Politechnika Wrocławska

Wizualizacja wielkich zbiorów danych

Mapa literacka dla korpusów dokumentacja

Przemysław Wujek 234983
Paweł Czarnecki 234974
Dawid Piechota 235851
Wojciech Wójcik 235621

Prowadzący:
dr inż. Tomasz Walkowiak

2 luty 2021

Spis treści

1 Zadanie programu

Zadaniem stworzonej aplikacji jest analiza tekstów literackich, która polega na wykrywaniu lokalizacji, wyznaczeniu ich częstości występowania oraz kolejności poszczególnych wystąpień. Aplikacja wykorzystuje narzędzie *NER* oraz *Polem* udostępniane przez *clarin-pl*.
Aplikacja klienta umożliwia wgranie tekstu, którego wyniki analizy zostają zaprezentowane w postaci mapy z zaznaczonymi punktami. Zaimplementowane funkcje umożliwiają wyświetlanie informacji o wszystkich znalezionych miejscach w tekście, które są prezentowane lokalizacją oznaczoną na mapie oraz oryginalnym słowami określającymi lokalizacje, znajdującymi się w tekście. Dodatkowo może być ustalana wielkość okna, którego początek i koniec ustalany jest na podstawie liczby zdań w tekście. Kliknięcie na punkt grupujący lokalizacje wyświetla listę miejsc w nim zawartych. Prezentacja punktów jest także możliwa w sposób chronologiczny w czasie. Użytkownik ponadto posiada możliwość zapisu do pliku. Dodatkowo aplikacja przechowuje w bazie danych wcześniej przeanalizowane teksty, aby użytkownik nie musiał zlecać zadania analizy tego samego tekstu dwa razy.

2 Link do repozytorium projektu

`https://github.com/Isild/Mapa-literacka-dla-korpus-w`

3 Technologia

Część serwerowa aplikacji zaimplementowana została z wykorzystaniem języka Python wraz z frameworkiem Flask.
Część kliencka wykorzystuje bibliotekę Vue z dodatkowym wykorzystaniem biblioteki *Leaflet*.

4 Uruchomienie

4.1 Uruchamianie ręczne

Po pobraniu projektu z repozytorium z githuba należy uruchomić aplikację serwera oraz aplikację klienta.
Aplikacja klienta znajduje się w katalogu */client*. W owym katalogu także znajduje się *README.md* w którym umieszczono instrukcję uruchomienia aplikacji.

Listing 1: Komendy budujące aplikacje klienta

```
1 # Komenda instalujaca wszystkie potrzebne biblioteki
2
3 yarn install
4
5 # Komenda kompilujaca aplikacje w trybie developmentu
6
7 yarn serve
8
9 # Komenda budujaca aplikacje w trybie produkcyjnym
10
11 yarn build
```

Aplikacja serwera znajduje się w katalogu */server*. Tutaj także umieszczony jest plik *ReadMe.md* zawierający instrukcję instalacji oraz uruchomienia aplikacji.

Listing 2: Komendy budujące aplikacje serwera

```
0.  Upewnij sie ze posiadasz zainstalowanego python3, pip i virtualenv
1.  Tworzenie venv
    **_Linux_**

    $ python3 -m venv env
    $ source env/bin/activate

    **_Windows_**

    $ py -m venv env
    $ .\env\Scripts\activate

2.  Instalacja wymaganych bibliotek z pliku `requirements.txt`

    (env) $ pip install -r requirements.txt

3.  Uruchamianie

    (env) $ python app_run.py run

    Podobny komunikat powinien zostac wyswietlony w terminalu jesli aplikacja
    poprawnie sie uruchomila i dziala:

    * Serving Flask app "app.services.app" (lazy loading)
    (...)
    * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)


# Komenda aplikacji
    run - uruchamia aplikacje serwera, pelne wywolanie 'python app_run.py run'
```

