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[7], 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 [6, Ogrodniczuk, 2012], 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 nierzeczowych 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 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)

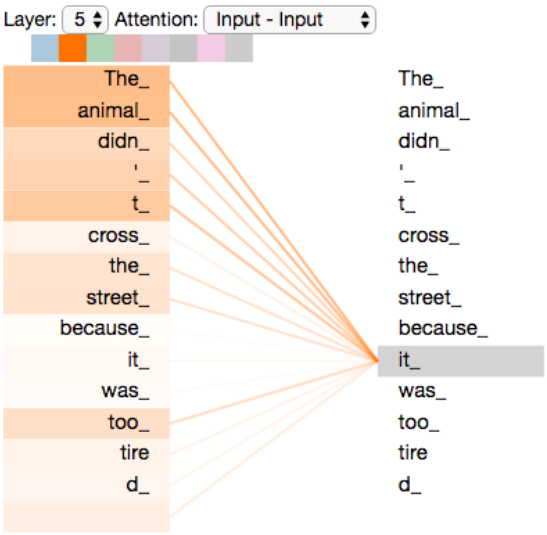
Rozdział 3

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*)[5], 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.

W 2018 r ukazała się praca prezentująca BERT (ang. *Bidirectional Encoder Representations from Transformers*)[3]. Był to pierwszy model kontekstowy. Oparty jest na transformerach[11] — 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.



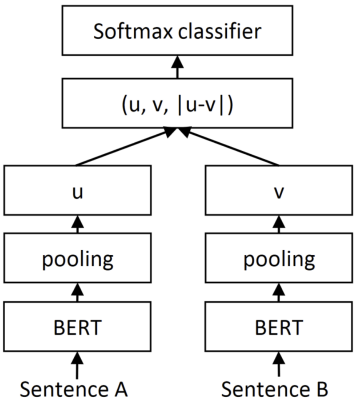
Rys. 3.1: Mechanizm atencji — wizualizacja wag poszczególnych słów podczas analizy słowa „it”. Źródło: https://jalammar.github.io/illustrated-transformer

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.

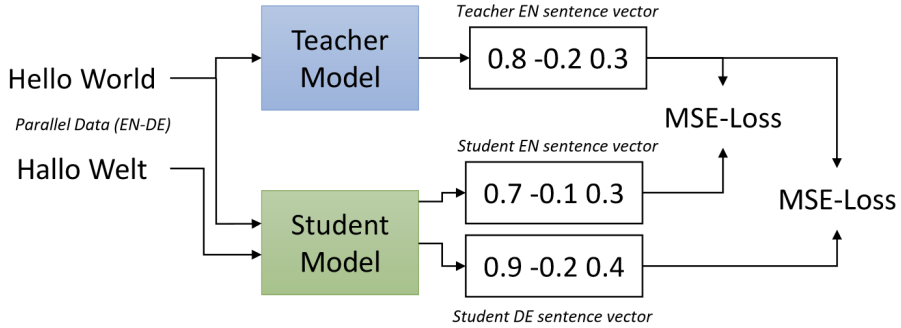
Rozwiązanie tego problemu proponują autorzy Sentence-BERT[8]. Model SBERT złożony jest z dwóch modli BERT połączonych w syjamską sieć. Na wyjściu każdego modelu BERT dołożona jest warstwa pooling zwracająca wektor ustalonego rozmiaru dla zdań różnych długości. Najlepszą strategią okazała się metoda uśredniania 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[9]. 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[1]. Model ten zajmuje obecnie ósme



Rys. 3.2: Architektura modelu Sentence-BERT[8]

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[9]

