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 W4/K9

WROCŁAW, 2021

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, consectetuer 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, consectetuer congue risus.

Słowa kluczowe: one, two, three, four

Spis treści

Spis rysunków

Spis tabel

Spis listingów

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 pozyskanymi 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[11], 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 [10, Ogrodniczuk, 2012], natomiast na potrzeby tej pracy przetworzone zostały jedynie następujące pliki:

- $\mathrm{header.xml}$ — metadane posiedzenia (data, lista mówców),
- $\mathrm{text\_structure.xml}$ — 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 nierzeczowych fragmentów.

Zdarza się, że w trakcie wypowiedzi występują wtrącenia innych osób. Takie komentarze są jednak odpowiednio oznakowane ($\#\mathrm{komentarz}$, czy $\#\mathrm{głosZSali}$). 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 Posłach[1]. 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 uwspólniono 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) połączonego 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

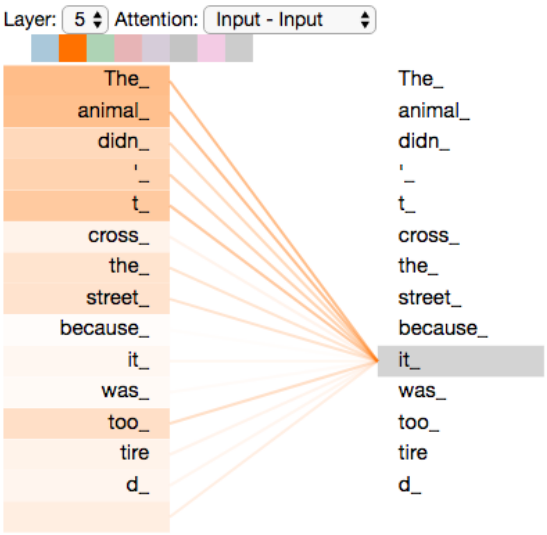
Reprezentacja wektorowa

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 semantyce dystrybucyjnej. 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*)[9], 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.

odnieść się do[16]

W 2018 r ukazała się praca prezentująca BERT (ang. *Bidirectional Encoder Representations from Transformers*)[4]. Był to pierwszy model kontekstowy. Oparty jest na transformerach[15] — 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 wagi 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 górne 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, w którym wypowiedzi często złożone są z kilkudziesięciu zdań. Problem ten opisywany jest szerzej w rozdziale 3.2.



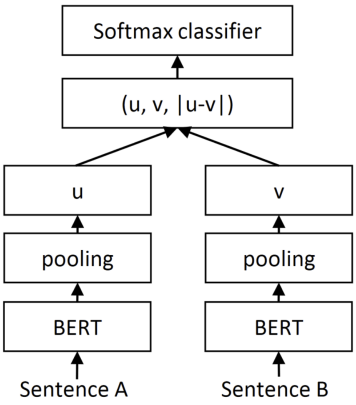
Rys. 3.1: Mechanizm atencji — wizualizacja wag poszczególnych słów podczas analizy słowa „it”. Źródło: https://jalammar.github.io/illustrated-transformer

3.1. Wektory zdań

Modele bazujące na BERT stworzone są z myślą generowania wektorów dla każdego tokena 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 tokena 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.