4.2 Docker

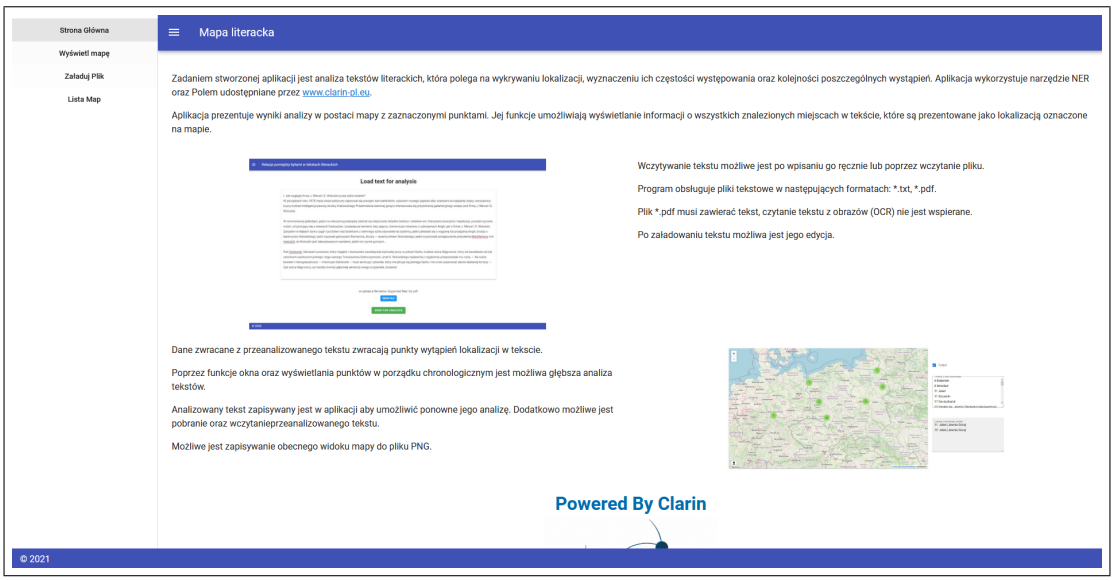
Aplikacja będzie możliwa do uruchomienia za pomocą narzędzia Docker wywołując komendę *docker-compose up*. Nie jest wspierana wersja Docker Toolbox.

5 Aplikacja

5.1 Menu

Do poruszania się po aplikacji wykorzystywane jest zwijane menu znajdujące się z prawej strony. Dzięki niemu można przemieszczać się pomiędzy stronami.

- Strona Główna - zawiera ogólną instrukcję
- Wyświetl mapę - wyświetla przeanalizowane teksty na mapie
- Załaduj Plik - pobiera tekst z pliku do analizy
- Lista Map - lista wszystkich przeanalizowanych tekstów



Rysunek 1: Strona domowa aplikacji

5.2 Wczytywanie tekstu do analizy

Interfejs użytkownika umożliwia wczytywanie tekstu podanego przez użytkownika ręcznie, oraz poprzez wczytanie pliku. Ręczne wczytanie tekstu przez użytkownika polega na wklejeniu tekstu do odpowiedniego miejsca na stronie przeznaczonej do wczytywania. Wczytywanie z pliku polega na wybraniu odpowiedniego pliku, gdzie jego zawartość jest wstawiana w pole na stronie. Umożliwia to jego dalszą edycję, bądź wysłanie.

5.2.1 Obsługiwane formaty

Program aktualnie wspiera wczytywanie tekstu z plików tekstowych (np. txt) oraz plików PDF. Ekstrakcja tekstu z pdf jest wykonywana przez bibliotekę `pdf.js`. Operacja ta jest wykonywalna lokalnie na komputerze użytkownika. Plik pdf musi zawierać tekst, czytanie tekstu z obrazów (OCR) nie jest wspierane.