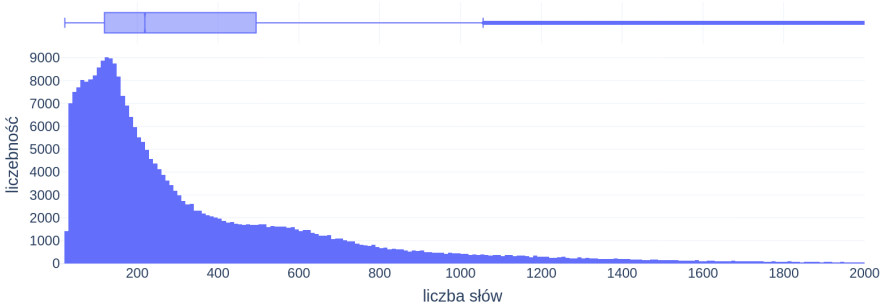
Trenowanie modelu ucznia wymaga par zdań w języku angielskim i języku docelowym. Jest to dużo mniejsze ograniczenie, niż w przypadku bezpośredniego uczenia SBERT w docelowym języku. Dostępne są wytrenowane modele[2], które obejmują również język polski. Zostały one uwzględnione w rankingu modeli generujących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

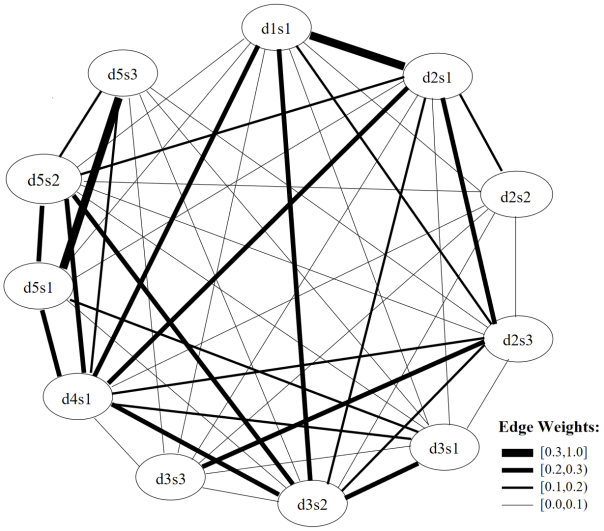
Do wygenerowania wag oraz rankingu potrzebna jest metoda oceniająca, na ile istotne jest każde zdanie w kontekście całej wypowiedzi. Stworzono w tym celu dwie metody: jedna wykorzystująca algorytm LexRank oraz druga wykorzystująca TF-IDF.

3.2.1. LexRank

LexRank[4] 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)[4]

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 wykładnicza narysowana na wykresie kolorem czerwonym. Aproksymacji poddane zostały trzy współczynniki według następującego wzoru funkcji wykładniczej:

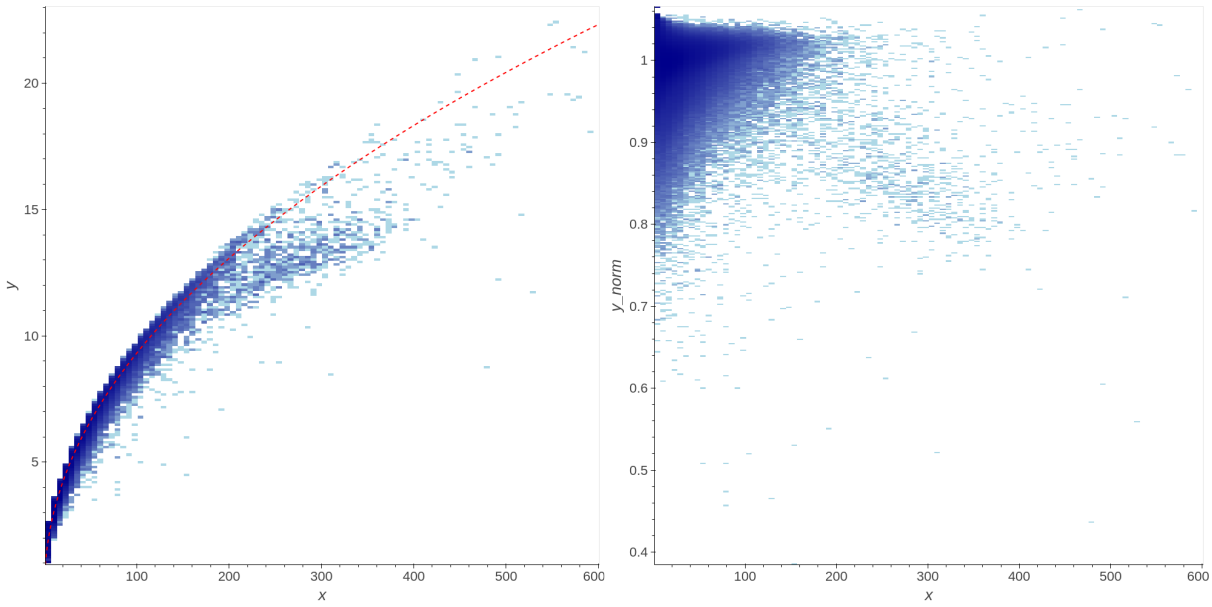
$$a \cdot x^b + c$$

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.

a) b)



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*)[13] 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[2]. 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 — $\mathrm{pl\_spacy\_model\_morfeusz}$[10] 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[12]. 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 minimum 5 razy, aby był uwzględniony w wynikowej macierzy.

[4] $\mathrm{http://zil.ipipan.waw.pl/SpacyPL}$ (dostęp dnia 21 maja 2021)

Rozdział 4

Redakcja pracy

4.1. Układ pracy

Istnieje wiele sposobów organizacji poszczególnych rozdziałów pracy. W dokumentacji klasy memoir (http://tug.ctan.org/tex-archive/macros/latex/contrib/memoir/memman.pdf) można np. znaleźć zalecenia co do edytorskiej strony amerykańskich prac dyplomowych. Na stronie 363 można przeczytać, że część początkowa pracy może zawierać:

1. Title page
2. Approval page
3. Abstract
4. Dedication (optional)
5. Acknowledgements (optional)
6. Table of contents
7. List of tables (if there are any tables)
8. List of figures (if there are any figures)
9. Other lists (e.g., nomenclature, definitions, glossary of terms, etc.)
10. Preface (optional but must be less than ten pages)
11. Under special circumstances further sections may be allowed

A dalej kolejne wymagania, w tym informacje o zawartości części końcowej:

1. Notes (if you are using endnotes and grouping them at the end)
2. References (AKA 'Bibliography' or 'Works Cited')
3. Appendices
4. Biographical sketch (optional)

Prace dyplomowe powstające na Wydziale Elektroniki Politechniki Wrocławskiej standardowo redaguje się w następującym układzie:

Strona tytułowa
Streszczenie
Strona z dedykacją (opcjonalna)
Spis treści
Spis rysunków (opcjonalny)
Spis tabel (opcjonalny)
Spis listingów (opcjonalny)
Skróty (wykaz opcjonalny)
Abstrakt
1. Wstęp
  1.1 Wprowadzenie
  1.2 Cel i zakres pracy
  1.3 Układ pracy
2. Kolejny rozdział
  2.1 Sekcja
    2.1.1 Podsekcja
      Nienumerowana podpodsekcja
        Paragraf
...
#. Podsumownie i wnioski
Literatura
A. Dodatek
  A.1 Sekcja w dodatku
...
$. Instrukcja wdrożeniowa
$. Zawartość płyty CD/DVD
Indeks rzeczowy (opcjonalny)

Streszczenie – syntetyczny opis tego, o czym jest i co zawiera praca, zwykle jednostronicowy, po polsku i po angielsku (podczas wysyłania pracy do analizy antyplagiatowej należy skopiować tekst z tej sekcji do formularza w systemie ASAP).

Spis treści – powinien być generowany automatycznie, z podaniem tytułów i numerów stron. Typ czcionki oraz wielkość liter spisu treści powinny być takie same jak w niniejszym wzorcu.

Spis rysunków, Spis tabel – powinny być generowane automatycznie (podobnie jak Spis treści). Elementy te są opcjonalne (robienie osobnego spisu, w którym na przykład są tylko dwie pozycje specjalnie nie ma sensu).

