Politechnika Wrocławska

Zastosowanie informatyki w gospodarce

Relacje pomiędzy bytami w tekstach literackich - dokumentacja

Lider grupy:
Przemysław Wujek 234983
Skład grupy:
Paweł Czarnecki 234974
Łukasz Łupicki 257536
Dawid Piechota 235851
Bartosz Rodziewicz 226105
Wojciech Wójcik 235621

Prowadzący:
dr inż. Tomasz Walkowiak

15 kwietnia 2020

Spis treści

1 Zadanie programu

Zadaniem stworzonej aplikacji jest analiza tekstów literackich, która polega na wykryciu bytów, wyznaczeniu ich częstości występowania oraz powiązań pomiędzy nimi. Aplikacja wykorzystuje narzędzie *NER* udostępniane przez *clarin-pl*.
Aplikacja klienta umożliwia wgranie tekstu, którego wyniki analizy zostają zaprezentowane w postaci grafu. Zaimplementowane funkcje umożliwiają przeglądanie oraz edycję prezentowanego grafu.
Użytkownik posiada możliwość zapisu do pliku edytowanego grafu. Dodatkowo aplikacja przechowuje w bazie danych wcześniej przeanalizowane grafy, aby użytkownik nie musiał zlecać zadania analizy tego samego tekstu dwa razy.

2 Link do repozytorium projektu

`https://github.com/Isild/ZIWG`

3 Technologia

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

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 instukcję 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 serweru 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, pe ne 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.

Na stronie domowej znajduje się instrukcja korzystania z programu. Przy użyciu przycisków nawigacji można przechodzić do kolejnych podstron z których można wczytywać tekst lub wyświetlać i edytować grafy.

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.

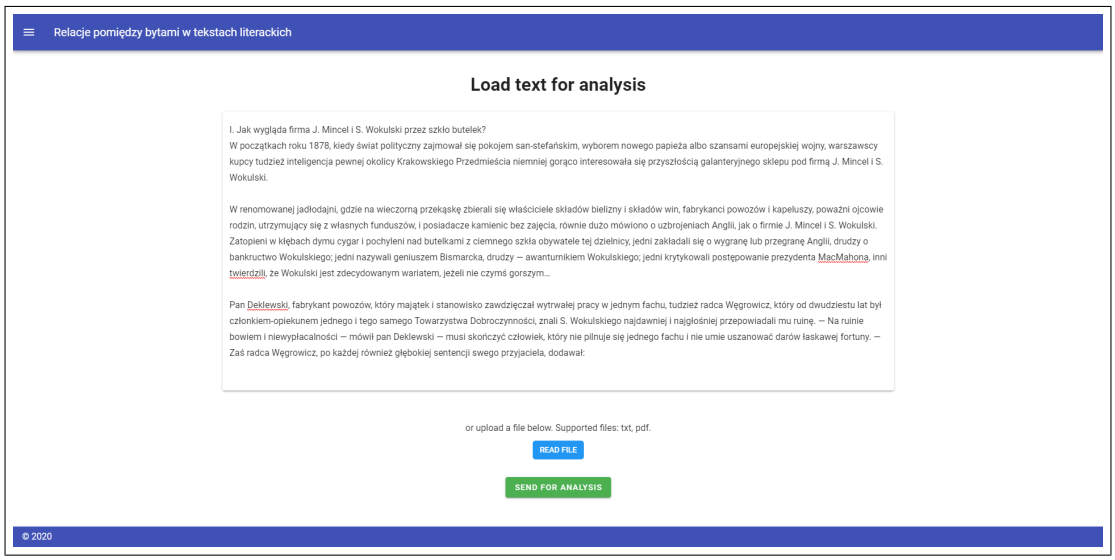
Rysunek 1: Strona domowa aplikacji

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.



Rysunek 2: Interfejs wczytywania tekstu

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 grafu, które zostają wyświetlone.