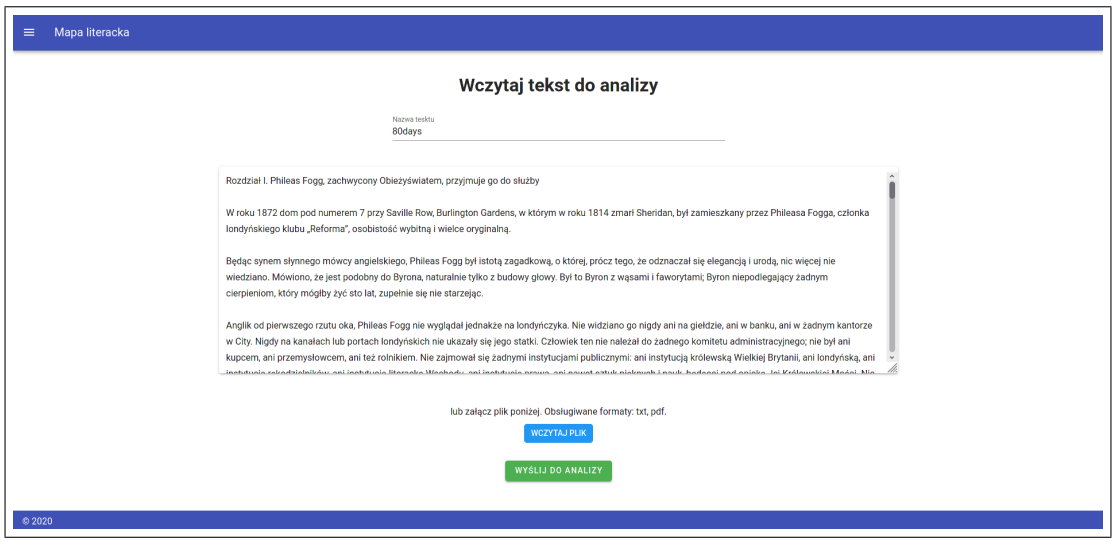
5.2.2 Ręczna modyfikacja

Po wczytaniu tekstu jest on uzupełniany do pola, w którym użytkownik może zmienić jego treść, usunąć część tekstu lub całkowicie zrezygnować z wysłania. Możliwe jest również ręczne uzupełnienie tego pola z pominięciem wczytywania pliku.

5.2.3 Dane

Cały tekst książki pakowany jest w JSONa i wysyłany metodą `POST` na endpoint wystawiony przez backend. Ten endpoint zwraca wtedy `ID` dokumentu w bazie danych. Ten `ID` używany jest przez podstronę wyświetlająca dane. Zaraz po wysłaniu backend rozpoczyna analizę tekstu oznaczając to stosownym statusem w bazie danych. Gdy analiza jest zakończona status ulega zmianie i możliwe jest pobranie danych grafu przez podstronę służącą do wizualizacji.

Pobranie tych danych polega na wysłaniu zapytania typu `GET` na ten sam endpoint. Gdy dane nie są gotowe, zwracany jest tylko odpowiedni status, a podstrona informuje stosownym komunikatem. Gdy analiza jest zakończona zwracane są dane mapy, które zostają wyświetlone. W momencie włączenia trybu turbo pobierane są wszystkie punkty

