

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: INFORMATYKA (INF)
SPECJALNOŚĆ: GRAFIKA I SYSTEMY MULTIMEDIALNE (IGM)

PRACA DYPLOMOWA
MAGISTERSKA

Metoda automatycznej oceny podobieństwa
semantycznego tekstów na potrzeby analizy
dyskursu publicznego

Automatic assessment of semantic similarity of
texts for the analysis of public discourse

AUTOR:

Marcin Kotas

PROWADZĄCY PRACĘ:
dr inż. Tomasz Walkowiak K30Wo4Do3

Streszczenie

Lorem ipsum dolor sit amet eleifend et, congue arcu. Morbi tellus sit amet, massa. Vivamus est id risus. Sed sit amet, libero. Aenean ac ipsum. Mauris vel lectus.

Nam id nulla a adipiscing tortor, dictum ut, lobortis urna. Donec non dui. Cras tempus orci ipsum, molestie quis, lacinia varius nunc, rhoncus purus, consectetur congue risus.

Słowa kluczowe: raz, dwa, trzy, cztery

Abstract

Lorem ipsum dolor sit amet eleifend et, congue arcu. Morbi tellus sit amet, massa. Vivamus est id risus. Sed sit amet, libero. Aenean ac ipsum. Mauris vel lectus.

Nam id nulla a adipiscing tortor, dictum ut, lobortis urna. Donec non dui. Cras tempus orci ipsum, molestie quis, lacinia varius nunc, rhoncus purus, consectetur congue risus.

Słowa kluczowe: one, two, three, four

Spis treści

1. Wstęp	8
1.1. wprowadzenie	8
1.2. Cel i zakres pracy	8
1.3. Układ pracy	8
2. Korpus	9
2.1. Format danych	9
2.2. Przetwarzanie danych	9
2.3. Uzupełnienie danych o posłach	10
2.4. Końcowy format korpusu	10
3. Reprezentacja wektorowa	11
3.1. Wektory zdań	12
3.2. Wektory wypowiedzi	14
3.2.1. LexRank	14
3.2.2. TF-IDF	15
3.3. Alternatywne rozwiązania	16
3.3.1. Sieć konwolucyjna	16
3.3.2. TF-IDF	17
4. Analiza tematyczna	18
4.1. Redukcja wymiarów	18
4.2. Grupowanie	19
4.3. Reprezentacja tematu	21
5. Ocena skuteczności modeli	24
5.1. Ewaluacja spójności reprezentacji tematów	24
5.2. Ewaluacja spójności rozkładu tematów	25
5.3. Końcowa ocena	26
5.3.1. Ocena liczby przyporządkowanych dokumentów	26
5.3.2. Ocena liczby tematów	26
5.3.3. Implementacja	27
5.4. Porównanie z LDA	28
6. Aplikacja	30
Literatura	31
A. Instrukcja wdrożeniowa	33
B. Opis załączonej płyty CD/DVD	34

Spis rysunków

3.1. Mechanizm atencji — wizualizacja wag poszczególnych słów podczas analizy słowa „it”. Źródło: https://jalammar.github.io/illustrated-transformer	12
3.2. Architektura modelu Sentence-BERT[16]	13
3.3. Układ modeli nauczyciel-uczeń w procesie uczenia wielojęzykowego SBERT[17]	13
3.4. Dystrybucja długości wypowiedzi (zakres do 2000 słów)	14
3.5. Graf ważony miarą podobieństwa między zdaniami (wierzchołkami)[9]	15
3.6. Suma wartości tf-idf w zależności od liczby wykrytych słów (zakres do 600 słów): a) dane nieznormalizowane z dopasowaną krzywą, b) dane znormalizowane krzywą	16
4.1. Dwuwymiarowa reprezentacja danych przez UMAP w zależności od wartości parametru n_neighbors	19
4.2. Minimalne drzewo rozpinające	20
4.3. Dendrogramy hierarchii klastrów: a) na podstawie drzewa rozpinającego, b) skondensowane	21
4.4. Wykryte klastry na wektorach: a) dwuwymiarowych, b) pięciowymiarowych	22
5.1. Wyniki pomiarów dla LDA: a) C_{NPMI} , b) C_{UMass}	28

Spis tabel

4.1. Reprezentacje wybranych tematów: a) bez MMR, b) z MMR	23
5.1. Wyniki pomiarów dla LDA w zależności od liczby tematów	29
5.2. Reprezentacje pięciu losowo wybranych tematów dla LDA: a) 80 tematów, b) 200 tematów	29

Spis listingów

5.1.	Algorytm obliczania spójności tematów	25
5.2.	Funkcja obliczająca spójność s_c	27
5.3.	Obliczanie finalnej oceny spójności tematów dla listy modeli	27

Skróty

PPC (ang. *Polish Parliamentary Corpus*)

NLP (ang. *Natural Language Processing*)

Rozdział 1

Wstęp

1.1. wprowadzenie

1.2. Cel i zakres pracy

Skonstruowanie narzędzia w języku Python do znajdowania różnic i podobieństw między wypowiedziami pozyskanyymi z korpusu dyskursu parlamentarnego. Wykorzystanie metod semantyki dystrybucyjnej do dokonywania analizy semantycznej.

1.3. Układ pracy

Rozdział 2

Korpus

W pracy skorzystano z Korpusu Dyskursu Parlamentarnego[15], zwanym dalej PPC. Korpus zawiera dane sejmowe wygenerowane z zapisów stenograficznych z lat 1991–2019 oraz ze zdigitalizowanych transkrypcji z lat 1918–1990. Zawarte są również dane z senatu oraz posiedzeń komisji i interpelacji, jednak w tej pracy skupiono się na danych z posiedzeń plenarnych sejmu III RP. Są to posiedzenia od początku Sejmu I Kadencji (25 listopada 1991), wybranego w pierwszych po wojnie w pełni wolnych wyborach.

2.1. Format danych

Korpus dostępny jest w formacie bazującym na XML TEI P5, opracowanym na potrzeby NKJP (*Narodowy Korpus Języka Polskiego*). Każde posiedzenie udokumentowane jest szeregiem plików reprezentujących inne aspekty analizy tekstu. Dostępne są m.in. znaczniki morfosyntaktyczne, lematy, czy grupy syntaktyczne. Pełna struktura opisana została w [14], natomiast na potrzeby tej pracy przetworzone zostały jedynie następujące pliki:

- header.xml — metadane posiedzenia (data, lista mówców),
- text_structure.xml — kolejne wypowiedzi z oznaczonym mówcą.

Adnotacje lingwistyczne nie zostały uwzględnione, gdyż ze względu na specyfikę poruszanego problemu wymagane są surowe, nieoznakowane teksty.

2.2. Przetwarzanie danych

W celu ograniczenia liczby wypowiedzi, które nie są istotne z punktu widzenia analizy, wykonano szereg czynności mających na celu odfiltrowanie nierzeczywitych fragmentów.

Zdarza się, że w trakcie wypowiedzi występują wtrącenia innych osób. Takie komentarze są jednak odpowiednio oznakowane (#komentarz, czy #głosZSali). Dodatkowo, wszelkie komentarze do wypowiedzi występują jako osobny wpis, którego treść zawarta jest w nawiasach. Takie fragmenty są usuwane, aby nie zaburzyć treści wypowiedzi danej osoby.

Ponadto przyjęto, że wypowiedzi marszałków oraz wicemarszałków ograniczają się do zarządzania procedowaniem sejmu, więc nie wprowadzają żadnej wartości. Na końcu usunięto wypowiedzi, których liczba znaków nie przekracza dwustu. Próg ten wyznaczony został empirycznie tak, aby odrzucić wypowiedzi nie zawierające żadnej konkretnej informacji.

2.3. Uzupełnienie danych o posłach

Korpus uzupełniono o dane na temat osób wypowiadających się. W tym celu skorzystano z internetowego Archiwum Danych o Posłach¹. Na stronie dostępne są dane z różnym poziomem szczegółowości o posłach wszystkich kadencji od roku 1952. Z archiwum pobrano następujące informacje dla wszystkich posłów III RP: klub parlamentarny, lista wyborcza, partia oraz okręg wyborczy.

Dane o posłach powiązano ze wpisami w korpusie na podstawie imienia i nazwiska mówcy. W ten sposób udało się dopasować zdecydowaną większość wypowiadających się — ponad 91%. W pozostałych przypadkach wypowiadają się zwykle osoby nie z rządu, np. prezydent.

Dokonano ręcznej weryfikacji spójności danych o okręgach wyborczych i uwspółniono okręgi z większych miast. Następnie do każdego okręgu dopasowano województwo.