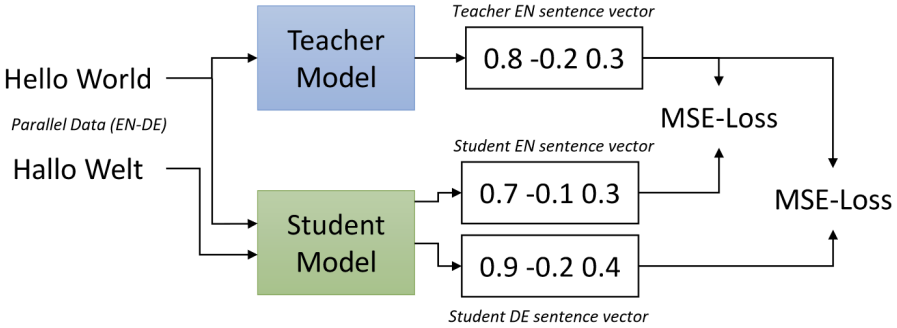
Rozwiązanie tego problemu proponują autorzy Sentence-BERT[12]. Model SBERT złożony jest z dwóch modli BERT połączonych w syjamską sieć. Na wyjściu każdego modelu BERT dołożona jest warstwa pooling zwracająca wektor ustalonego rozmiaru dla zdań różnych długości. Najlepszą strategią okazała się metoda uśredniania wektorów wszystkich tokenów. Następnie wyjściowe dwa wektory wynikowe są łączone i obliczana jest funkcja straty klasyfikatorem $\mathrm{softmax}$. 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 odległości 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 nauczonego modelu angielskiego na modele wielojęzykowe[13]. 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 architekturze RoBERTa (wariacja BERT stworzona przez Facebook), wytrenowany na 2.5TB danych z CommonCrawl w 100 językach, w tym polskim[2]. Model ten zajmuje obecnie ósme



Rys. 3.2: Architektura modelu Sentence-BERT[12]

miejsce w rankingu KLEJ[1] oraz drugie miejsce po dostrojeniu (ang. *fine-tuning*) na dodatkowych polskich korpusach.



Rys. 3.3: Układ modeli nauczyciel-uczeń w procesie uczenia wielojęzykowego SBERT[13]

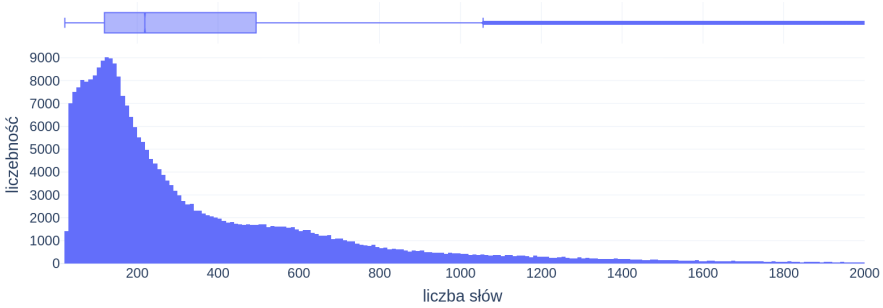
Trenowanie modelu ucznia wymaga par zdań w języku angielskim i języku docelowym. Jest to dużo mniejsze ograniczenie, niż w przypadku bezpośredniego uczenia SBERT w docelowym języku. Dostępne są wytrenowane modele[2], które obejmują również język polski. Zostały one uwzględnione w rankingu modeli generujących reprezentacje zdań w języku polskim[3]. 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 źródłowym[3]).

[1] https://klejbenchmark.com/leaderboard (dostęp dnia 20 maja 2021)
[2] https://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.



Min	21
Q1	119
Mediana	219
Q3	494
Górny wąs	1056
Max	28.546

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

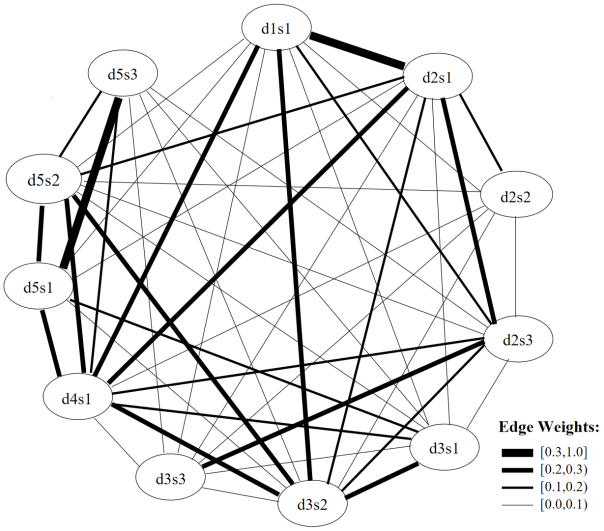
Do wygenerowania wag oraz rankingu potrzebna jest metoda oceniająca, na ile istotne jest każde zdanie w kontekście całej wypowiedzi. Stworzono w tym celu dwie metody: jedna wykorzystująca algorytm LexRank oraz druga wykorzystująca TF-IDF.