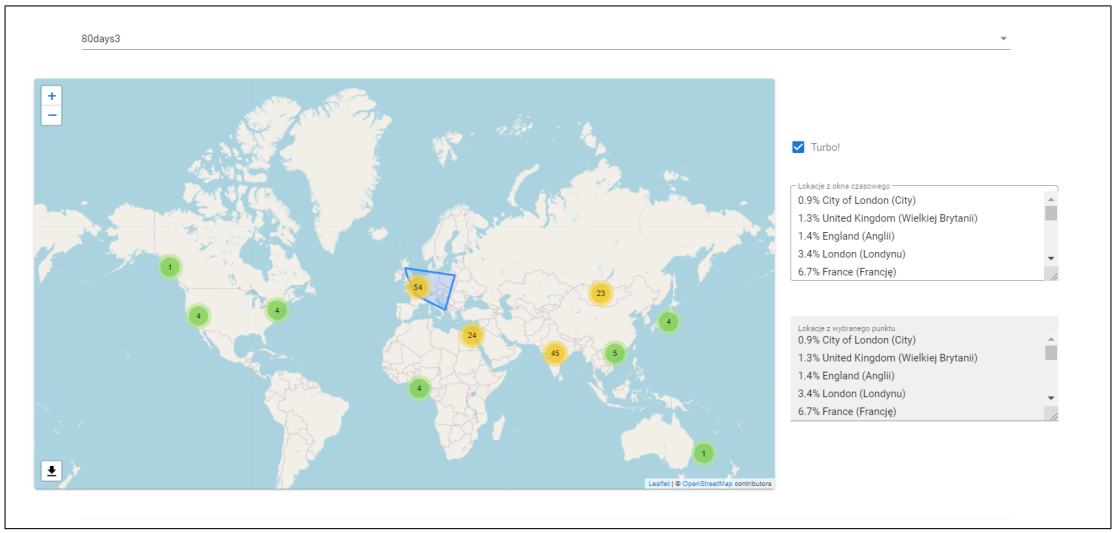


Rysunek 2: Interfejs wczytywania tekstu

z puli książki. W przeciwnym trybie jako parametry dodatkowo wysyłane są współrzędne wyświetlanego fragmentu mapy i pobierane dane są tylko z tego obszaru.

5.3 Prezentacja danych

Po pobraniu wygenerowanych lokacji z serwera w formacie JSON, apliakcja po stronie użytkownika prezentuje dane na mapie w postaci wielokolorowych klasterów znaczników [rys3].



Rysunek 3: Prezentacja lokacji na mapie

5.3.1 Biblioteka wizualizacji danych

Do wizualizacji danych wykorzystano bibliotekę *Leaflet* z pluginem *marker-cluster*. *Leaflet* jest elastyczną biblioteką typu open-source umożliwiającą wyświetlanie oraz modyfikowanie mapy świata. Obraz mapy jest pobierany z *OpenStreetMap*, darmowego API, które umożliwia pobieranie mapy w kawałkach zależnych od przybliżenia oraz rozmiarów okna. W projekcie wykorzystano grupowanie znaczników w klastry, tak aby mapa była czytelna.

Dokonując przybliżania danego obszaru biblioteka dynamicznie rozdziela grupy na mniejsze zbiory punktów w celu aktualizacji rozmieszczeń punktów.
Dzięki przystosowaniu biblioteki do urządzeń mobilnych duża ilość punktów na mapie nie obciąża urządzenia. Na testowanym laptopie posiadającym 4 GB RAMu aplikacja działała płynnie nawet przy dwóch tysiącach punktów.

5.4 Funkcjonalności reprezentacji graficznej

5.4.1 Wybór przeanalizowanego tekstu

Przy przejściu do widoku *Mapy* możliwe jest wybranie już przeanalizowanego tekstu znajdującego się w bazie danych. Ułatwia to przełączanie się pomiędzy analizowanymi tekstami w tym widoku. W przypadku dużej ilości przeanalizowanych tekstów łatwe znalezienie szukanego jest możliwe dzięki filtrowaniu przez wpisanie w pole wyboru nazwy przeanalizowanego tekstu lub jej części.

5.4.2 Zmiana sposobu pobierania danych

Użytkownik posiada opcje wyboru pomiędzy pobieraniem lokacji z serwera w całości oraz pobieraniem tylko tych lokacji, które aktualnie są w polu widzenia użytkownika. Druga opcja jest odpowiedzią na problemy z pamięcią przy dużej ilości lokacji (»5000). Między dwoma trybami można się przełączyć używając checboxa *Turbo!*, który domyślnie jest włączony i oznacza pobieranie całego zbioru lokacji. Autorzy aplikacji zalecają wyłączenie trybu turbo tylko w przypadku problemów z wydajnością.

