie województwa, numeru ostrzeżenia zbiorczego, treści ostrzeżeń jednostkowych i autora ostrzeżenia. Treść ostrzeżeń jest następnie przeszukiwana z pomocą wyrażenia `PATTERN`, które identyfikuje pojedyncze ostrzeżenia i wykrywa w nich konkretne dane. Kolejne wiersze odpowiadają za identyfikację:

- Zjawiska i stopnia zagrożenia (a także bycia modyfikacją/usunięciem danego ostrzeżenia)
- Zagrożonych powiatów
- Czasu rozpoczęcia
- Daty rozpoczęcia
- Czasu zakończenia
- Daty zakończenia
- Prawdopodobieństwa
- Opisu przebiegu
- Jeżeli ogłoszenie jest odwołaniem ostrzeżenia:
  - Czasu odwołania
  - Przyczyny odwołania
- Komunikatu SMS
- Komunikatu RSO
- Uwag do ostrzeżenia

Wyniki tego przeszukiwania są następnie transformowane do ramki danych, która uzupełniana jest o pozostałe informacje pochodzące z przeszukiwania z pomocą `START_PATTERN` oraz numer ostrzeżenia w danym pliku (na podstawie tego można potem utworzyć klucz składający się z nazwy pliku i numeru ostrzeżenia w pliku).

```
# Przy uruchamianiu programu funkcja jest uruchamiana wraz z setupem,
# nie trzeba jej uruchamiać drugi raz.
extract <- function(txt) {
  START_PATTERN <- paste(
    "Zasięg ostrzeżeń w województwie WOJEWÓDZTWO (?<voivodeship>[\\w\\-]+)",
    "OSTRZEŻENIA METEOROLOGICZNE ZBIORCZO NR (?<id>\\d+) WYKAZ OBOWIĄZUJĄCYCH OSTRZEŻEŃ",
    "o godz\\. \\d\\d:\\d\\d dnia \\d\\d\\.\\d\\d\\.\\d{4}",
    "(?<text>(?:\\n|.)+)",
```

```
    "Dyżurny synoptyk (?<creator>.+?(?= IMGW-PIB))",
    sep=" "
  )
  PATTERN <- paste(
    "Zjawisko/Stopień zagrożenia (?<event>[\\w ]+)/(?<lvl>\\d+)(?<messtype>| \\w+)",
    "Obszar \\(w nawiasie numer powiaty: (?<regions>(?:[\\w- ]+\\(\\d+\\)(?:, )*)+)",
    "(?:Ważność od godz\\. (?<starthour>\\d\\d\\:\\d\\d)",
    "dnia (?<startday>\\d\\d\\.\\d\\d\\.\\d{4})",
    "do godz\\. (?<endhour>\\d\\d\\:\\d\\d)",
    "dnia (?<endday>\\d\\d\\.\\d\\d\\.\\d{4})",
    "Prawdopodobieństwo (?<prob>\\d{1,3}\\%)",
    "Przebieg (?<how>.+?(?= SMS))|Czas",
    "odwołania godz\\. (?<hour>\\d\\d:\\d\\d) dnia (?<day>\\d\\d\\.\\d\\d\\.\\d{4})",
    "Przyczyna (?<cancelcause>.+?(?= SMS)))",
    "SMS (?<sms>.+?(?= RSO))",
    "RSO (?<rso>.+?(?= Uwagi))",
    "Uwagi (?<remarks>[^\\n]+)",
    sep=" "
  )
  pat <- str_match(txt, START_PATTERN)
  df <- as.data.frame(str_match_all(pat[4], PATTERN))
  df$voivodeship <- str_to_lower(pat[2], locale = 'pl')
  df$warn_id <- pat[3]
  df$author <- pat[5]
  df$infile <- rownames(df)
  return (df)
}
```

Jak nietrudno sobie wyobrazić strategia od tego momentu jest dość prosta. Oprogramowanie łączy ze sobą kolejne ramki danych:

```
file <- ost_files$Name[1]
df <- get_warns(file)

if (length(ost_files$Name) > 1) {
    for (file in ost_files$Name[2:length(ost_files$Name)]) {
        df <- rbind(df, get_warns(file))
    }
}

df
```

Obróbka danych

W tym momencie może zostać włączone filtrowanie tylko nowych ostrzeżeń. Program zmienia też zawartość zestawu kolumn:

- `regions` od teraz przechowuje wektor powiatów zamiast ciągu znakowego,
- `file` jest pozbawiane nazwy folderu,
- `prob` konwertowane jest na ułamek zamiart ciągu znakowego
- Daty kompilowane są do wystandaryzowanego formatu,
- Liczby całkowite konwertowane są do formatu liczb całkowitych.