2.4. Końcowy format korpusu

Po przetworzeniu korpusu otrzymano 291415 wypowiedzi na przestrzeni 28 lat. Dla każdej z nich dostępne są następujące informacje:

- id — identyfikator wypowiedzi, który składa się z identyfikatora posiedzenia (z PPC) połączzonego z tagiem wypowiedzi (np. div-123),
- mówca — imię i nazwisko tak, jak występuje bezpośrednio w PPC,
- data — data posiedzenia, z którego pochodzi dana wypowiedź,
- text — treść wypowiedzi,
- poseł* — imię i nazwisko posła wzięte z archiwum,
- klub* — klub parlamentarny, do którego należy poseł,
- lista* — lista wyborcza, z której startował poseł,
- partia** — partia, do której należy poseł,
- okręg* — okręg wyborczy posła,
- kadencja — kadencja, z której pochodzi dana wypowiedź

* dostępne dla ok. 91% wypowiedzi

** dostępne dla ok. 18% wypowiedzi

¹<https://orka.sejm.gov.pl/ArchAll2.nsf> (dostęp dnia 14 maja 2021)

Rozdział 3

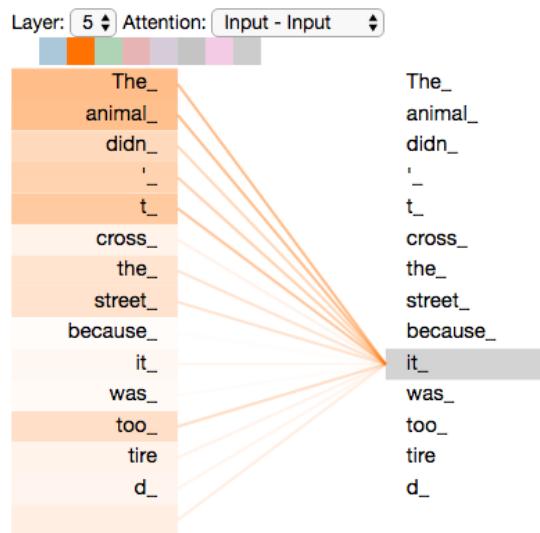
Reprezentacja wektorowa

W zagadnieniach NLP (ang. *Natural Language Processing*) słowa wyrażone są w postaci wektorów liczb rzeczywistych (ang. *word embedding*). W powstałej przestrzeni odległość między wektorami reprezentującymi podobne słowa jest mniejsza, niż odległość między słowami o różnym znaczeniu. Tradycyjne metody opierają się na semantycznej dystrybucji. Zgodnie z tą teorią, słowa występujące w podobnym kontekście mają podobne znaczenie. Bazując na tej zasadzie skonstruowano wiele algorytmów, zdolnych do wygenerowania wektorów słów analizując dużą ilość nieustrukturyzowanych tekstów. Architektura takich modeli opiera się na sieciach neuronowych próbujących przewidzieć słowo w zależności od kontekstu (modele CBOW — ang. *Continuous bag-of-words*) lub kontekst w zależności od słowa (ang. *skip-gram*)[12], gdzie kontekst to otaczające słowa. Wadą tych modeli jest fakt, że są one niekontekstowe. Oznacza to, że po nauczeniu modelu, gdy chcemy go wykorzystać do obliczenia wektora danego słowa, to otrzymany wektor nie bierze pod uwagę kontekstu, w jakim występuje to słowo. Przykładowo słowo „zamek” będzie reprezentowane przez taki sam wektor niezależnie od tego, czy w danym zdaniu odnosi się do zabezpieczenia antywłamaniowego, czy też widowiskowej dolnośląskiej budowli obronnej.

Nie oznacza to jednak, że modele bazujące na *bag-of-words* i *word-to-vec* dają złe wyniki. Wręcz przeciwnie, jak pokazano w [22] modele te nie tylko dają dobre rezultaty, ale wraz ze wzrostem liczby dokumentów dodatkowe wstępne przetwarzanie (lematyzacja) przestają poprawiać dokładność. Należy jednak mieć na uwadze, iż w tej pracy uzyskane wektory posłużyły do nadzorowanego wytrenowania klasyfikatora opartego o wielowarstwowy perceptron. Nie jest jasne, czy podczas nienadzorowanego wykrywania tematów wektory te spiszą się równie dobrze.

W 2018 r ukazała się praca prezentująca BERT (ang. *Bidirectional Encoder Representations from Transformers*)[8]. Był to pierwszy model kontekstowy. Oparty jest na transformerach[21] — mechanizmach wykorzystujących bloki atencji. Mechanizmy te wykrywają zależności między słowami w zdaniu. Do warstwy atencji wszystkie słowa wprowadzane są równolegle, a na wyjściu dla każdego słowa otrzymujemy ważoną sumę wektorów wszystkich słów. Im większe powiązanie słowa ze słowem analizowanym, tym większa waga. Przykład zilustrowany na rysunku 3.1 przedstawia wagę poszczególnych słów w zdaniu „The animal didn't cross the street because it was too tired” w kontekście słowa „it”. Zaimek ten może odnosić się zarówno do zwierzęcia z początku zdania (ang. *The animal*), jak i ulicy (ang. *the street*). Mechanizm przypisał największą wagę do zwierzęcia, co jest poprawnym działaniem w kontekście tego zdania.

Równoległe wprowadzanie wszystkich słów wymusza górną ograniczenie na długość zdania. Dla większości modeli bazujących na transformerach limit wynosi 512 tokenów. Tokenem może być zarówno słowo, jak i część zdania typu przecinek, stąd faktyczny limit liczony w słowach jest jeszcze niższy. Jest to mocno ograniczający limit w kontekście wykorzystywanego korpusu,



Rys. 3.1: Mechanizm atencji — wizualizacja wag poszczególnych słów podczas analizy słowa „it”.
Źródło: <https://jalammar.github.io/illustrated-transformer>

w którym wypowiedzi często złożone są z kilkudziesięciu zdań. Problem ten opisywany jest szerzej w rozdziale 3.2.

3.1. Wektory zdań

Modele bazujące na BERT stworzone są z myślą generowania wektorów dla każdego tokenu wejściowego. Otrzymane w ten sposób wektory kodują informacje semantyczne o wszystkich słowach zdania, jednak nie zawierają informacji o samym zdaniu. Jedną z możliwości jest wykorzystanie wartości wektora dla tokenu CLS — jest to token wykorzystywany podczas uczenia modelu i reprezentuje znaczenie zdania w kontekście zadań klasyfikacji. Można również uśrednić wektory wszystkich słów, jednak nie wszystkie słowa w zdaniu niosą tak samo dużo znaczenia.

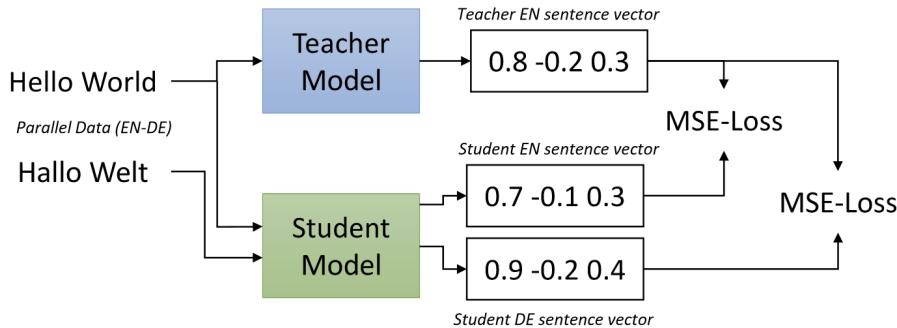
Rozwiążanie tego problemu proponują autorzy Sentence-BERT[16]. Model SBERT złożony jest z dwóch modli BERT połączonych w syjamską sieć. Na wyjściu każdego modelu BERT dołożona jest warstwa pooling zwracająca wektor ustalonego rozmiaru dla zdań różnych długości. Najlepszą strategią okazała się metoda uśredniania wektorów wszystkich tokenów. Następnie wyjściowe dwa wektory wynikowe są łączone i obliczana jest funkcja straty klasyfikatorem softmax. Architektura ta przedstawiona została na rysunku 3.2. W celu uzyskania wektorów kodujących semantykę całych zdań, model uczony jest na korpusach zawierających pary zdań oznaczone pod kątem podobieństwa. Optymalizuje się odległość między podobnymi zdaniami, jednocześnie maksymalizując odległość między zdaniami różnymi.

Ze względu na specyfikę potrzebnego korpusu do trenowania modeli SBERT (oznaczone pary zdań) nie istnieją obecnie modele wytrenowane w języku polskim. Jednak w roku 2020 ci sami autorzy opracowali metodę przenoszenia wiedzy (ang. *knowledge distillation*) z nauczzonego modelu angielskiego na modele wielojęzykowe[17]. Opiera się na koncepcji, według której te same zdania w różnych językach powinny być reprezentowane przez podobne wektory w tej samej przestrzeni. Przekazanie wiedzy polega na minimalizacji różnicy wyników między modelem nauczającym, a modelem uczącym się (ang. *teacher-student*) — Rys. 3.3. W tym przykładzie modelem-nauczycielem jest SBERT wytrenowany na angielskim korpusie, natomiast uczniem jest model XLM-R (XLM-RoBERTa). Jest to wielojęzykowy model bazujący na



Rys. 3.2: Architektura modelu Sentence-BERT[16]

architekturze RoBERTa (wariacja BERT stworzona przez Facebook), wytrenowany na 2.5TB danych z CommonCrawl w 100 językach, w tym polskim[6]. Model ten zajmuje obecnie ósme miejsce w rankingu KLEJ¹ oraz drugie miejsce po dostrojeniu (ang. *fine-tuning*) na dodatkowych polskich korpusach.



Rys. 3.3: Układ modeli nauczyciel-uczeń w procesie uczenia wielojęzykowego SBERT[17]

Trenowanie modelu ucznia wymaga par zdań w języku angielskim i języku docelowym. Jest to dużo mniejsze ograniczenie, niż w przypadku bezpośredniego uczenia SBERT w docelowym języku. Dostępne są wytrenowane modele², które obejmują również język polski. Zostały one uwzględnione w rankingu modeli generujących reprezentacje zdań w języku polskim[7]. W pracy tej autorzy przetłumaczyli popularny angielski korpus SICK (ang. *Sentences Involving Compositional Knowledge*) zawierający pary zdań oznaczone pod kątem implikacji (ang. *entailment*). Każda para może być albo powiązana (prawdziwość pierwszego zdania oznacza prawdziwość drugiego), neutralna (pierwsze zdanie nie mówi nic o prawdziwości drugiego), albo sprzeczna. Na podstawie tego korpusu stworzono dwa zadania odpowiadające angielskim — SICK-E (zadanie klasyfikacji) oraz SICK-R (zadanie przewidywania dystrybucji podobieństwa, mierzone miarą korelacji Pearsona).