5.4.3 Dostępne akcje z poziomu mapy

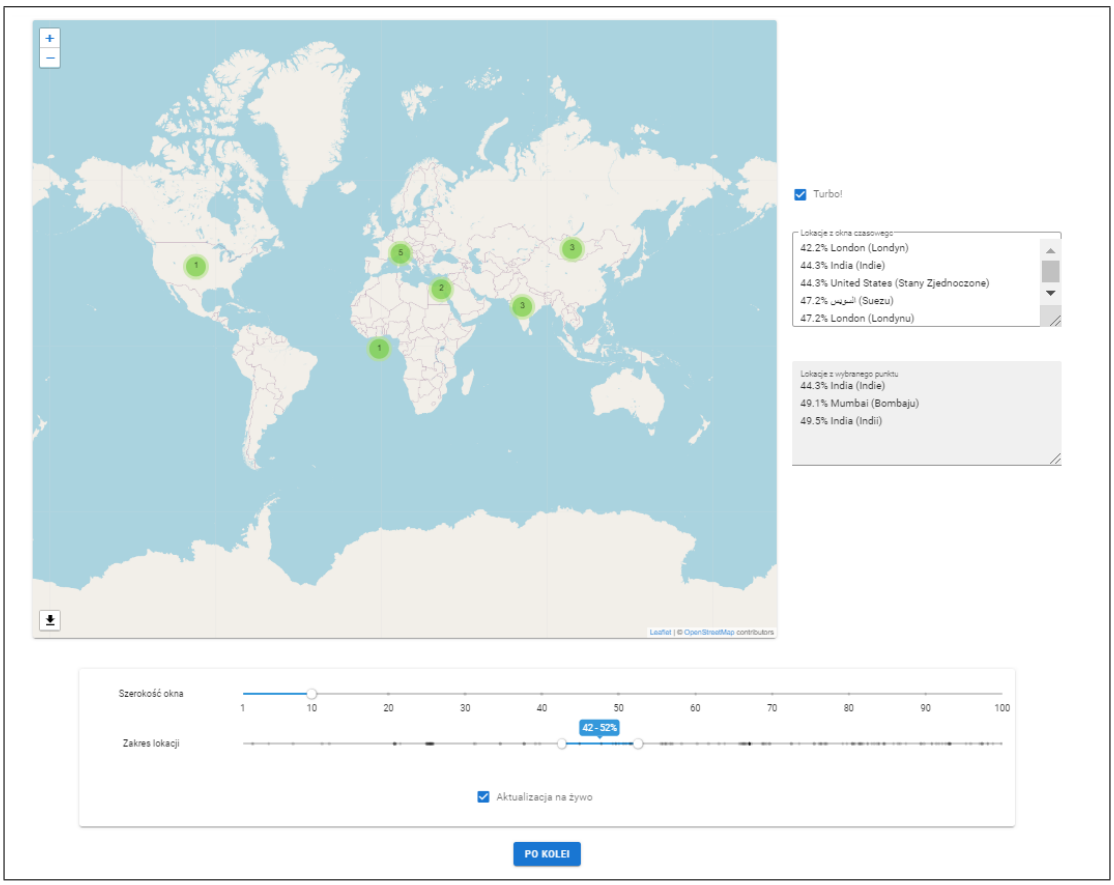
Mapę można przesuwać, przybliżać oraz oddalać. W przypadku wyłączenia trybu turbo, po każdym zmianie stanu mapy wysyłane jest zapytanie do serwera o lokacje w aktualnej pozycji.
Po zaznaczeniu jednego z znaczników, lub klastra znaczników, w oknie tekstowym po prawej stronie mapy pojawiają się informacje dotyczące zaznaczonego miejsca.