3.2.1. LexRank

LexRank[5] 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 wagi oryginalnym zdaniom.

Algorytm konstruuje graf, gdzie wierzchołki odpowiadają zdaniom, a ich wartość obliczana jest algorytmem TF-IDF. Krawędzie grafu odpowiadają wartości podobieństwa między dokumentami i mierzone są na miarą 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:

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



Rys. 3.5: Graf ważony miarą podobieństwa między zdaniami (wierzchołkami)[5]

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 częstość występowania danego słowa w dokumencie. IDF oznacza *inverse document-frequency*, zawiera wartości dla każdego słowa, informujące jak bardzo jest ono powszechne. 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 częstość występowania słowa we wszystkich dokumentach. Wartość $tfidf(t)$ 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 liczę 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

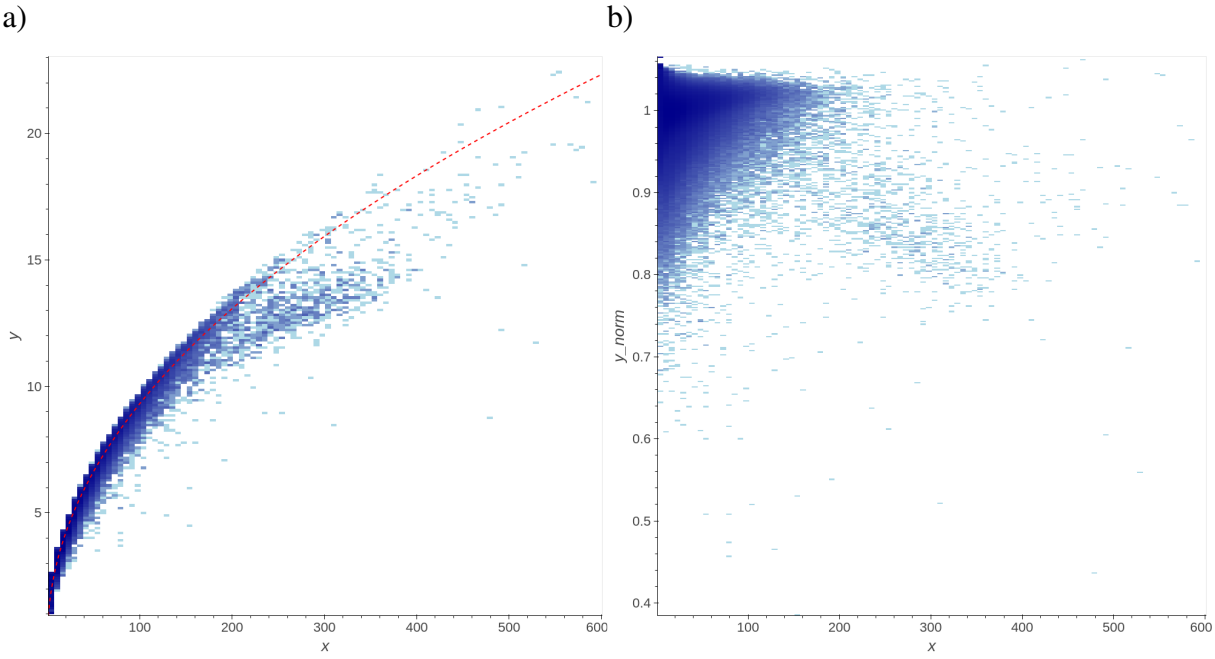
z mniej znaczących słów. Postanowiono więc zbadać, jak zmienia się suma wartości tf-idf w zależności od liczby wykrytych słów. Zależność ta przedstawiona została na wykresie 3.6a. Poprzez aproksymację punktową do danych dopasowana została funkcja potęgowa narysowana na wykresie kolorem czerwonym. Aproksymacji 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, 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)[18] 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[3]. 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 — $\mathrm{pl\_spacy\_model\_morfeusz}$[14] udostępniony przez Instytut Podstaw Informatyki Polskiej Akademii Nauk[4]. 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[17]. 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 $\mathrm{min\_df}$ równy 5 oznaczający, że dany token musi wystąpić w minimum pięciu dokumentach, aby był uwzględniony w wynikowej macierzy.