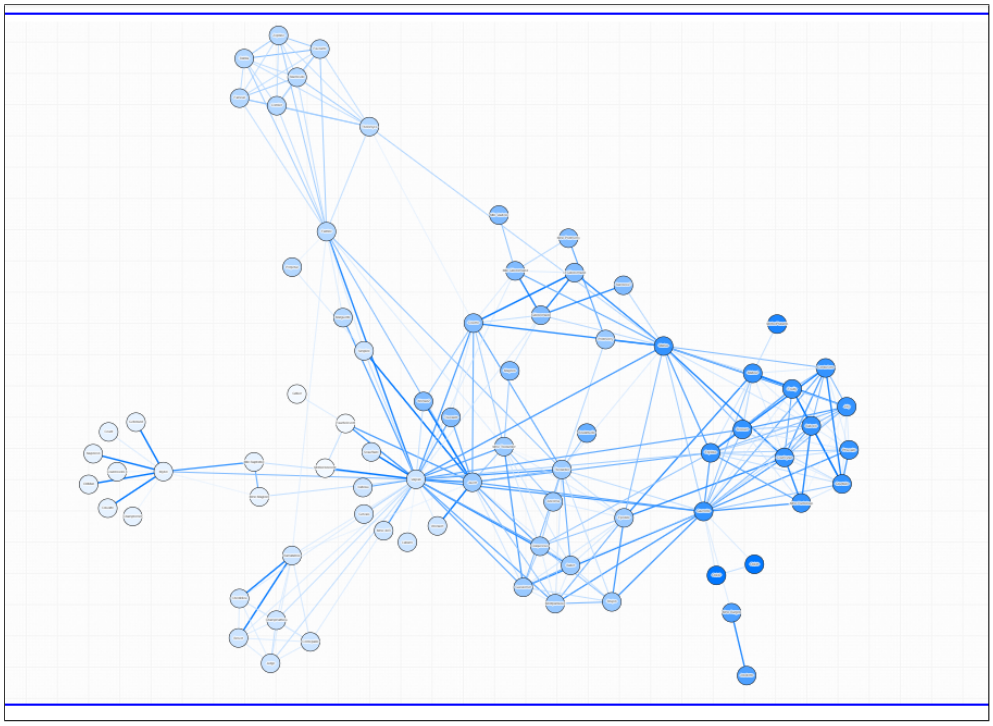
5.3 Prezentacja danych

5.3.1 Metoda prezentacji danych

Do prezentacji wygenerowanych relacji został wybrany graf skierowany siłą (ang. *force-directed graph*). Algorytm rysowania takiego grafu wykorzystuje jedynie informacje zawarte w strukturze danych (węzły oraz połączenia między nimi). Grafy wygenerowane za pomocą takich algorytmów są estetyczne i uporządkowane. Posiadają także niewielką ilość przecinających się połączeń [2]. Te cechy są szczególnie przydatne w przypadku wizualizowania dużych zbiorów danych. Generowanie grafu polega na przypisaniu sił między zbiorem krawędzi i zbiorem węzłów w oparciu o ich względne pozycje, a następnie użyciu tych sił do symulacji ruchu krawędzi i węzłów lub do zminimalizowania ich energii[3]. W pierwszym przypadku siły są liczone w czasie rzeczywistym, czego wynikiem jest graf, w którego układ może ingerować użytkownik (np. poprzez przeciąganie węzłów w inne pozycje). W drugim przypadku wyświetlenie grafu następuje po zminimalizowaniu sił, czego wynikiem jest statyczny graf. Odświeżanie widoku grafu w czasie rzeczywistym znacząco zwiększa czas minimalizacji sił, ponieważ widok grafu jest często odświeżany. Graf statyczny osiąga minimalne siły w dużo krótszym czasie, jednak pozbawiony jest możliwości interakcji z węzłami. W projekcie została zaimplementowana pierwsza metoda, ponieważ kluczowa jest ingerencja użytkownika w pozycje wierzchołków grafu.

Do generowania grafu została wykorzystana biblioteka D3.js [1]. Wybór biblioteki jest umotywowany wysoko konfigurowalną implementacją grafu skierowanego siłą.