Skróty – jeśli w pracy występują liczne skróty, to w tej części należy podać ich rozwinięcia.

Wstęp – pierwszy rozdział, w którym powinien znaleźć się opis dziedziny, w jakiej osadzona jest praca, oraz wyjaśnienie motywacji do podjęcia tematu (na to przeznaczona jest sekcja „Wprowadzenie"). W sekcji „Cel i zakres" powinien znaleźć się opis celu oraz zadań do wykonania, zaś w sekcji „Układ pracy" – opis zawartości kolejnych rozdziałów.

Podsumowanie – w rozdziale tym powinny być zamieszczone: podsumowanie uzyskanych efektów oraz wnioski końcowe wynikające z realizacji celu pracy dyplomowej.

Literatura – wykaz źródeł wykorzystanych w pracy (do każdego źródła musi istnieć odpowiednie cytowanie w tekście). Wykaz ten powinien być generowany automatycznie.

Dodatki – miejsce na zamieszczanie informacji dodatkowych, jak: Instrukcja wdrożeniowa, Instrukcja uruchomieniowa, Podręcznik użytkownika itp. Osobny dodatek powinien być przeznaczony na opis zawartości dołączonej płyty CD/DVD. Założono, że będzie to zawsze ostatni dodatek.

Indeks rzeczowy – miejsce na zamieszczenie kluczowych wyrazów, do których czytelnik będzie chciał sięgnąć. Indeks powinien być generowany automatycznie. Jego załączanie jest opcjonalne.

4.2. Styl

Zasady pisania pracy (przy okazji można tu zaobserwować efekt wyrównania wpisów występujących na liście wyliczeniowej uzależnione od długości etykiety):

1. Praca dyplomowa powinna być napisana w formie bezosobowej („w pracy pokazano ...”). Taki styl przyjęto na uczelniach w naszym kraju, choć w krajach anglosaskich preferuje się redagowanie treści w pierwszej osobie.
2. W tekście pracy można odwołać się do myśli autora, ale nie w pierwszej osobie, tylko poprzez wyrażenia typu: „autor wykazał, że ...”.
3. Odwołując się do rysunków i tabel należy używać zwrotów typu: „na rysunku pokazano ...”, „w tabeli zamieszczono ...” (tabela i rysunek to twory nieżywotne, więc „rysunek pokazuje” jest niepoprawnym zwrotem).
4. Praca powinna być napisana językiem formalnym, bez wyrażeń żargonowych („sejwowanie” i „downloadowanie”), nieformalnych czy zbyt ozdobnych („najznamienitszym przykładem tego niebywałego postępu ...”)
5. Pisząc pracę należy dbać o poprawność stylistyczną wypowiedzi
   - trzeba pamiętać, do czego stosuje się „liczba”, a do czego „ilość”,
   - nie „szereg funkcji” tylko „wiele funkcji”,
   - redagowane zdania nie powinny być zbyt długie (lepiej podzielić zdanie wielokrotnie złożone na pojedyncze zdania),
   - itp.
6. Zawartość rozdziałów powinna być dobrze wyważona. Nie wolno więc generować sekcji i podsekcji, które mają zbyt mało tekstu lub znacząco różnią się objętością. Zbyt krótkie podrozdziały można zaobserwować w przykładowym rozdziale 6.
7. Niedopuszczalne jest pozostawienie w pracy błędów ortograficznych czy tzw. literówek – można je przecież znaleźć i skorygować automatycznie.

10005. Niedopuszczalne jest pozostawienie w pracy błędów ortograficznych czy tzw. literówek – można je przecież znaleźć i skorygować automatycznie.