Na potrzeby tej pracy za najbardziej istotne zadanie uznano SICK-R, ponieważ sprawdza skuteczność oceniania podobieństwa semantycznego dwóch tekstów. W tym zadaniu najlepszy wynik uzyskał wielojęzykowy model SBERT xlm-r-distilroberta-base-paraphrase-v1 (aktualne wyniki znajdują się na stronie z kodem źródłowym³).

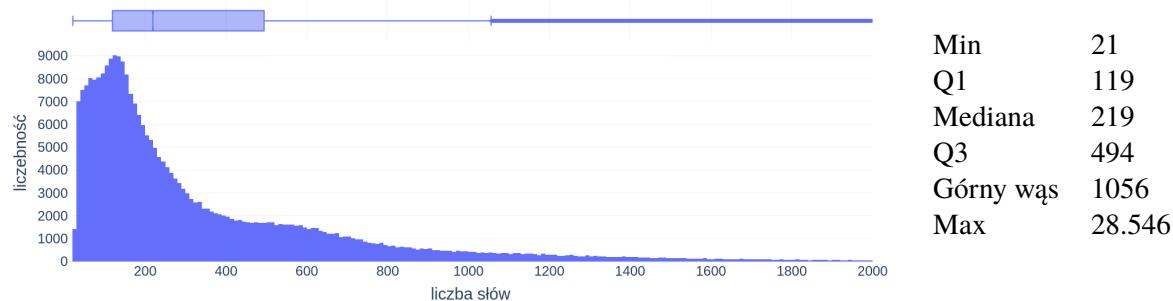
¹<https://klejbenchmark.com/leaderboard> (dostęp dnia 20 maja 2021)

²https://www.sbert.net/docs/pretrained_models (dostęp dnia 20 maja 2021)

³<https://github.com/sdadas/polish-sentence-evaluation> (dostęp dnia 20 maja 2021)

3.2. Wektory wypowiedzi

Modele bazujące na architekturze BERT mają górne ograniczenie na długość zdania wejściowego. Wynosi ono 512 tokenów, co odpowiada około 300 słowom. Dodatkowo, zbiór danych uczących wykorzystywanych przez modele SBERT zawiera zwykle krótsze zdania, przez co skuteczność tych modeli dla pełnego zakresu 512 tokenów może być gorsza niż oczekiwana. Jak pokazano na rysunku 3.4, wypowiedzi sejmowe są często znacznie dłuższe niż ten limit. Mediana liczby słów wynosząca 219 mieści się w limicie, jednak wypowiedzi złożone z więcej niż 300 słów stanowią prawie 39% zbioru.



Rys. 3.4: Dystrybucja długości wypowiedzi (zakres do 2000 słów)

Z tego powodu postanowiono dzielić wypowiedzi na zdania, generować wektory dla każdego zdania, a następnie połączyć je w jeden wektor wynikowy dla całej wypowiedzi. Rozważono następujące strategie:

- średnia,
- średnia ważona,
- wektor pierwszy w rankingu,
- średnia pierwszych pięciu wektorów w rankingu

Do wygenerowania wag oraz rankingu potrzebna jest metoda oceniająca, na ile istotne jest każde zdanie w kontekście całej wypowiedzi. Stworzono w tym celu dwie metody: jedna wykorzystująca algorytm LexRank oraz druga wykorzystująca TF-IDF.

3.2.1. LexRank

LexRank[9] to stworzony w 2004r. stochastyczny algorytm bazujący na teorii grafów, służący do generowania podsumowań długich tekstów. Jako podsumowanie traktowany jest zbiór najistotniejszych zdań z tekstu — jest to metoda ekstrakcyjna, która nie zmienia struktury zdań. Istnieją nowe metody bazujące na uczeniu maszynowym, które jako podsumowanie generują nowy tekst zawierający sens dokumentu, jednak na potrzeby tej pracy wymagany jest algorytm przypisujący wagę oryginalnym zdaniom.

Algorytm konstruuje graf, gdzie wierzchołki odpowiadają zdaniom, a ich wartość obliczana jest algorymem TF-IDF. Krawędzie grafu odpowiadają wartości podobieństwa między dokumentami i mierzone są na miarę kosinusową. Wizualizacja takiego grafu przedstawiona została na rysunku 3.5. Graf przedstawiony w postaci macierzy, po znormalizowaniu wierszy, jest macierzą stochastyczną (macierz kwadratowa, w której każdy wiersz sumuje się do wartości 1), którą można potraktować jako łańcuch Markova. Udowodnione jest, że łańcuch Markova jest zbieżny do rozkładu stacjonarnego, jeśli spełniony jest następujący warunek:

$$\lim_{x \rightarrow \infty} X^n = 1^T r$$



Rys. 3.5: Graf ważony miarą podobieństwa między zdaniami (wierzchołkami)[9]

gdzie X to macierz stochastyczna, a r to rozkład stacjonarny łańcucha Markova. Rozkład stacjonarny można interpretować jako prawdopodobieństwo że skończymy na danym elemencie przechodząc przez łańcuch, niezależnie od stanu początkowego. Możemy go więc potraktować jako wektor centralny (ang. *centrality vector*), gdzie wartość każdego elementu informuje, jak bardzo centralne jest zdanie na danej pozycji.

W oryginalnej pracy autorzy mierzyli odległości pomiędzy wektorami TF-IDF. Nic nie stoi jednak na przeszkodzie, aby w ich miejsce wstawić wektory wygenerowane przez model BERT. Otrzymany w ten sposób wektor centralny posłuży jako wagi wektorów zdań oraz jako ranking.

3.2.2. TF-IDF

Alternatywną metodę ważenia zdań oparto na metodzie TF-IDF. TF odnosi się do *term-frequency* i oznacza częstotliwość występowania danego słowa w dokumencie (oznaczone dalej jako $tf(t)$). IDF oznacza *inverse document-frequency*, zawiera wartości dla każdego słowa, informujące jak bardzo jest ono powszechnie. Im w większej liczbie dokumentów występuje dane słowo, tym mniej informacji niesie o konkretnym dokumencie. Wartość IDF obliczana jest według następującego wzoru:

$$idf(t) = \log \left(\frac{1 + n}{1 + df(t)} \right) + 1$$

gdzie t oznacza słowo, n to liczba wszystkich dokumentów, a $df(t)$ to częstotliwość występowania słowa we wszystkich dokumentach. Wartość $tf \cdot idf(t)$ obliczana jest jako iloczyn $tf(t)$ i $idf(t)$.

W pierwszym kroku wyuczono model na całym korpusie, aby skonstruować pełen słownik oraz wartości macierzy IDF. Następnie podczas generowania wektorów wypowiedzi, obliczana jest macierz tf-idf, gdzie wierszami są kolejne zdania. Dla każdego zdania kolumny są sumowane, dzięki czemu otrzymujemy wektor o długości równej liczbie zdań oznaczający, na ile znaczące są słowa w zdaniu w kontekście całego korpusu (macierz IDF zawiera wartości uwzględniające wszystkie dane). Sama ta wartość nie może posłużyć jednak do porównywania zdań, gdyż dłuższe zdania będą miały wyższy wynik, nawet jeśli będą złożone z samych nieznaczących słów. Podzielenie tych wartości przez liczbę wykrytych słów w danym zdaniu (liczbę niezerowych kolumn w danym wierszu macierzy tf-idf) będzie miało odwrotny efekt — preferowane będą krótkie zdania z małą liczbą słów o wysokim wyniku tf-idf. Dłuższe zdania, które mogą zawierać istotne informacje, będą miały niski wynik, gdyż naturalnie składają się również

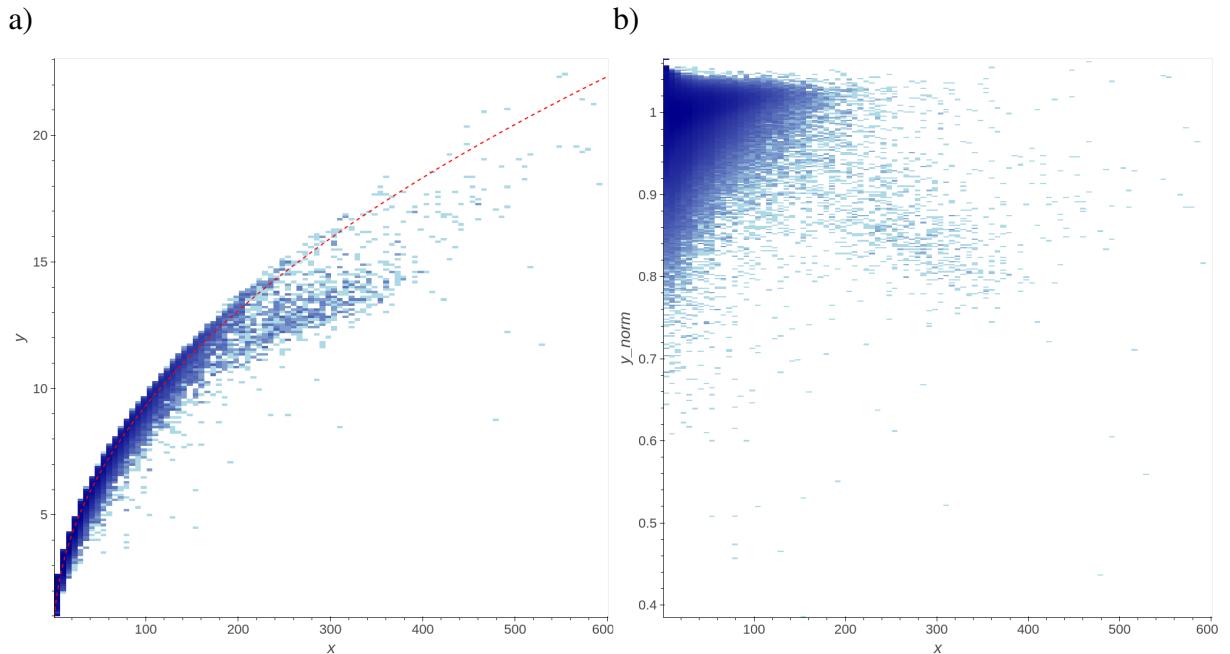
z mniej znaczących słów. Postanowiono więc zbadać, jak zmienia się suma wartości tf-idf w zależności od liczby wykrytych słów. Zależność ta przedstawiona została na wykresie 3.6a. Poprzez aproksymację punktową do danych dopasowana została funkcja potęgowa narysowana na wykresie kolorem czerwonym. Aproksymacji poddane zostały trzy współczynniki według następującego wzoru funkcji:

$$a \cdot x^b + c$$

dzięki czemu otrzymano następującą funkcję, której dobrym przybliżeniem jest pierwiastek kwadratowy:

$$1.003 \cdot x^{0.486} - 0.065$$

Odzwierciedla ona średnią sumę wartości tf-idf dla danej długości zdania. Można więc przyjąć, że zdania poniżej tej krzywej są mniej istotne, niż zdania powyżej. Wartości wszystkich zdań normalizowane są wartością tej funkcji, dzięki czemu nie są promowane ani długie zdania, ani krótkie. Znormalizowane wartości zdań przedstawione zostały na wykresie 3.6b. Wartości te użyte zostały jako wagi wektorów BERT oraz jako ranking zdań.



Rys. 3.6: Suma wartości tf-idf w zależności od liczby wykrytych słów (zakres do 600 słów): a) dane nieznormalizowane z dopasowaną krzywą, b) dane znormalizowane krzywą

3.3. Alternatywne rozwiązania

Modele oparte o transformery zajmują obecnie wysokie miejsca w rankingach[19, 7], jednak nie są to jedyne metody dające dobre wyniki. W celu porównania wyników wygenerowano wektory dwoma alternatywnymi metodami — siecią konwolucyjną oraz standardowym TF-IDF.

3.3.1. Sieć konwolucyjna

Sieci konwolucyjne wykorzystują warstwy połączone filtrami konwolucyjnymi nakładanymi na lokalne cechy. Początkowo wykorzystywane do zagadnień związanych z przetwarzaniem obrazów, znalazły zastosowanie również w pozostałych dziedzinach uczenia maszynowego, w tym NLP. W 2019r. Google udostępniło wielojęzykowy wariant modelu USE (ang. *Universal*

Sentence Encoder)[24] wspierający 16 języków, w tym polski. W pracy tej opisano dwa warianty — jeden wykorzystujący CNN oraz drugi opierający się na transformerach. Model oparty na transformerach osiągnął drugi najlepszy wynik w zadaniu SICK-R w polskim rankingu Polish Sentence Evaluation[7]. Wersja CNN modelu jest szybsza kosztem mniejszej dokładności. Umożliwia ona jednak kodowanie znacznie dłuższych zdań (256 tokenów kontra 100). Oba modele wytrenowane zostały na danych zebranych z forów dyskusyjnych (w postaci pytania i odpowiedzi) oraz na zbiorze SNLI (Stanford Natural Language Inference) przetłumaczonym na pozostałe 15 języków za pomocą Google Translate.

3.3.2. TF-IDF

W przeciwieństwie do omawianych wyżej modeli, w przypadku TF-IDF długość wypowiedzi działa na korzyść algorytmu. Im dłuższa wypowiedź, tym większa szansa na uzyskanie kombinacji słów dokładniej identyfikującej dokument. Algorytm ten nie posiada żadnego górnego ograniczenia na długość dokumentu. Algorytm podczas zliczania liczby wystąpień danego słowa nie bierze pod uwagę odmian słów, przez co słowa „budżet” i „budżetu” są traktowane jako osobne kategorie. Jest to szczególnie istotne w językach słowiańskich, ponieważ wykorzystywanych jest w nich mnóstwo odmian słów.

Aby zminimalizować wynikającą z tego działania utratę informacji, zdecydowano się wstępnie przetworzyć dokumenty przed zliczeniem słów. Wykorzystano w tym celu polski model do biblioteki spaCy — pl_spacy_model_morfeusz[20] udostępniony przez Instytut Podstaw Informatyki Polskiej Akademii Nauk⁴. Biblioteka umożliwia między innymi lematyzację (sprowadzenie słowa do formy podstawowej) każdego słowa w zdaniu. Wykorzystuje przy tym wyuczony model, który rozpoznaje strukturę zdania — przykładowo, czy dane słowo jest w danym kontekście rzeczownikiem, czy czasownikiem. Następnie na podstawie tej wiedzy może określić formę podstawową słowa. Polski model na tym etapie wykorzystuje słownik morfologiczny Morfeusz2[23]. Zbudowany jest on na danych ze Słownika gramatycznego języka polskiego.

Podczas tokenizacji dokumentów generowane są również n-gramy z zakresu (1,3). Oznacza to, że zliczane będą również wystąpienia dwóch oraz trzech kolejnych słów (np. „Unia Europejska”, „Rzecznik praw obywatelskich”). Dodatkowo, przyjęto próg odrzucenia min_df równy 5 oznaczający, że dany token musi wystąpić w minimum pięciu dokumentach, aby był uwzględniony w wynikowej macierzy.

⁴<http://zil.ipipan.waw.pl/SpacyPL> (dostęp dnia 21 maja 2021)

Rozdział 4

Analiza tematyczna

Analiza tematyczna (ang. *topic modeling*) służy do organizowania, wyjaśniania, analizy w czasie, przeszukiwania i streszczania dużych zbiorów danych. Najpopularniejszą metodą modelowania tematów jest LDA (ang. *Latent Dirichlet Allocation*)[2]. Jest to generatywny model probabilistyczny. Dokonuje dyskretyzacji ciągłej przestrzeni tematów na z góry określoną liczbę t tematów i modeluje dokumenty jako kombinację tematów, dla każdego określając prawdopodobieństwo. Każdy temat jest rozkładem prawdopodobieństwa wystąpienia mają powszechnie słowa nie niosące istotnej informacji, np. zaimki czy łączniki. Problem ten jest częściami rozwiązywany listą nieinformacyjnych słów (ang. *stop words*), które są odfiltrowywane z dokumentów. Często lista ta musi być dopasowana do danego zbioru tekstów. Dokumenty reprezentowane są jako zbiór słów, w którym zliczane jest wystąpienie każdego słowa — BOW (ang. *bag-of-words*). Wadą takiego formatu jest fakt, że nie zachowuje informacji o kolejności słów, ani ich semantycznych.

Celem LDA jest znalezienie takich tematów, które mogą posłużyć do odtworzenia oryginalnych rozkładów słów z dokumentów z minimalnym błędem. Model nie rozróżnia pomiędzy słowami informatycznymi, a nieinformacyjnymi, ma za zadanie jedynie jak najlepiej odzwierciedlać oryginalne dokumenty. Z tego powodu najbardziej prawdopodobne słowa w temacie niekoniecznie odzwierciedlają faktyczne tematy dokumentów. Na podstawie analizy LDA otrzymujemy nie tylko klasifikację dokumentu do tematu, lecz cały wektor prawdopodobieństw przynależności dokumentu do każdego z tematów. Wektor ten ma więc rozmiar równy liczbie wszystkich tematów. Jak pokazano w [22], nadzorowane metody klasifikacji wykorzystujące te wektory dają podobne wyniki do metod opartych o *word-to-vec*.

Modele oparte o transformery osiągają najlepsze wyniki w wielu zagadnieniach z dziedziny NLP[19], więc powstały również bazujące na nich metody analizy tematycznej. Jedną z nich jest BERTopic[10], algorytm generujący tematy na podstawie wektorów BERT. Wykorzystuje UMAP[11] w celu redukcji wymiarowości wektorów do 5 wymiarów. Następnie za pomocą algorytmu HDBSCAN[3] wektory grupowane są w klastry. Każda grupa reprezentuje pewien temat. Reprezentacje tematów generowane są za pomocą wariantu TF-IDF, gdzie wszystkie dokumenty w klastrze traktowane są jako jeden dokument i porównywane między sobą. Poszczególne etapy rozwinięte zostały w kolejnych podrozdziałach.