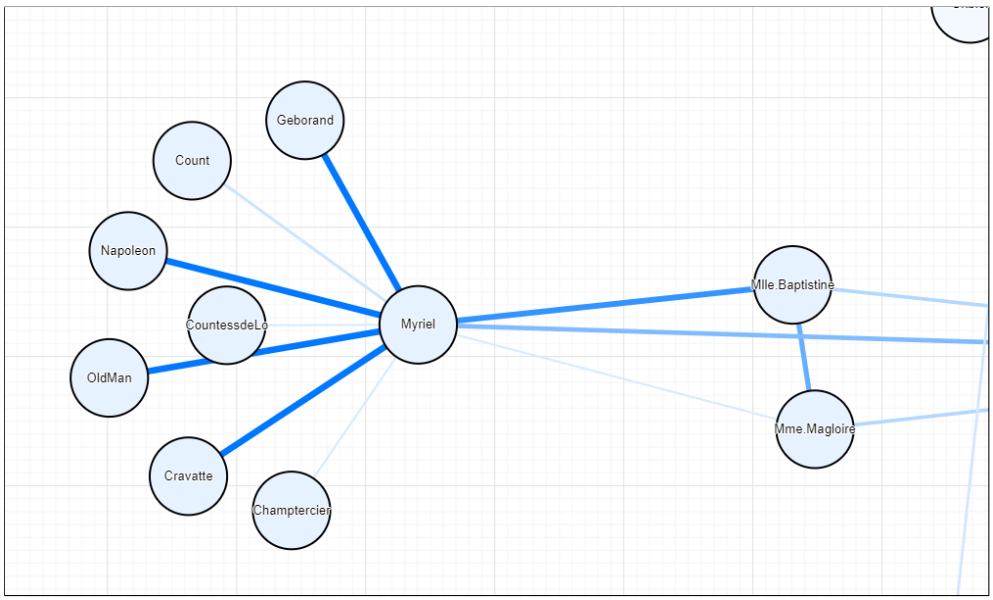
Rysunek 3: Wygenerowany graf współwystępowania postaci w Les Misérables



5.3.2 Metoda prezentacji relacji bytów

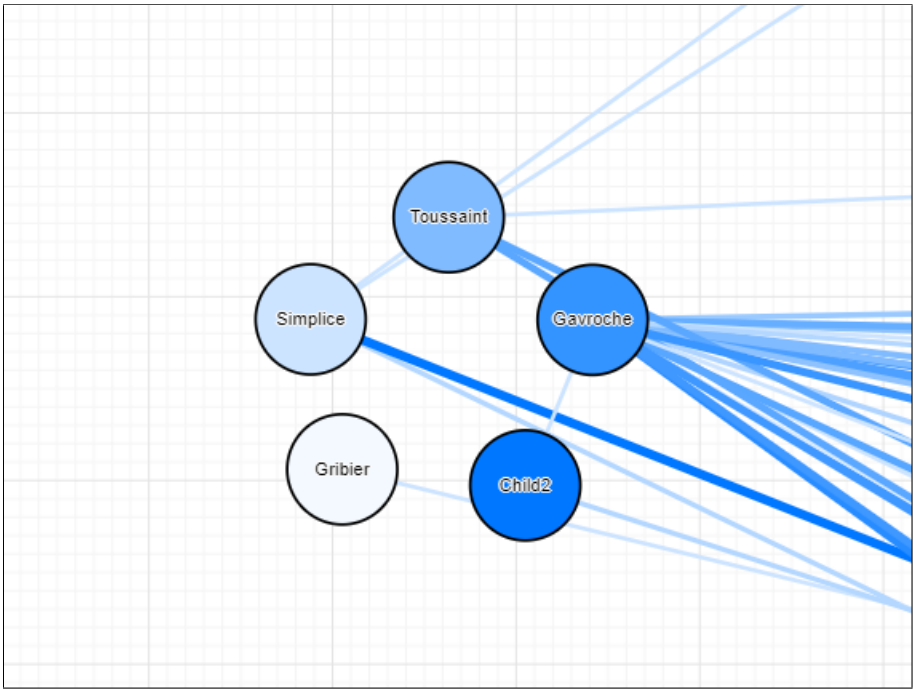
Relacje bytów zostały zobrazowane poprzez zmienny kolor oraz grubość połączenia między poszczególnymi bytami. Im mocniejsza relacja między bytami, tym połączenie jest grubsze oraz posiada bardziej intensywny kolor [rys 4].

Rysunek 4: Wizualizacja relacji bytów



Dodatkowo, wierzchołki reprezentujące byty posiadają kolor zależny od częstości ich występowania. Im częściej dany byt pojawia się w analizowanym teście, tym bardziej intensywny kolor wierzchołka [rys 5].