Rozdział 5

Uwagi techniczne

5.1. Zasady redakcji

W pracy należy dbać o poprawność redakcyjną zgodnie z zaleceniami:

- nie zostawiać znaku spacji przed znakami interpunkcji („powiedziano , że ..." -> „powiedziano, że ..."),
- kropki po skrótach, które nie są jednocześnie kropkami kończącymi zdanie sklejać z kolejnym wyrazem znakiem tyldy, np. jak tutaj (np.~jak tutaj) lub wstawiać za nimi ukośnik, np. jak tutaj (np.\ jak tutaj)
- nie zapominać o dobrym sformatowaniu wyliczenia (należy zaczynać małymi literami lub dużymi oraz kończyć przecinkami, średnikami i kropkami – w zależności od kontekstu danego wyliczenia),
- nie zostawiać samotnych literek na końcach linii (można je „skleić" z wyrazem następnym stosując znaczek tyldy, jak w~przykładzie).
- nie zostawiać pojedynczych wierszy na końcu lub początku strony (należy kontrolować „sieroty" i „wdowy"),
- nie zostawiać odstępu pomiędzy tekstem a nawiasami czy znakami cudzysłowów (znaki te powinny przylegać do tekstu, który obejmują „jak w tym przykładzie"),
- wyrazy obcojęzyczne powinny być pisane czcionką pochyłą (preferowane \emph{}) wraz ze skrótem oznaczającym język, w szczególności ma to zastosowanie przy rozwijaniu skrótów, np. OGC (ang. *Open Geospatial Consortium*) (w kodzie latexowym wygląda to tak: np.~OGC (ang.~\emph{Open Geospatial Consortium})),
- każdy zastosowany skrót powinien zostać rozwinięty podczas pierwszego użycia, później może już występować bez rozwinięcia (skrót i jego rozwinięcie powinny trafić również do wykazu Skróty, jeśli taki wykaz jest dołączany do dokumentu).
- nie wolno zostawiać zbyt dużo białej przestrzeni bez żadnego uzasadnienia.

Odnosząc się do ostatniej zasady, to przekłada się ona na gospodarowanie białą przestrzenią. Poniżej przedstawiono przykład złego użycia listy wyliczeniowej. Jeśli bowiem elementy na liście są reprezentowane przez krótki tekst, to wtedy powstaje biała plama. Widać to szczególnie przy długich listach.

Moje ulubione kolory to:

- biały,
- niebieski,
- czerwony.

Dużo lepiej w takim przypadku albo po prostu wyliczyć wartości po przecinkach:

Moje ulubione kolory to: biały, niebieski, czerwony.

albo wypisać listę w kilku kolumnach:

Moje ulubione kolory to:

- biały,
- niebieski,
- czerwony.

5.2. Rysunki

W niniejszym szablonie numeracja rysunków odbywa się automatycznie według następujących reguł: rysunki powinny mieć numerację ciągłą w obrębie danego rozdziału, sam zaś numer powinien składać się z dwóch liczb rozdzielonych kropką. Pierwsza liczbą ma być numer rozdziału, drugą – kolejny numer rysunku w rozdziale. Przykładowo: pierwszy rysunek w rozdziale 1 powinien mieć numer 1.1, drugi – numer 1.2 itd., pierwszy rysunek w rozdziale 2 powinien mieć numer 2.1, drugi – numer 1.2 itd. Wszystko to załatwia szablon.

Rysunki powinny być wyśrodkowane na stronie wraz z podpisem umieszczonym na dole. Podpisy nie powinny kończyć się kropką. Czcionka podpisu powinna być mniejsza od czcionki tekstu wiodącego o 1 lub 2 pkt (w szablonie jest to czcionka rozmiaru small). Ponadto należy zachowywać odpowiedni odstęp między rysunkiem, podpisem rysunku a tekstem rozdziału. Przykłady, jak to zrobić, zamieszczono w szablonie.

W przypadku korzystania z szablon odstępy te regulowane są automatycznie. Podpis i grafika muszą stanowić jeden obiekt. Chodzi o to, że w edytorach tekstu typu Office podpis nie scala się z grafiką i czasem trafia na następną stronę, osieracając grafikę. Korzystającym z niniejszego szablonu i otoczenia \figure takie osierocenie nigdy się nie zdarzy.