5.4.4 Filtrowanie po oknie czasowym

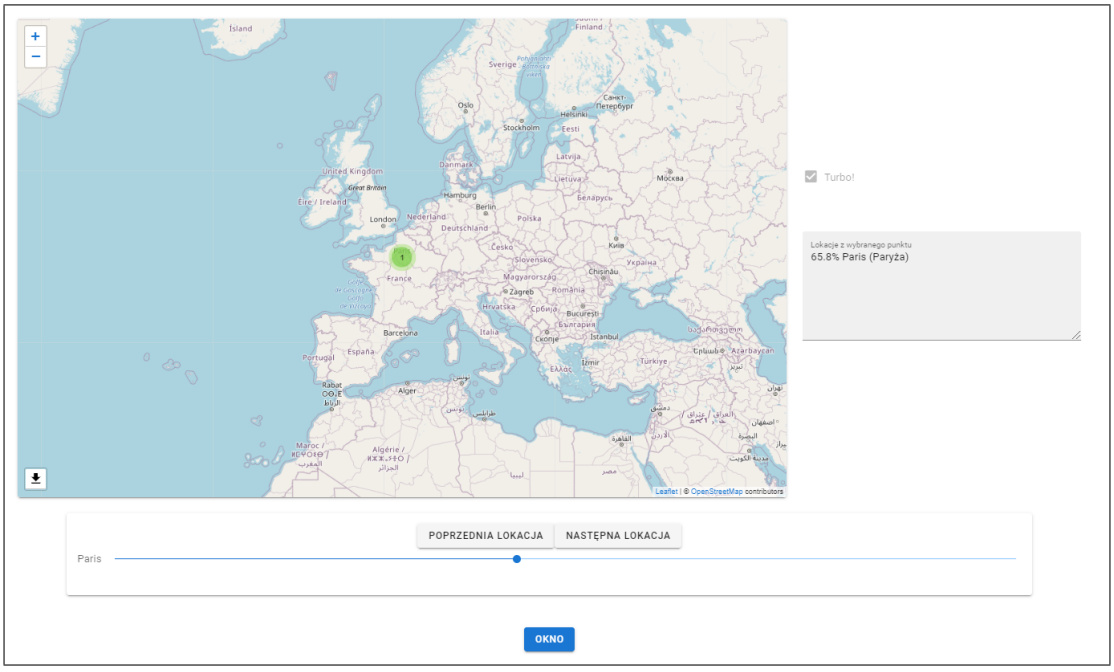
Interfejs umożliwia filtrowanie lokacji po oknie czasowym. Użytkownik wybiera szerokość okna oraz na jego podstawie - zakres okna. Filtrowanie jest oparte o miejsce wystąpienia lokacji w analizowanym tekście, gdzie przykładowo wartość okna 42-52% filtruje znaczniki na mapie tak, aby przedstawiły lokacje z części tekstu od 42% do 52%. Przy zaznaczeniu opcji *Aktualizacja na żywo*, lokacje są uaktualniane w czasie przesuwania okna. W przeciwnym przypadku, dzieje się to po puszczeniu klawisza myszy. Po każdej aktualizacji okna czasowego, po prawej stronie mapy wyświetlane są informacje o lokacjach z danego przedziału [rys4].

5.4.5 Podróż między punktami w chronologicznej kolejności

Drugim sposobem filtrowania danych jest podróż po jednej lokacji w chronologicznej kolejności. Użytkownik może użyć przycisków *Poprzednia lokacja* i *Następna lokacja*, lub przesuwać paskiem postępu [rys5].



Rysunek 4: Okno czasowe



Rysunek 5: Chronologiczna podróż

5.4.6 Eksport mapy do pliku JSON

Do zapisania aktualnego stanu modelu do pliku wykorzystane zostało API przeglądarki do wygenerowania pliku, który następnie jest pobierany.

Format: Dane zwracane przez funkcje parsująca zawierają się w formacie danych JSON. Format prezentuje się następująco:

```
{
  'id': 5,
  'nodesData': [
    {
        'id': 1, 'time': 1, 'name': 'Bialystok', 'orth': 'Bialegostoku',
        'ctag': 'subst:sg:gen:m3', 'ann': 'nam_loc_gpe_city', 'word_cnt': 1,
        'coords': {
                    'lat': 53.127505049999996,
                    'lng': 23.147050870161664
                  }
    },
    {
        'id': 2, 'time': 1, 'name': 'Wroc aw', 'orth': 'Wroc awiem',
        'ctag': 'subst:sg:inst:m3', 'ann': 'nam_loc_gpe_city', 'word_cnt': 1,
        'coords': {
                    'lat': 51.1263106,
                    'lng': 16.97819633051261
                  }
    },
    ...
  ],
  'name': 'Test_2',
  'settings': '',
  'status': 'ready'
}
```

Sekcja *nodes* odpowiada za przekazanie danych wszystkich lokalizacji wraz z ich czasem wystąpienia, nazwami(nazwą znalezioną na mapie oraz oryginalną nazwą), ctagiem, tagiem ann, licznikiem słów oraz współrzędnymi na mapie.