Rysunek 5: Wizualizacja częstości występowania bytów



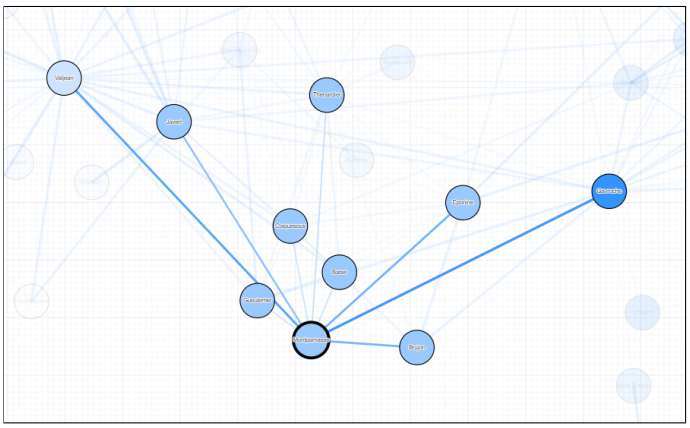
5.4 Dostępne akcje w reprezentacji graficznej

5.4.1 Akcje w obrębie pola grafu

W celu zapewnienia większej czytelności zostały zaimplementowane następujące funkcjonalności:

- przybliżanie oraz oddalanie widoku kółkiem myszy,
- przesuwanie widoku poprzez przeciąganie tła myszą.
- zaznaczanie połączonych węzłów po najechaniu myszą [rys 6]
- przeciąganie węzłów

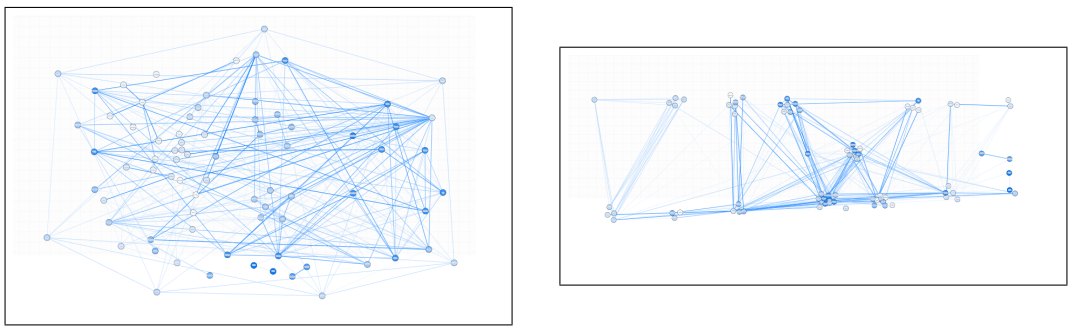
Rysunek 6: Zaznaczenie pojedynczego bytu



5.4.2 Modyfikowanie parametrów wyświetlania grafu

Użytkownik ma możliwość zmiany następujących parametrów wyświetlanego grafu:

- Włączanie i wyłączanie sił w grafie – Podczas gdy siły są wyłączone, użytkownik może dowolnie przestawiać wierzchołki[rys 7].
- Zmiana sprężystości grafu – Modyfikacja tego parametru zmienia siłę oddziałującą na wierzchołki grafu.
- Czułość krawędzi – możliwość określenia jak bardzo znaczące krawędzie będą wyświetlane
- Czułość wierzchołków – możliwość określenia jak bardzo znaczące wierzchołki będą wyświetlane



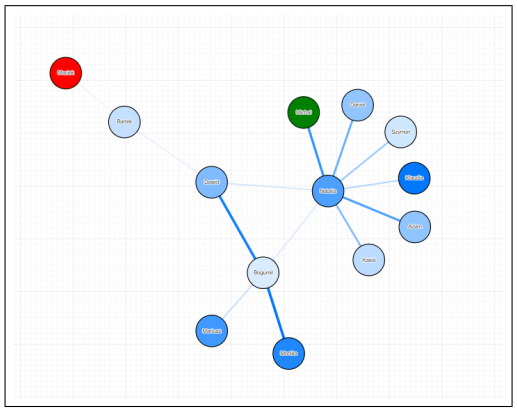
Rysunek 7: Przestawione wierzchołki po wyłączeniu sił