Do każdego rysunku musi istnieć odwołanie w tekście (inaczej mówiąc: niedopuszczalne jest wstawienie do pracy rysunku bez opisu). Odwołania do rysunków powinny mieć postać: „Na rysunku 3.3 przedstawiono...” lub „... co ujęto na odpowiednim schemacie (rys. 1.7)”.

Jeśli odwołanie stanowi część zdania, to wtedy wyraz „rysunek” powinien pojawić się w całości. Jeśli zaś odwołanie jest ujęte w nawias (jak w przykładzie), wtedy należy zastosować skrót „rys.”. Jeśli do stworzenia obrazka wykorzystano jakieś źródła, to powinny one być zacytowane w podpisie tegoż rysunku.

Należy pamiętać o tym, że „rysunki” to twory nieżywotne. W związku z tym nie mogą ”pokazywać”. Dlatego „rysunek 1.1 pokazuje ...” jest stylistycznie niepoprawne. Zamiast tego zwrotu trzeba użyć „ na rysunku 1.1 pokazano ...”.

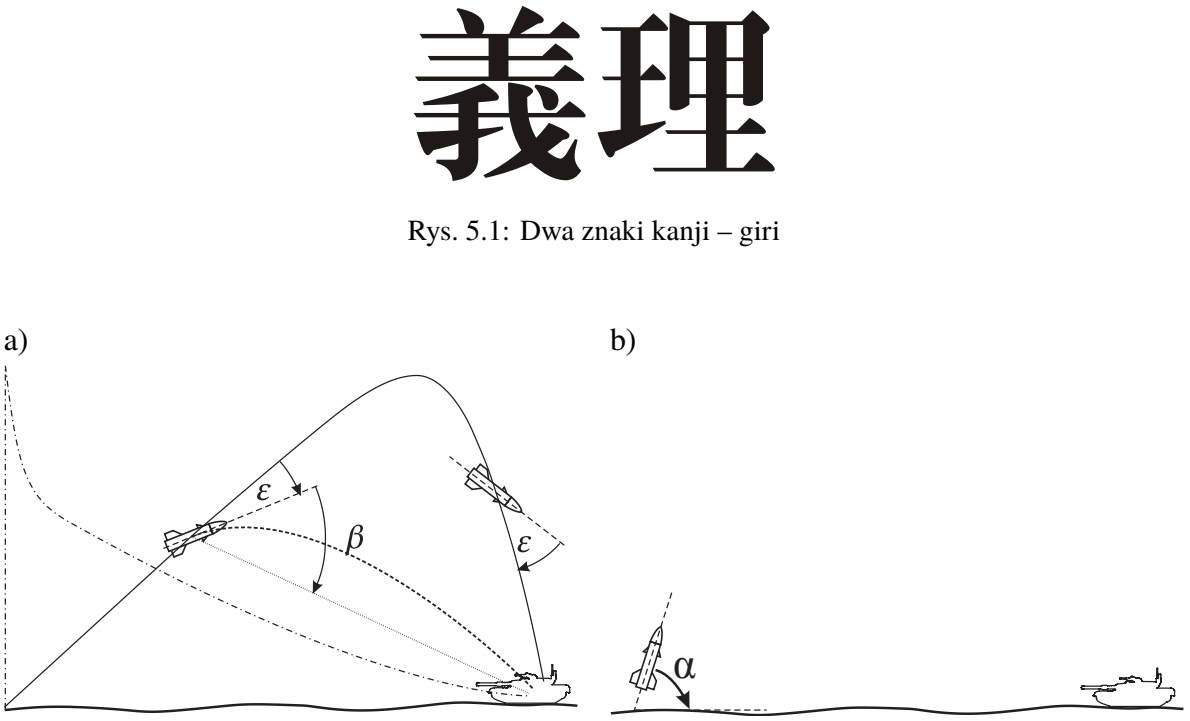
Rysunki można wstawiać do pracy używając polecenia \includegraphics. Zalecane jest, aby pliki z grafikami były umieszczane w katalogach odpowiadających numerom rozdziałów czy literom dodatków: rys01, rysA itd. Sposób wstawiania rysunków do pracy zademonstrowano na przykładze rysunków 5.1 i 5.2.