4.1. Redukcja wymiarów

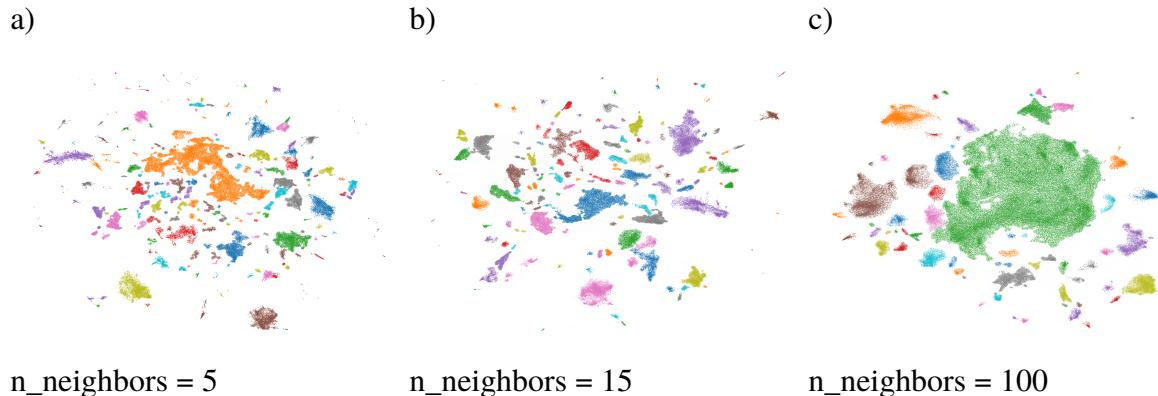
Wektory powstałe w wyniku analizy wypowiedzi za pomocą SBERT mają 768 wymiarów. W przypadku wykorzystania USE jest to 512 wymiarów, natomiast wektory TF-IDF mają tyle wymiarów, ile jest wszystkich tokenów w korpusie (dla wykorzystywanego korpusu jest to ponad

2mln). Wykorzystywany algorytm grupowania HDBSCAN działa lepiej na danych w mniejszym wymiarze, autorzy testowali algorytm do 50 wymiarów[3]. Z tego powodu konieczna jest redukcja wymiaru danych.

Wykorzystano w tym celu algorytm UMAP (ang. *Uniform Manifold Approximation and Projection*)[11]. Oparty jest on o techniki topologicznej analizy danych bazując na geometrii riemannowskiej. Konstruuje rozmytą reprezentację topologiczną wysokowymiarowych danych, a następnie optymalizuje reprezentację tych danych w docelowym wymiarze tak, aby jej rozmyta reprezentacja topologiczna była jak najbardziej podobna mierząc entropią krzyżową. W ten sposób algorytm dąży do tego, aby dane w niskowymiarowej przestrzeni jak najlepiej odzwierciedlały topologiczną strukturę oryginalnych danych. Dzięki temu zachowane są zarówno lokalne, jak i globalne zależności między danymi.

Redukcja do zbyt niskiego wymiaru może skutkować zbyt wielkiej utracie informacji, podczas gdy redukcja do większego wymiaru może dawać gorsze rezultaty podczas grupowania. Postanowiono redukować dane do pięciu wymiarów, tak jak w algorytmie BERTopic. Podczas redukcji wymiarowości wektorów SBERT i USE korzystano z metryki kosinusowej. Dla wektorów TF-IDF zastosowano metrykę Hellingera, która mierzy podobieństwo rozkładów prawdopodobieństwa (wektory tf-idf można traktować jak prawdopodobieństwo znalezienia danego tokenu w dokumencie).

Istotnym parametrem algorytmu UMAP jest `n_neighbors`. Balansuje on między skupieniem się na lokalnej strukturze danych, a globalną strukturą. Ogranicza liczbę sąsiadujących wektorów, które są analizowane podczas nauki struktury rozmaitości topologicznej danych. Dla niskich wartości koncentruje się na lokalnej strukturze nie biorąc pod uwagę pełnego obrazu danych. Wysokie wartości skutkują utratą szczegółów, ukazując bardziej ogólne zależności. Wpływ tego parametru zobrazowany został na rysunku 4.1 dla wartości 5, 15 oraz 100. W celu wizualizacji dane zredukowane zostały do dwóch wymiarów, a kolory odpowiadają klastrom wykrytym przez HDBSCAN (dla minimalnego rozmiaru klastra równego 100). Dla niskiej wartości parametru `n_neighbors` powstało wiele lokalnych grup o niskiej liczbowości. Wraz ze wzrostem wartości parametru lokalne grupy łączą się w coraz większe klastry.



Rys. 4.1: Dwuwymiarowa reprezentacja danych przez UMAP w zależności od wartości parametru `n_neighbors`

4.2. Grupowanie

HDBSCAN to algorytm klastrowania hierarchicznego bazujący na DBSCAN (ang. *Density-based spatial clustering of applications with noise*). W celu zobrazowania działania algorytmu, wybrano ograniczony podzióbór danych. Z klastrów wykrytych na wszystkich danych wylosowano pięć, a z każdego z nich wylosowano dziesięć wektorów. Dodatkowo, wybrano

dziesięć losowych wektorów bez przypisanego klastra. W ten sposób otrzymano 60 dokumentów, które poddano ponownej analizie HDBSCAN (wykorzystując wektory uzyskane z redukcji wymiarów wszystkich dokumentów do 5).

Algorytm określa klastry jako rejony o większej gęstości otoczone rzadziej rozmieszczonymi punktami. Aby określenia gęstości używana jest metryka odległości do najbliższego k -tego sąsiada oznaczona dalej jako $\text{core}_k(x)$. Na jej podstawie stworzono metrykę określającą odległość wzajemnej osiągalności (ang. *mutual reachability distance*):

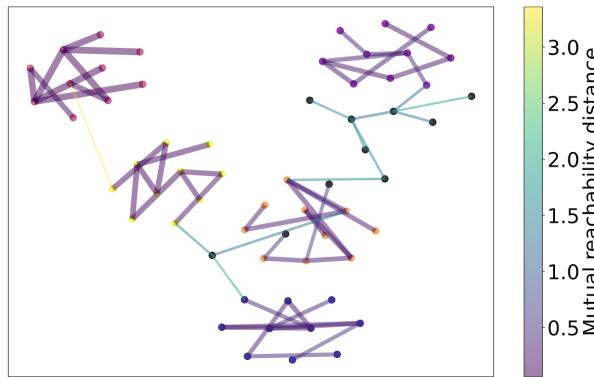
$$d_k(a, b) = \max \{\text{core}_k(a), \text{core}_k(b), d(a, b)\}$$

gdzie:

- $\text{core}_k(x)$ odległość do najbliższego k -tego sąsiada,
- $d(a, b)$ pierwotna metryka odległości.

Dzięki takiej metryce punkty które są blisko siebie pozostają w pierwotnych odległościach, natomiast rzadsze punkty są od siebie odpychane.

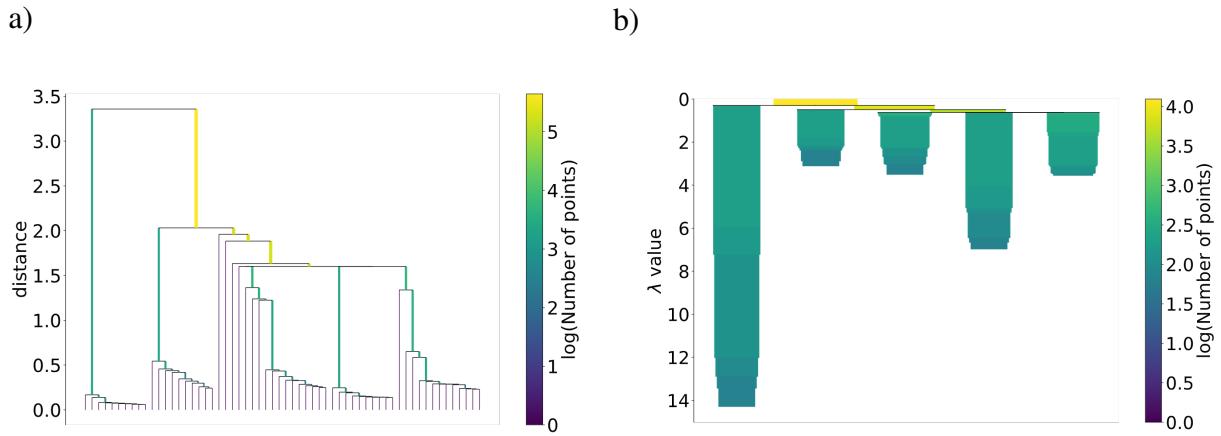
Na podstawie obliczonych odległości konstruowane jest minimalne drzewo rozpinające algorytmem Prima. Wierzchołkami grafu są punkty danych, a wagami krawędzi obliczone odległości. Na rys. 4.2 przedstawiono powstałe drzewo dla wyselekcjonowanych danych. Grubość krawędzi i kolor zależy od wartości odległości, gdzie grubszego linie oznaczają mniejszą odległość. Punkty pokolorowane są zgodnie z pierwotnymi klastrami, a na czarno zaznaczone są punkty, które nie miały klastrów. Zgodnie z oczekiwaniemi, punkty nieprzypisane pierwotnie do żadnego klastra mają znacznie większe odległości od punktów, które tworzą zbiory. Widać też, że klastry nie nachodzą na siebie, a więc dane są dobrze odseparowane.



Rys. 4.2: Minimalne drzewo rozpinające

Następnie powstałe krawędzie łączone są w hierarchiczne grupy, poprzez scalanie kolejnych krawędzi o najmniejszej odległości. Powstałą w ten sposób hierarchię klastrów można zobrazować jako dendrogram (rys. 4.3a). Celem algorytmu jest jednak otrzymanie zbioru klastrów, nie hierarchii. W tym celu HDBSCAN kondensuje hierarchię do mniejszego drzewa. Algorytm analizuje hierarchię i podczas każdego podziału sprawdza powstałe dwie grupy. Jeśli obie grupy mają rozmiar większy niż minimalny (minimalny rozmiar klastra to parametr algorytmu), to taki podział jest traktowany jako faktyczne dwie grupy. Jeśli jednak któraś z powstałych grup ma mniejszy rozmiar niż minimalny to uznaje się, że są to punkty wypadające z klastra niestanowiące osobnej grupy i zostawiana jest jedynie większa grupa. W ten sposób otrzymuje się dużo mniej grup, które wraz ze wzrostem odległości między punktami zmniejszają swój rozmiar. Wynik takiego działania przedstawiony jest na rysunku 4.3b.