5.4.3 Łączenie mylnie rozdzielonych bytów

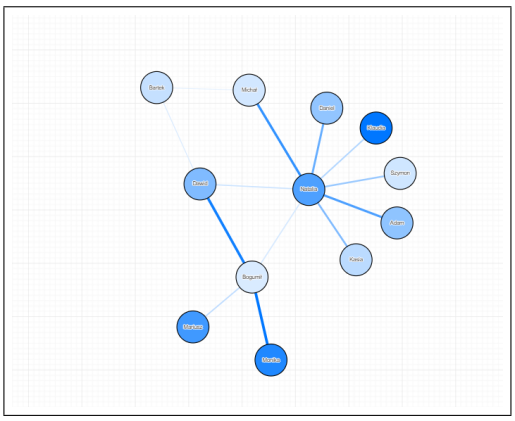
Użytkownik posiada możliwość ręcznego złączenia dwóch bytów w jeden [rys 9]. Funkcjonalność ta jest przydatna w przypadku gdy narzędzie analizujące tekst mylnie rozróżni jeden byt jako dwa osobne. Łączenie bytów polega na usunięciu pierwszego z nich oraz przypisaniu odpowiednich połączeń do bytu drugiego. Wynik takiej operacji został przedstawiony na rysunku 8. Wszelkie zmiany dokonane przez użytkownika mogą zostać przywrócone przyciskiem *Przywróć zmiany*. W celu zwiększenia przejrzystości, byt który ma zostać włączony do drugiego zmienia kolor na czerwony oraz byt który ma zostać pozostawiony zmienia kolor na zielony.

5.5 D3.js - biblioteka wizualizacji danych

D3.js to znacznie więcej niż biblioteka do wizualizacji, a raczej całe podejście do tworzenia dokumentów opartych na danych. Nie jest to monolit, D3 w obecnej wersji jest biblioteką modułową, zawiera w sobie szereg narzędzi wspomagających operacje na danych: takich jak przygotowanie skali, renderowanie wykresów, grafów i map. Biblioteka ta pozwala na dodanie animacji, stylów i customowych zachowań. D3.js jest bardzo uniwersalne ale przychodzi to kosztem dużej inwestycji w naukę biblioteki i zrozumienia powiązań wszystkich modułów. Dzięki podejściu D3.js cała biblioteka jest niezwykle lekka, biorąc pod uwagę możliwości. W najnowszej wersji waży niewiele ponad 70kB w wersji min. Wiele innych bibliotek bazuje na D3.js, rozszerzając pewne funkcje, często specjalizując się i ułatwiając konkretne aspekty wizualizacji danych.

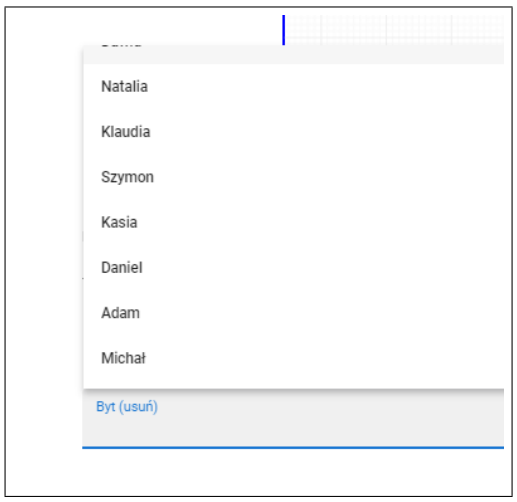


(a) Przed złączeniem

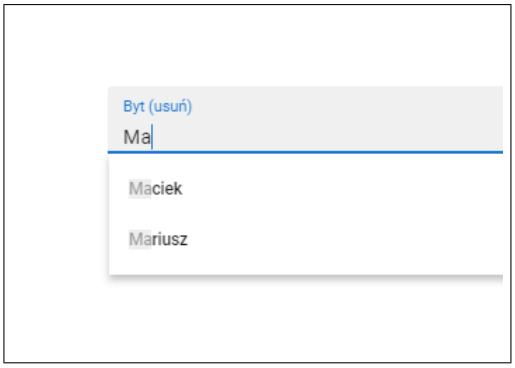


(b) Po złączeniu

Rysunek 8: Efekt złączenia bytów



(a) Lista



(b) Autouzupełnianie

Rysunek 9: Wybór bytów do złączenia

5.6 Zapis do plików

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

5.6.1 Zapis modelu wykresu do pliku GDF

GDF to format tekstowy przypominający CSV. Pozwala definiować węzły oraz krawędzi grafu. Wspiera przechowywanie wielu atrybutów o różnych typach i odnoszących się zarówno do krawędzi, jak i węzłów. Opis grafu w tym formacie może zostać zaimportowany do narzędzia *Gephi*.
Plik składa się z dwóch sekcji:

- definicje węzłów,
- definicje krawędzi

Każda z sekcji posiada swój nagłówki w formacie:

- nodedef> atr1 typ1, atr2 typ2, ...
- edgedef> atr1 typ1, atr2 typ2, ...

w naszym przypadku jest to:

- nodedef> name VARCHAR, label VARCHAR, class VARCHAR, value DOUBLE
- edgedef> source VARCHAR, target VARCHAR, type VARCHAR, value DOUBLE

Poniżej nagłówków znajdują się deklaracje poszczególnych krawędzi oraz węzłów składające się z atrybutów zdefiniowanych w nagłówku rozdzielonych przecinkami. Każdy wiersz odpowiada jednemu elementowi grafu.