[4] $\mathrm{http://zil.ipipan.waw.pl/SpacyPL}$ (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*)[1]. 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 słów. Często największe prawdopodobieństwo wystąpienia mają powszechne słowa nie niosące istotnej informacji, np. zaimki czy łączniki. Wymagana jest lista nieinformatywnych 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*). Taki format nie zachowuje informacji o kolejności słów, ani ich semantyce. Celem LDA jest znalezienie takich tematów, które mogą posłużyć do odtworzenia oryginalnych rozkładów słów z dokumentów z minimalnym błędem. Model nie rozróżnia pomiędzy słowami informatywnymi, a nieinformatywnymi, ma za zadanie jedynie jak najlepiej odzwierciedlać oryginalne dokumenty. Z tego powodu najbardziej prawdopodobne słowa w temacie niekoniecznie odzwierciedlają faktyczne tematy dokumentów.

LDA dla pl[16]

Modele oparte o transformery osiągają najlepsze wyniki w wielu zagadnieniach z dziedziny NLP, więc powstały również bazujące na nich metody analizy tematycznej. Jedną z nich jest BERTopic[6], algorytm generujący tematy na podstawie wektorów BERT. Wykorzystuje UMAP[8] w celu redukcji wymiarowości wektorów do 5 wymiarów. Następnie za pomocą algorytmu HDBSCAN[7] 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[7]. Z tego powodu konieczna jest redukcja wymiaru danych.

Wykorzystano w tym celu algorytm UMAP (ang. *Uniform Manifold Approximation and Projection*)[8]. 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 $\mathrm{n\_neighbors}$. 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 $\mathrm{n\_neighbors}$ powstało wiele lokalnych grup o niskiej liczebności. Wraz ze wzrostem wartości parametru lokalne grupy łączą się w coraz większe klastry.

a) b) c)

n_neighbors = 5 n_neighbors = 15 n_neighbors = 100

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

4.2. Grupowanie

Rozdział 5

Analiza skuteczności modeli

Rozdział 6

Podsumowanie

Literatura

[1] D. M. Blei, A. Y. Ng, M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, Mar. 2003.

[2] 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.

[3] S. Dadas, M. Perelkiewicz, R. Poswiata. Evaluation of sentence representations in polish. *CoRR*, abs/1910.11834, 2019.

[4] J. Devlin, M. Chang, K. Lee, K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[5] 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.

[6] M. Grootendorst. Bertopic: Leveraging bert and c-tf-idf to create easily interpretable topics., 2020.

[7] L. McInnes, J. Healy, S. Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11):205, 2017.

[8] L. McInnes, J. Healy, N. Saul, L. Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[9] T. Mikolov, K. Chen, G. Corrado, J. Dean. Efficient estimation of word representations in vector space, 2013.

[10] 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).

[11] 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).

[12] 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.

[13] 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.

[14] R. Tuora, Łukasz Kobyliński. Integrating polish language tools and resources in spacy. *Proceedings of PP-RAI'2019 Conference*, Wrocław, Poland, 10 2019.

[15] 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.

[16] 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.

[17] 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).

[18] 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.