Na końcu z powstałego skondensowanego drzewa wybierane są najstabilniejsze klastry. Stabilność klastra określona na podstawie wartości $\lambda = \frac{1}{\text{distance}}$. Dla każdego klastra możemy



Rys. 4.3: Dendogramy hierarchii klastrów: a) na podstawie drzewa rozpinającego, b) skondensowane

oznaczyć λ_{birth} i λ_{death} jako wartości λ odpowiednio w momencie powstania klastra i rozbicia na mniejsze klastry. Dla każdego punktu p w klastrze, możemy zdefiniować λ_p jako wartość λ w momencie, w którym punkt odłączył się od klastra, gdzie $\lambda_{birth} < \lambda_p < \lambda_{death}$. Następnie dla klastra obliczamy jego stabilność jako:

$$\sum_{p \in cluster} (\lambda_p - \lambda_{birth})$$

Aby wybrać końcowe klastry najpierw zaznaczane są wszystkie liście drzewa. Jeśli suma stabilności dwóch łączących się klastrów jest większa niż stabilność powstałego klastra, to ustawiamy stabilność klastra na tą sumę. Natomiast jeśli stabilność złączonego klastra jest większa niż jego potomków, to wybieramy ten klaster i oznaczamy wszystkich potomków. W ten sposób zaznaczamy kolejne klastry aż dojdziemy do korzenia drzewa, a powstałe zaznaczenie to zbiór klastrów. Na rysunku 4.3b zaznaczonych zostało pięć klastrów w kolorze morskim. Pozostałe punkty nienależące do tych klastrów uznawane są za szum i nie mają przypisanych grup.

Parametr k w implementacji algorytmu HDBSCAN nazwany jest `min_samples` i domyślnie jego wartość jest równa wartości `min_cluster_size`. Na rysunku 4.1 wartość ta była równa 100. Zmniejszenie wartości `min_samples` do 1 skutkuje wykryciem większej ilości mniejszych grup. Rezultat przedstawiony został na rys. 4.4a.

Rys. 4.4b przedstawia klastry wykryte na pięciowymiarowych wektorach, a następnie zwizualizowane na dwóch. Wykrywanie klastrów w przestrzeni pięciowymiarowej pozwala na dokładniejsze wykrycie zależności między punktami danych, które mogą zostać utracone przy redukcji do zbyt małej liczby wymiarów. W szczególności, punkty oznaczone jako szum w pięciu wymiarach, w dwóch wymiarach wykrywane są jako klastry.

4.3. Reprezentacja tematu

Każdy temat reprezentowany jest zbiorem słów najlepiej opisujących zawarte w nim dokumenty. Jednocześnie słowa te powinny być specyficzne dla danego zbioru tak, aby widoczne były różnice między tematami. Jest to problem podobny do wyszukiwania słów identyfikujących dokument algorymem TF-IDF. Tym razem jednak chcemy znaleźć słowa identyfikujące zbiór dokumentów na tle innych zbiorów. Autor BERTopic nazwał tą wariację algorytmu c-TF-IDF (ang. *class-based TF-IDF*):

$$c - TF - IDF_i = \frac{t_i}{w_i} \cdot \log \frac{m}{\sum_j^n t_j}$$

gdzie:

a)



b)



Rys. 4.4: Wykryte klastry na wektorach: a) dwuwymiarowych, b) pięciowymiarowych

- t_i częstość występowania słowa t w klasie i ,
- w_i liczba słów w klasie i ,
- m liczba wszystkich pierwotnych dokumentów (niezgrupowanych),
- $\sum_j^n t_j$ suma częstości występowania słowa t we wszystkich klasach.

W ten sposób uzyskano ranking wszystkich słów w danej klasie. Na podstawie tego rankingu wybiera się 5–10 słów z najwyższym wynikiem, które stanowią reprezentację tematu. Często zdarza się jednak, że otrzymane słowa będą bardzo podobne. Przykładowe reprezentacje powstałe tą metodą przedstawione zostały w tabeli 4.1. Aby zminimalizować liczbę podobnych słów w reprezentacji tematu i jednocześnie zdwywersyfikować opis tematu, w BERTopic zastosowany jest algorytm MMR (ang. *Maximal Marginal Relevance*)^[4]:

$$MMR = \arg \max_{D_i \in R \setminus S} \left[\lambda \left(Sim_1(D_1, Q) - (1 - \lambda) \max_{D_j \in S} Sim_2(D_i, D_j) \right) \right]$$

gdzie:

- D_i analizowany kandydat,
- $R \setminus S$ zbiór kandydatów, bez już wybranych,
- λ stała w zakresie $[0, 1]$, kontrolująca zróżnicowanie wyników (0 dla maksymalnie zdwywersyfikowanych, 1 dla listy odpowiadającej wzięciu wprost najlepszych wyników),
- S zbiór już wybranych kandydatów,
- Q zapytanie,
- Sim_1 metryka podobieństwa kandydata do zapytania,
- Sim_2 metryka podobieństwa kandydatów między sobą.

W kontekście poszukiwania najlepszej reprezentacji tematu, kandydatami są słowa z najwyższymi wynikami $c\text{-}TF\text{-}IDF$, a zapytanie to wektor całego tematu. Wektor tematu obliczany jest wybranym modelem SBERT kodującym dokument powstały z połączenia trzydziestu słów tematu o najwyższych wynikach $c\text{-}TF\text{-}IDF$. Natomiast jako obie metryki Sim_1 i Sim_2 wykorzystywane jest podobieństwo kosinusowe. Wartość parametru λ ustalono na 0 (maksymalna dywersyfikacja). W ten sposób otrzymuje się reprezentacje tematów dużo lepiej opisujące zbior, co jest szczególnie istotne w języku polskim zawierającym wiele odmian tych samych słów. Tabela 4.1 przedstawia reprezentacje tematów otrzymane po zastosowaniu MMR.

Tab. 4.1: Reprezentacje wybranych tematów: a) bez MMR, b) z MMR

	A	morskich statków morskiego statku morskiej
a)	B	lotnicze lotnictwa lotniczego lotnisk lotniczych
	C	celnego celnych towarów celne celny
	A	statków statkach marynarzy gospodarki morskiej administracji morskiej
b)	B	lotnicze lotnictwa lotnictwa cywilnego prawo lotnicze ruchu lotniczego
	C	celnego celnych prawo celne kodeks celny prawa celnego

Rozdział 5

Ocena skuteczności modeli

Analizowane algorytmy są nienadzorowane, co oznacza, że nie ma jednoznacznej metody oceny wyników. Jedyną metodą ewaluacji modeli tematycznych dającą pewność jest manualna weryfikacja poprzez np. ocenianie w trzystopniowej skali, które reprezentacje tematów są spójne, a które nie. Jest to jednak kosztowna metoda, dlatego tworzy się algorytmy dające dobre przybliżenie ocen człowieka. Ocena spójności tematów według takich algorytmów porównywana jest z ocenami człowieka poprzez współczynnik korelacji Pearsona.

Alternatywną metodą zaproponowaną w [5] jest podmiana słowa w reprezentacji tematu (ang. *word intrusion task*). W wygenerowanych zestawach słów podmieniane jest jedno słowo na losowe. Następnie uczestnicy badania muszą zidentyfikować podmienione słowo. Jeśli tematy są spójne, to niepasujące słowo powinno być łatwe do zidentyfikowania, natomiast w mniej spójnych tematach słowa będą sprawiały wrażenie bardziej losowych, co utrudni poprawne rozwiązanie problemu.

5.1. Ewaluacja spójności reprezentacji tematów

Najpopularniejszą metodą ewaluacji wygenerowanych tematów jest analiza reprezentacji tematów (zbioru słów opisujących temat). Im lepiej słowa opisujące temat reprezentują wszystkie dokumenty w danym temacie, tym lepszy model. W pracy [13] autorzy proponują algorytm (zwany dalej C_{UMass}) analizujący pary słów z reprezentacji tematu pod kątem współwystępowania w dokumentach korpusu. Jeśli para słów często występuje razem w dokumentach, to są one powiązane. Można więc określić spójność całej reprezentacji tematu badając wszystkie w niej zawarte pary słów. Miara ta porównana została z wynikami zadania podmiany słowa i uzyskała ona lepszą korelację z ocenami ludzi.

Autorzy [1] proponują rozwiązanie bazujące na wektorach kontekstowych słów tematu (zwane dalej C_{NPMI}). Wektory te są obliczane na podstawie współwystępujących słów w kontekście analizowanego słowa. Kontekst jest definiowany jako tzw. przesuwne okno (ang. *sliding window*) zwierające ± 5 otaczających słów. Następnie wektory są ważone algorytmem NPMI (ang. *Normalized Point Mutual Information*). Końcowa wartość obliczana jest jako średnia odległość kosinusowa między wszystkimi parami wektorów kontekstowych słów tematu.

W pracy [18] autorzy stworzyli bazową strukturę pozwalającą na zdefiniowanie metryk C_{UMass} , C_{NPMI} oraz innych miar spójności w jednej przestrzeni. W projekcie skorzystano z biblioteki Gensim¹, która implementuje miary spójności na podstawie tych definicji.