5.6.2 Zapis modelu wykresu do pliku JSON

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

```
{
    "nodes": [
        {
            "name": "Pawel",
            "occurrence": 50
        },
        {
            "name": "Przemek",
            "occurrence": 45
        },
        {
            "name": "Wojtek",
            "occurrence": 25
        },
        ...
    ],
    "links": [
        {
            "source": 0,
            "target": 1,
            "value": 67,
        },
        {
            "source": 0,
            "target": 2,
            "value": 15,
        },
        {
            "source": 0,
            "target": 3,
            "value": 98,
        },
        ...
    ]
}
```

Sekcja *nodes* odpowiada za przekazanie danych wszystkich węzłów bytów wraz z ich nazwami(name) oraz sklasyfikowaną po stronie backendu częstością występowania(occurrence). Częstość występowania zależna jest od maksymalnej wartości występowania postaci.

Sekcja *links* odpowiada za informacje dotyczące własności połączeń. Opisuje ona między którymi wierzchołkami następuje połączenie(source i target), wartość value, która określa moc połączenia(im większa tym silniejsze połączenie). Moc połączeń jest klasyfikowana odnosząc się do maksymalnej wartości połączenia.

5.6.3 Eksport wykresu do pliku graficznego

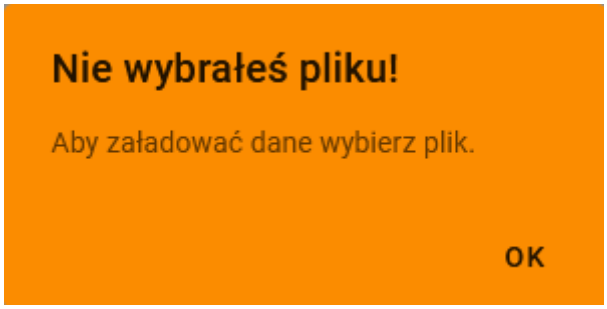
Eksport wykresu odbywa się poprzez generowanie pliku PNG przez przeglądarkę. Jest on generowany na podstawie obecnej zawartości tagu *¡svg¿* w którym znajduje się renderowany wykres. Plik graficzny zawiera legendę oraz wszystko co jest widoczne na wykresie w czasie kliknięcia przycisku *Zapisz graf do pliku graficznego*. Takie działanie umożliwia ustawienie grafu w odpowiadającym użytkownikowi stanie i eksport tego grafu, jego części lub zbliżenia na daną część do pliku graficznego.

5.7 Odczyt z pliku

Struktura pliku jest identyczna do tej opisanej w rozdziale 5.6.2. Przy wczytywaniu z pliku użytkownik wybiera plik z dysku z danymi wykresu. W przypadku gdy ktoś niepo-