5.4.7 Eksport mapy do pliku graficznego

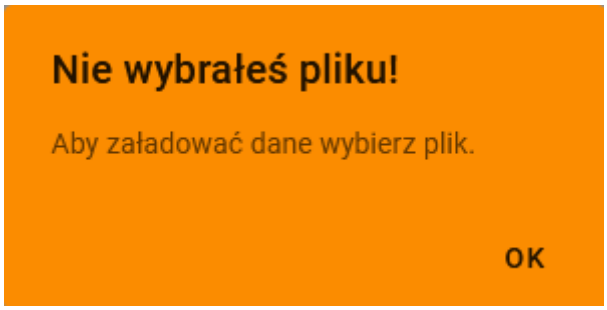
Eksport mapy odbywa się poprzez generowanie pliku PNG przez bibliotekę *leaflet-easyprint*. Jest on generowany na podstawie obecnej zawartości tagu komponentu mapy w którym znajduje się obraz mapy oraz znaczniki. Plik graficzny zawiera wszystko co jest widoczne w czasie kliknięcia przycisku z ikoną pobierania z wyłączeniem komponentów nawigacyjnych. Takie działanie umożliwia ustawienie mapy w odpowiadającej użytkownikowi stanie i jej eksport.

5.4.8 Odczyt z pliku

Struktura pliku jest identyczna do tej opisanej w rozdziale 5.4.6. Przy wczytywaniu z pliku użytkownik wybiera plik z dysku z danymi wykresu. W przypadku gdy ktoś niepo-

Rysunek 6: Element UI odpowiadający za wczytanie danych grafu

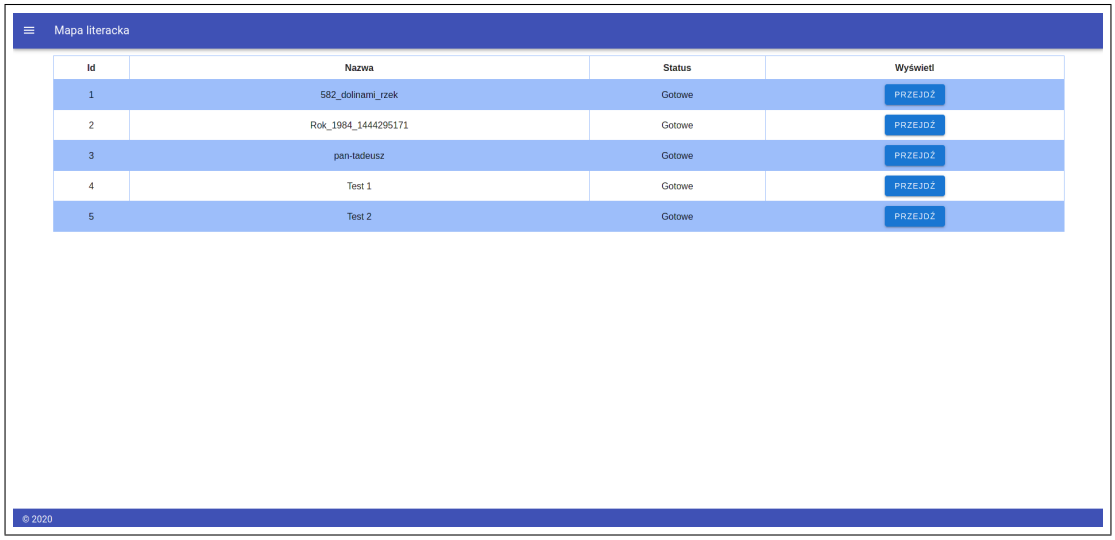
prawnie wypełni pole z wyborem pliku i kliknie przycisk Wczytaj dane z pliku zostanie mu wyświetlona informacja ukazana na rysunku 7