¹<https://radimrehurek.com/gensim/models/coherencemodel.html> (dostęp dnia 29 maja 2021)

5.2. Ewaluacja spójności rozkładu tematów

Ze względu na specyfikę korpusu możliwe jest również sformułowanie metryki wykorzystującej porządek obrad sejmu. Każde posiedzenie przebiega zgodnie z ustaloną strukturą, gdzie marszałek zarządza obrady na temat danej ustawy, poprawki czy interpelacji, a następnie kolejni posłowie wyrażają swoją opinię na ten temat. Można więc założyć, że w obrębie debaty nad jednym punktem obrad, wypowiedzi powinny zostać zakwalifikowane do tego samego zbioru tematycznego. Stworzono więc algorytm wykorzystujący tę zależność, przedstawiony na listingu 5.1. Implementacja algorytmu w języku Python pokazana jest na listingu 5.2. Założono, że w zdecydowanej większości przypadków, kolejne dwie wypowiedzi powinny mieć przypisany ten sam temat.

Algorytm analizuje osobno każdy dzień posiedzeń. Wypowiedzi sortowane są zgodnie z kolejnością wynikającą z transkrypcji przemówień. Jeśli w ciągu dnia jest mniej niż dwie wypowiedzi, to dzień jest pomijany. W zbiorze danych występuje jeden taki przypadek, natomiast mediana wynosi 138. Wśród wypowiedzi danego dnia porównywane są tematy par sąsiadujących wypowiedzi. Zliczane są zarówno znalezione pasujące pary, jak i różniące się. Jeśli temat analizowanej wypowiedzi jest nieznany (algorytm HDBSCAN nie przypisał dokumentu do żadnej grupy), to jest ona pomijana. Mechanizm ten zapobiega sytuacji, w której duża liczba nieprzypisanych dokumentów zaważyłaby wynik. Dzieje się tak ze względu na to, że nieprzypisane dokumenty często stanowią znaczną część zbioru (30–50%), przez co istnieje duże prawdopodobieństwo znalezienia par takich dokumentów. Dodatkowo, od sumy różniących się par odejmowana jest liczba o 1 mniejsza od liczby tematów danego dnia. Działanie to rekompensuje przejścia między tematami, które muszą wystąpić i nie są błędem. Takich przejść w ciągu dnia wystąpi co najmniej tyle, ile jest tematów minus jeden. Końcowy wynik obliczany jest jako stosunek zgadzających się par do wszystkich zliczonych par.

Listing 5.1: Algorytm obliczania spójności tematów

```

1 input: speeches sorted by original order of speaking , with date and assigned
      topic identifier
2 output: consistency score between 0 and 1 with higher score indicating more
      consistent topics
3
4 same_per_day := []
5 different_per_day := []
6
7 for each speeches in data grouped by date do
8   if len(speeches) < 2
9     continue
10
11  same := 0
12  different := 0
13  for i in range(len(speeches) - 1) do
14    if speeches[i].topic is unknown
15      continue
16
17    if speeches[i].topic == speeches[i + 1].topic
18      same := same + 1
19    else
20      different := different + 1
21
22  topic_count := len(speeches.topic.unique())
23  different := different - (topic_count - 1)
24
25  same_per_day.append(same)
26  different_per_day.append(different)
27
28 score = sum(same_per_day) / sum(same_per_day , different_per_day )
29 return score

```

5.3. Końcowa ocena

Powysze metody nie poddają ocenie samej liczby tematów i liczby nieprzyporządkowanych dokumentów. Z tego powodu, wysoki wynik mógłby osiągnąć model, który znalazłby zaledwie 5 tematów. Taki model przyporządkuje wszystkie dokumenty generując bardzo ogólne tematy. Podobnie wysoki wynik osiągnie model, który wygeneruje tysiące bardzo szczegółowych tematów. Takie modele można odfiltrować definiując odpowiednie progi, jednak taka metoda wymusza ustanowienie założeń, które niekoniecznie będą optymalne. Zamiast tego, stworzone zostały dwie dodatkowe miary: jedna ocenia liczbę przyporządkowanych dokumentów s_f , druga ocenia liczbę tematów s_T . Końcowa ocena modelu jest średnią ocen C_{UMass} , s_c , s_f , s_T przeskalowanych do pełnego zakresu (0, 1). Skalowanie odbywa się przy pomocy klasy MinMaxScaler z biblioteki scikit-learn, jak przedstawiono w linijce 18 listingu 5.3.

Pokazać
jakieś dane-
rysunek
albo tabelka

ustalić
końcowy
zestaw

5.3.1. Ocena liczby przyporządkowanych dokumentów

Miara oceniająca liczbę przyporządkowanych dokumentów korzysta bezpośrednio z wyniku grupowania. Grupa oznaczona jako '-1' oznacza dokumenty, które nie zostały zgrupowane, a więc nie mają przypisanego tematu. Rozmiar tej grupy odejmowany jest od rozmiaru korpusu uzyskując liczbę dokumentów z rozpoznanymi tematami. Liczba ta dzielona jest przez rozmiar korpusu, co daje miarę w przedziale (0, 1). Działanie to przedstawione jest w linijce 16 listingu 5.3, gdzie scores['not_found'] to kolumna tabeli zawierająca rozmiar grupy nieprzyporządkowanych dokumentów dla każdego modelu.

jakiś ry-
sunek czy
coś

5.3.2. Ocena liczby tematów

Dużo trudniej jest ocenić liczbę tematów, gdyż zarówno osiemdziesiąt ogólnych tematów, jak i tysiąc szczegółowych mogą zostać uznane za odpowiednie. Zważając jednak na specyfikę korpusu, przyjęto że różnorodność poruszanych kwestii powinna skutkować relatywnie dużą liczbą tematów. Z tego powodu uznano, że zbyt ogólne tematy nie są odpowiednie w tym kontekście. Nadmiernie szczegółowe, a więc o małej liczebności tematy również tracą informacje o zależnościach między dokumentami, więc także uznane zostały za niepożądane. W takim wypadku za odpowiednią liczbę tematów przyjęto medianę liczby tematów dla wszystkich testowanych modeli, która wyniosła 219.

W celu określenia wyniku każdego modelu wykorzystano zmodyfikowaną standaryzację Z. Standaryzacja Z obliczana jest zgodnie z następującym wzorem:

$$z = \frac{x - \mu}{\sigma}$$

gdzie:

- x wartość zmiennej,
- μ średnia z populacji,
- σ odchylenie standardowe populacji.

Średnia arytmetyczna i odchylenie standardowe są jednak podatne na wpływ wartości leżących daleko od średniej. Z tego powodu często wykorzystywana jest zmodyfikowana standaryzacja Z, która mierzy odległości od mediany:

$$\tilde{z} = \frac{x - \tilde{x}}{k \cdot MAD}$$

gdzie:

- x wartość zmiennej,

- \tilde{x} mediana z populacji,
- $k = 1.486$ wartość korekcyjna stanowiąca przybliżenie odchylenia standardowego,
- MAD średnie odchylenie bezwzględne — $median(|x_i - \tilde{x}|)$.

Tak ustardaryzowane dane rozłożone są wokół zera, gdzie liczby tematów mniejsze od mediana mają ujemny wynik, a większe dodatni. Takie odległości interpretowane są jako odległości od najlepszej liczby tematów, im większa odległość tym gorszy powinien być wynik takiego modelu. Aby wyrazić te odległości w skali $(0, 1)$, gdzie 1 to najlepszy wynik, znormalizowano dane poprzez odwrócenie wartości bezwzględnej standaryzacji Z powiększonej o 1 (aby minimalna wartość była równa 1):

$$s_T = \frac{1}{|\tilde{z}| + 1}$$

Implementacja tej miary przedstawiona została na listingu 5.3 jako funkcja modified_zscore. ↗ rys

5.3.3. Implementacja

Algorytmy zaimplementowane zostały w języku Python. Wykorzystane zostały biblioteki numpy, pandas, scikit-learn oraz scipy.

Listing 5.2: Funkcja obliczająca spójność s_c

```

1 | def score_consistency(df: pd.DataFrame) -> float:
2 |     same_per_day = []
3 |     different_per_day = []
4 |
5 |     for _, day in df.sort_values('speech_order').groupby(by='date'):
6 |         if len(day) < 2:
7 |             continue
8 |
9 |         same = 0
10 |         different = 0
11 |         for i in range(len(day) - 1):
12 |             if day.iloc[i].topic == -1:
13 |                 continue
14 |
15 |             if day.iloc[i].topic == day.iloc[i + 1].topic:
16 |                 same += 1
17 |             else:
18 |                 different += 1
19 |
20 |         topics = len(day.topic.unique())
21 |         different -= (topics - 1)
22 |
23 |         same_per_day.append(same)
24 |         different_per_day.append(different)
25 |
26 |     s_c = np.sum(same_per_day) / np.sum([*same_per_day, *different_per_day])
27 |
28 |     return s_c

```

Listing 5.3: Obliczanie finalnej oceny spójności tematów dla listy modeli

```

1 | from sklearn.preprocessing import MinMaxScaler
2 | from scipy import stats
3 | import numpy as np
4 |
5 | def modified_zscore(x: List[float]) -> List[float]:
6 |     median = np.median(x)
7 |     deviation_from_med = x - median

```

```

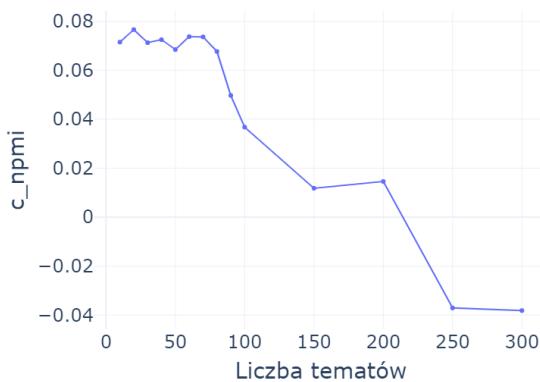
8     mad = np.median(np.abs(deviation_from_med))
9
10    consistency_correction = 1.4826
11    mod_zscore = deviation_from_med / (consistency_correction * mad)
12
13    return mod_zscore
14
15
16 scores['s_f'] = [(total - x) / total for x in scores['not_found']]
17
18 scores['s_T'] = 1 / (np.abs(modified_zscore(scores['topic_count'])) + 1)
19
20 scaler = MinMaxScaler()
21 minimax = scaler.fit_transform(scores[['s_c', 'u_mass', 's_f', 's_T']])
22
23 scores['avg'] = np.nanmean(minimax, axis=1)