Rysunek 10: Element UI odpowiadający za wczytanie danych grafu

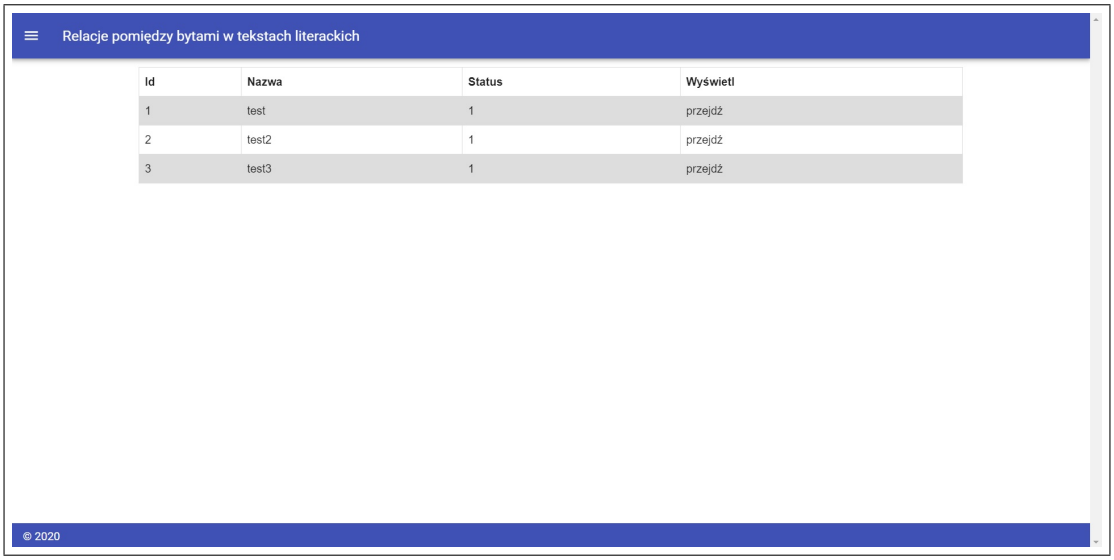
prawnie wypełni pole z wyborem pliku i kliknie przycisk Wczytaj dane z pliku zostanie mu wyświetlona informacja ukazana na rysunku 11



Rysunek 11: Komunikat informujący o braku pliku do wczytania

5.8 Lista grafów

Po wczytaniu tekstu zostaje on zapisany w bazie danych na serwerze. W zakładce znajduje się lista z wygenerowanymi już grafami. Można je przeglądać i otwierać. Po wybraniu i kliknięciu w interesujący nas graf, zostajemy przeniesieni do jego wizualizacji. Tam możemy dokonać jego przekształceń, edycji oraz zapisu do pliku.



Rysunek 12: lista wygenerowanych grafów

5.9 Metoda szukania powiązania

Ekstrakcja bytów z przeanalizowanego tekstu odbywa się z wykorzystaniem narzędzia *NER*. W wynikowym pliku XML zliczane są wystąpienia wszystkich tagów oznaczonych kategoriami:

- subst – rzeczownik,
- m1 – męskoosobowy,
- m2 – męskożywotny,
- f – żeński,
- n – nijaki,

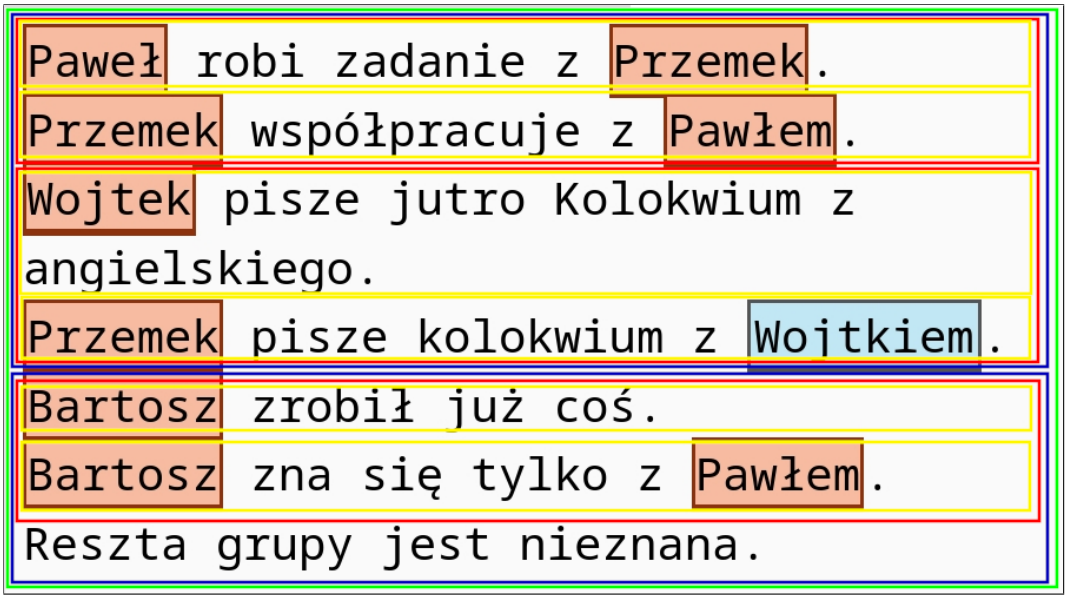
oraz zawierające pole *nam_liv* z wartością większą od 0.