Warto w tym momencie pochylić się nad `format_time` oraz `get_places`. Pierwsza skleja ze sobą dwie kolumny i przetwarza z użyciem `strptime`. Następnie pola `NA` uzupełniane są przez `NULL`. Druga natomiast jest prostym regexem, który dodatkowo konwertuje nazwy powiatów na małe litery.

```
format_time <- function(date, time) {
    t <- strptime(paste(date, time), "%d.%m.%Y %H:%M")
    t[is.na(t)] <- NULL
    return(t)
}

get_places <- function(place_list) {
    p <- " ?([^,]+)\\(\\d+\\),?"
    match <- str_match_all(place_list, p)
    return(lapply(match, function(x) {
        return(str_to_lower(x[, 2], locale = "pl"))
    }))
}
```

Poza tym program usuwa zbędne kolumny. W tym też momencie nadchodzi pora na jakże długą linię służącą do odsortowania ostrzeżeń niedotyczących danego miejsca. Taki wynik jest zwracany użytkownikowi w formacie JSON.

```
fin <- df %>%
  # filter(messtype == '0') %>%
  mutate(
    regions = get_places(df$regions),
    file = gsub("tmp/", "", file),

    prob = as.numeric(sub("%", "", prob)) / 100,

    starttime = format_time(df$startday, df$starthour),
    endtime = format_time(df$endday, df$endhour),
    canceltime = format_time(df$day, df$hour),

    lvl = as.numeric(lvl),
    messtype = as.numeric(messtype),
    infile = as.numeric(infile),
    warn_id = as.numeric(warn_id),
    ) %>%
  # mutate(include = unlist(lapply(regions, function(x) { place %in% x })))
  # %>% filter(include) %>% select(-c(include)) %>%
  # ^ umożliwia sortowanie po powiatach
  select(-c(V1, startday, starthour, endday, endhour, day, hour))

knitr::kable(fin %>% select(event, starttime, endtime, rso),
             caption = "Przykładowy wynik działania algorytmu")
```

Table 1: Przykładowy wynik działania algorytmu

event	starttime	endtime	rso
Silny mróz	2022-01-11 21:00:00	2022-01-12 08:00:00	Woj. małopolskie (1 powiat), IMGW-PIB wydał ostrzeżenie pierwszego stopnia o silnych mrozach
Silny mróz	2022-01-11 23:00:00	2022-01-12 08:00:00	Woj. podlaskie (10 powiatów), IMGW-PIB wydał ostrzeżenie pierwszego stopnia o silnych mrozach
Silny mróz	2022-01-11 21:00:00	2022-01-12 08:00:00	Woj. podkarpackie (2 powiaty), IMGW-PIB wydał ostrzeżenie pierwszego stopnia o silnych mrozach

Oczywiście, jeśli żadne ostrzeżenia nie zostaną znalezione, API zwróci pusty wektor, który w JSONie jest reprezentowany jako `[]`. Ten warunek nie jest tu jednak wprowadzony.

Sprawdzenie założeń

Autor niestety wciąż nie jest pewien, czy opracowane przez niego wyrażenie regularne działa w pełni. Można je co najwyżej sprawdzić na zestawie wszystkich poprzednich ostrzeżeń, co jednak jest trudne, ze względu na ich inny sposób przechowywania na stronie (są one uprzednio kompresowane).

Interpretacja wyników

We wszystkich komunikatach, które autor napotkał algorytm zadziałał poprawnie, wskazuje to na jego wysoką skuteczność. Niestety aktualne rozwiązanie działa stosunkowo wolno i jest ograniczone, jak chodzi o zmiany formatu ogłoszeń. Założenia na temat ich formy także są dość silne, co jednak wynika z bierności IMGW. Cała potrzeba produkcji takiego oprogramowania w XXI wieku wydaje się absurdalna. Zdaniem autora, dostęp do tego typu ostrzeżeń, jako informacjach umożliwiających ludziom bezpieczną egzystencję, w systemach automatycznych powinien być jak najprostszy. Program pozostawia więc wiele do życzenia, ale wciąż uzupełnia lukę ignorowaną przez organ ostrzeżenia wydający.

Informacja o użytych pakietach

Nazwa paczki	Opis	Miesięczne pobrania	Autorzy	Data opublikowania
`xml2`	Pobieranie dokumentów HTML	620 159	Hadley Wickham	30.11.2021
`tidyverse`	Zestaw bibliotek wspomagających pracę w R	689 554	Hadley Wickham	15.04.2021
`pdftools`	Scrapowanie tekstu z plików PDF	brak danych	Jeroen Ooms	06.05.2021
`stringr`	Przetwarzanie ciągów znakowych, regex	806 997	Hadley Wickham	10.02.2019
`plumber`	Generowanie API ze skryptów R	11 776	Barret Schloerke	24.03.2021

Polecana literatura oraz linki do wykorzystanych/przydatnych stron

- Zdecydowanie nie dokumentacja API IMGW
- https://www.rplumber.io/
- https://dplyr.tidyverse.org
- Stack Overflow
- https://regexr.com/
- Regular Expression Language - Quick Reference
- Markdown table generator