```

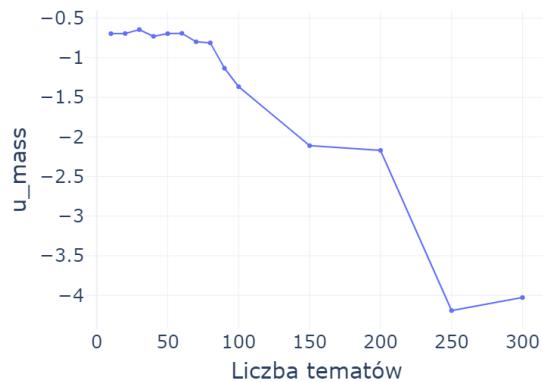
5.4. Porównanie z LDA

Tematy uzyskane metodą BERTopic zostały porównane z popularną metodą analizy tematycznej — LDA. W tym celu wykonano pomiary spójności algorytmami opisanymi w poprzednich rozdziałach. Jedną z zasadniczych wad algorytmu LDA jest fakt, iż liczba tematów jest jego parametrem. Aby znaleźć optymalną liczbę tematów, przeprowadzono pomiary dla różnych wartości parametru z zakresu [10–300]. Dokumenty przetworzono tą samą metodą, co w przypadku generowania wektorów TF-IDF w rozdziale 3.3.2, mianowicie lematyzacja słownikami *Morfeusz2*, usunięcie polskich słów bez znaczenia (ang. *stop words*) oraz wygenerowanie bigramów. Ze względu na ograniczenia techniczne związane z pamięcią operacyjną maszyny (128GB), analizie poddano sto tysięcy losowo wybranych dokumentów, co stanowi ok. 34% całości korpusu. Wyniki tych pomiarów przedstawione są na rysunku 5.1 oraz w tabeli 5.1. Z danych można odczytać, że dokładność modelu utrzymuje się na podobnym poziomie do ok. 80 tematów, potem zaczyna gwałtownie spadać. Za optymalną liczbę tematów wybrano więc 80 ze względu na zróżnicowanie korpusu, który z pewnością zawiera dużo tematów.

a)



b)

Rys. 5.1: Wyniki pomiarów dla LDA: a) C_{NPMI} , b) C_{UMass}

Dla znalezionych tematów wygenerowano reprezentacje składające się z pięciu najwyżej ocenianych słów. Losowo wybrane 5 tematów przedstawiono w tabeli 5.2a. Dodatkowo, ze względu na to, iż mediana liczby znalezionych tematów algorymem HDBSCAN wyniosła 219,

Tab. 5.1: Wyniki pomiarów dla LDA w zależności od liczby tematów

	c_npmi	u_mass
10	0.071499	-0.696868
20	0.076610	-0.694938
30	0.071255	-0.645520
40	0.072496	-0.730301
50	0.068460	-0.696550
60	0.073730	-0.692332
70	0.073619	-0.799463
80	0.067665	-0.813064
90	0.049716	-1.132477
100	0.036761	-1.366198
150	0.011804	-2.110305
200	0.014556	-2.170508
250	-0.037109	-4.191849
300	-0.038166	-4.027447

postanowiono zbadać również model LDA dla 200 tematów (tabela 5.2b). Dla obu tych modeli widoczny jest problem opisany w rozdziale 4. Mianowicie, reprezentacje tematów zawierają wiele podobnych słów specyficznych dla całego korpusu, a nie tematu („ustawa”, „minister”, czy „poseł”). Podjęto próby poszerzenia listy słów nieinformatywnych, jednak szybko okazało się, iż takich słów jest zbyt wiele — po usunięciu słów specyficznych dla korpusu nadal reprezentacje tematów składały się często ze słów typu „czas”, „kwestia”, „chodzi”. Nie stwierdzonoauważalnej różnicy między dokładnością reprezentacji wygenerowanych dla 80 tematów, a tymi dla 200.

Tab. 5.2: Reprezentacje pięciu losowo wybranych tematów dla LDA: a) 80 tematów, b) 200 tematów

A	rok mieć europejski minister r
B	państwo ustawa mieć rok Polska
a)	C minister państwo dziecko rok szkoła
	D minister poseł pytanie mieć chcieć
	E ustawa mieć poseł wysoki poprawka
A	ustawa projekt ustawy komisja wysoki
B	rok mieć minister poseł Maryja
b)	C ustawa zmiana projekt prawo ustawy
	D trybunał konstytucyjny ustawa rok Trybunału
	E ustawa mieć rok Pan sprawą

Rozdział 6

Aplikacja

Literatura

- [1] N. Aletras, M. Stevenson. Evaluating topic coherence using distributional semantics. *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, strony 13–22, Potsdam, Germany, Mar. 2013. Association for Computational Linguistics.
- [2] D. M. Blei, A. Y. Ng, M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, Mar. 2003.
- [3] R. J. G. B. Campello, D. Moulavi, A. Zimek, J. Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Trans. Knowl. Discov. Data*, 10(1), Lip. 2015.
- [4] J. Carbonell, J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’98, strona 335–336, New York, NY, USA, 1998. Association for Computing Machinery.
- [5] J. Chang, J. Boyd-Graber, S. Gerrish, C. Wang, D. M. Blei. Reading tea leaves: How humans interpret topic models. *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NIPS’09, strona 288–296, Red Hook, NY, USA, 2009. Curran Associates Inc.
- [6] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov. Unsupervised cross-lingual representation learning at scale, 2020.
- [7] S. Dadas, M. Perelkiewicz, R. Poswiata. Evaluation of sentence representations in polish. *CoRR*, abs/1910.11834, 2019.
- [8] J. Devlin, M. Chang, K. Lee, K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.
- [9] G. Erkan, D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22:457–479, Dec 2004.
- [10] M. Grootendorst. Bertopic: Leveraging bert and c-tf-idf to create easily interpretable topics., 2020.
- [11] L. McInnes, J. Healy, N. Saul, L. Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.
- [12] T. Mikolov, K. Chen, G. Corrado, J. Dean. Efficient estimation of word representations in vector space, 2013.
- [13] D. Mimno, H. M. Wallach, E. Talley, M. Leenders, A. McCallum. Optimizing semantic coherence in topic models. *Proceedings of the Conference on Empirical Methods in*

Natural Language Processing, EMNLP '11, strona 262–272, USA, 2011. Association for Computational Linguistics.

- [14] M. Ogrodniczuk. The Polish Sejm Corpus. *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012*, strony 2219–2223, Istanbul, Turkey, 2012. European Language Resources Association (ELRA).
- [15] M. Ogrodniczuk. Polish Parliamentary Corpus. D. Fišer, M. Eskevich, F. de Jong, redaktorzy, *Proceedings of the LREC 2018 Workshop ParlaCLARIN: Creating and Using Parliamentary Corpora*, strony 15–19, Paris, France, 2018. European Language Resources Association (ELRA).
- [16] N. Reimers, I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.
- [17] N. Reimers, I. Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2020.
- [18] M. Röder, A. Both, A. Hinneburg. Exploring the space of topic coherence measures. *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM '15*, strona 399–408, New York, NY, USA, 2015. Association for Computing Machinery.
- [19] P. Rybak, R. Mroczkowski, J. Tracz, I. Gawlik. KLEJ: comprehensive benchmark for polish language understanding. *CoRR*, abs/2005.00630, 2020.
- [20] R. Tuora, Łukasz Kobyliński. Integrating polish language tools and resources in spacy. *Proceedings of PP-RAI'2019 Conference*, Wrocław, Poland, 10 2019.
- [21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin. Attention is all you need. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett, redaktorzy, *Advances in Neural Information Processing Systems*, wolumen 30. Curran Associates, Inc., 2017.
- [22] T. Walkowiak, S. Datko, H. Maciejewski. Bag-of-words, bag-of-topics and word-to-vec based subject classification of text documents in polish - a comparative study. W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak, J. Kacprzyk, redaktorzy, *Contemporary Complex Systems and Their Dependability*, strony 526–535, Cham, 2019. Springer International Publishing.
- [23] M. Woliński. Morfeusz reloaded. N. Calzolari, K. Choukri, T. Declerck, H. Loftsson, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, S. Piperidis, redaktorzy, *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014*, strony 1106–1111, Reykjavík, Iceland, 2014. European Language Resources Association (ELRA).
- [24] Y. Yang, D. Cer, A. Ahmad, M. Guo, J. Law, N. Constant, G. H. Abrego, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, R. Kurzweil. Multilingual universal sentence encoder for semantic retrieval, 2019.

Dodatek A

Instrukcja wdrożeniowa

Jeśli praca skończyła się wykonaniem jakiegoś oprogramowania, to w dodatku powinna pojawić się instrukcja wdrożeniowa (o tym jak skompilować/zainstalować to oprogramowanie). Przydałoby się również krótkie how to (jak uruchomić system i coś w nim zrobić – zademonstrowane na jakimś najprostszym przypadku użycia). Można z tego zrobić osobny dodatek,

Dodatek B

Opis załączonej płyty CD/DVD

Tutaj jest miejsce na zamieszczenie opisu zawartości załączonej płyty. Należy wymienić, co zawiera.