Rysunek 7: Komunikat informujący o braku pliku do wczytania

5.5 Lista map

Po wczytaniu tekstu zostaje on zapisany w bazie danych na serwerze. W zakładce znajduje się lista z wygenerowanymi już mapami. Można je przeglądać i otwierać. Po wybraniu i kliknięciu w interesującą nas mapę, zostajemy przeniesieni do jej wizualizacji. Tam możemy dokonać przeglądu mapy oraz jej zapisu do pliku.



Rysunek 8: lista wygenerowanych grafów

5.6 Metoda wyciągania danych o lokalizacjach z formatu XML

Ekstrakcja lokalizacji z przeanalizowanego tekstu odbywa się z wykorzystaniem narzędzia *NER*. Do analizy wykorzystywana jest wersja NER'a wykorzystująca model Linera2 obsługującego 82 kategorie. Wynikowy plik XML dzieli tekst na zdania i poszczególne tokeny ze słowami umieszcza w tagach przypisanych do konkretnego zdania. Wyszukiwanie lokacji odbywa się odrębnie dla każdego zdania w tekście. Z punktu widzenia zadania wyszukiwania miejsc, najistotniejsze są atrybuty:

- *nam_loc_gpe_city*
- *nam_loc_gpe_country*
- *nam_fac_goe*

które są umieszczane w tagu *ann* w tokenach należących do zdań, w których pojawia się jakaś lokalizacja. Niezerowa wartość w tagu *ann* mówi o tym, że dany tag odpowiada słowu, które jest częścią nazwy lokalizacji. Wartość ta jest również identyfikatorem poszczególnych lokalizacji w obrębie zdania. Dzięki tej własności zrealizowano wyszukiwanie pełnych nazw miejscowości, włącznie z nazwami składającymi się z dwóch lub więcej wyrazów.

Mając listę słów odpowiadających każdej napotkanej w tekście miejscowości wyprowadzana jest ich właściwa postać za pomocą narzędzia lematyzującego *Polem*. Podstawowa forma nazwy miejscowości otrzymana z Polema jest wykorzystywana do pobrania informacji o koordynatach za pomocą *geocodera dla Open Street Maps* o nazwie *Nominatim*. Komunikacja z serwisem odbywa się za pośrednictwem biblioteki *geopy*. Nominatim dla każdej wykrytej nazwy miejscowości zwraca długość i szerokość geograficzną, które wraz z pozostałymi informacjami, jak np. numer zdania w którym pojawiła się lokalizacja, nazwa lokalizacji, przekazywane są w formacie JSON do innych części programu. Raz wygenerowane z narzędzia *NER* pliki XML zostają poddane analizie przez algorytm, a wyniki są zapisywane w formacie JSON i przechowywane w serwerze.

5.7 Grupowanie danych po stronie backendowej

Chcąc ograniczyć ilość punktów przekazywanych do mapy stworzono trzy mechanizmy.
Pierwszy z nich polega na przesłaniu punktów widocznych tylko z widzianego obszaru mapy.
Drugie rozwiązanie polega na pokryciu widzialnej mapy szachownicą, w której każdym polu sprawdzana jest liczba punktów, a następnie zwracana ich ilość. Niestety takie rozwiązanie gubi istotną część informacji jaką są nazwy lokalizacji. Bez tej informacji użytkownik nie jest w stanie zobaczyć listy punktów w wybranej grupie. Ta opcja została wyłączona z aplikacji.
Trzecie rozwiązanie polega na przesłaniu danych z wybranego czasu dzieła literackiego. Jako parametr zostaje przekazany początek oraz koniec zakresu książki. Czas podany jest procentowo, to znaczy jaka część książki jest brana do okna czasowego na podstawie ilości zdań w utworze.