Opracowano dwa warianty algorytmu polegającego na wydzielaniu z XML-a bytów występujących w poruszającym się po tekście oknie o zmiennej wielkości.

W pierwszej metodzie algorytm rozpoczyna się od zsumowania wystąpień wszystkich bytów w całym wprowadzonym tekście, następnie dokonuje się podziału tekstu na pół. W otrzymanych częściach liczone są wystąpienia bytów. Czynność jest powtarzana aż do osiągnięcia fragmentów tekstu wielkości jednego zdania. Schemat podziału tekstu pokazano na rysunku 13. Wystąpieniom bytów w danym etapie analizy (rozmiar analizowanego fragmentu tekstu) przydzielane są wagi według określonej charakterystyki (stała, liniowa, kwadratowa, etc.) zachowując prawidłowość: im mniejszy analizowany fragment tekstu, tym większa waga.

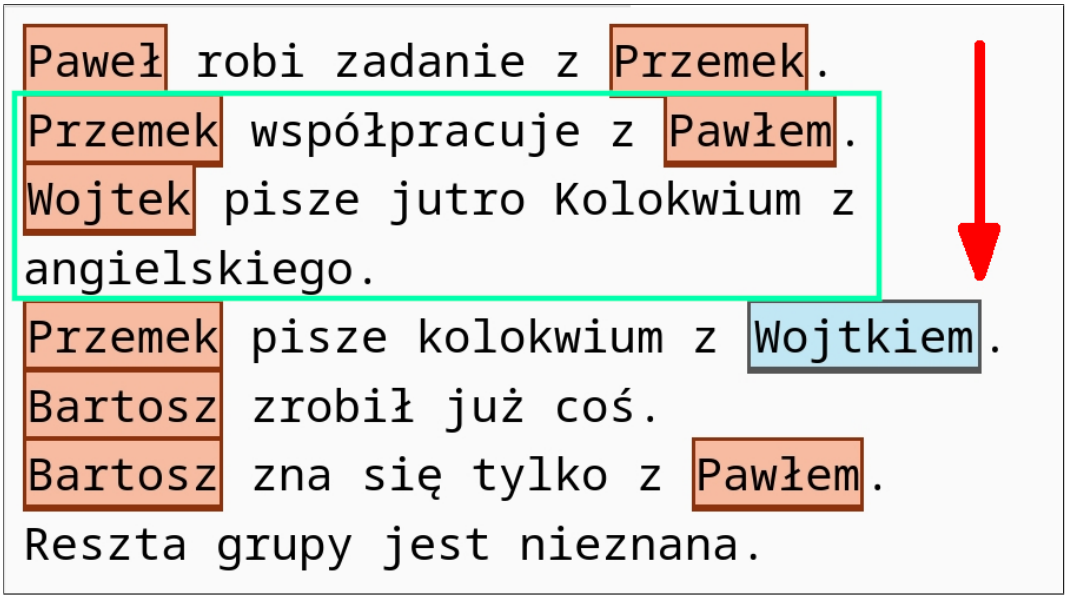
Druga metoda polega na przemieszczaniu okna po tekście począwszy od pierwszego zdania, aż do końca tekstu w krokach co jedno zdanie, tak jak zaprezentowano na rysunku 14. W każdym kroku, z tekstu znajdującego się w obrębie okna, wydzielane i sumowane będą wystąpienia poszczególnych bohaterów utworu. Każdemu wystąpieniu przydzielana jest waga zależna od rozmiaru okna – po zakończeniu analizy oknem o największym rozmiarze następuje zmniejszenie okna oraz zwiększenie wagi wystąpienia bytu i powtórzenie analizy z nowym oknem – większą wagę będą miały dwa byty występujące na obszarze

Rysunek 13: Metoda Pierwsza, podział tekstu



dwóch zdań, niż inne, przeanalizowane na etapie okna o rozmiarze np. 50 zdań. Wystąpienia postaci w obrębie jednego rozmiaru okna, wraz z ilością wystąpień, charakteryzują relację między tymi postaciami. Po osiągnięciu rozmiaru okna równego jednemu zdaniu, relacje pomiędzy bytami z wszystkich etapów są sumowane z uwzględnieniem wag, co daje finalny efekt.

Rysunek 14: Metoda Druga, podział tekstu



Raz wygenerowane z narzędzia *NER* pliki XML zostają poddane analizie przez algorytm, a wyniki są zapisywane w formacie JSON i przechowywane w serwerze.