Listing 5.1: Kod źródłowy przykładów wstawiania rysunków do pracy

```
\begin{figure}[ht]
 \centering
  \includegraphics[width=0.3\linewidth]{rys05/kanji-giri}
 \caption{Dwa znaki kanji - giri}
 \label{fig:kanji-giri}
\end{figure}

\begin{figure}[htb]
 \centering
  \begin{tabular}{@{}ll@{}}
  a) & b) \\
  \includegraphics[width=0.475\textwidth]{rys05/alfa1} &
```

```
  \includegraphics[width=0.475\textwidth]{rys05/beta1}
        % jeśli obraki są różnej wysokości, można je wyrównać do góry
            ↪ stosując vtop jak niżej
        % \vtop{\vskip-2ex\hbox{{\includegraphics[width=0.475\textwidth]{
            ↪ rys05/beta1}}}} &
        % \vtop{\vskip-2ex\hbox{{\includegraphics[width=0.475\textwidth]{
            ↪ rys05/alfa1}}}}
  \end{tabular}
 a) trzy podejścia, b) podejście praktyczne}
 \label{fig:alfabeta}
\end{figure}
```



Rys. 5.1: Dwa znaki kanji – giri

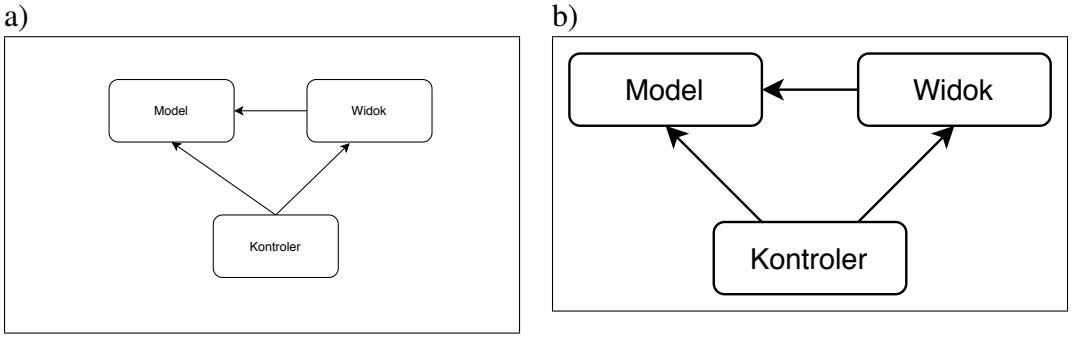
Rys. 5.2: Wyznaczanie trajektorii lotu rakiety: a) trzy podejścia, b) podejście praktyczne

Grafiki wektorowe powinny być dostarczone w plikach o formacie pdf. Rozmiar strony w pliku pdf powinien być troszeczkę większy niż zamieszczona na nim grafika (proszę spojrzeć na przykłady grafik wykorzystanych w niniejszym szablonie). Chodzi o to, aby na rysunku nie pojawiała się niepotrzebna biała przestrzeń (rozmiar płótna ma odpowiadać rozmiarowi grafiki bez żadnych marginesów, elementy grafiki powinny być ciasno ułożone). Grafiki rastrowe (głównie zrzuty z ekranu bądź zdjęcia) powinny być dostarczane w plikach o formacie png z kompresją bezstratną. Zastosowanie kompresji stratnej, jak jpg, wprowadza niepotrzebne artefakty. Podobnie jak w przypadku grafik wektorowych, grafiki rastrowe nie powinny mieć białych marginesów.

Niezłą ścieżką zapewniającą wygenerowanie ładnej grafiki wektorowej, w szczególności diagramów, jest:

- w diagrams.net (dawniej draw.io): narysowanie diagramu i wyeksportowanie do pdf
- w inkscape: zaimportowanie pdf, rozdzielenie grupy, wykasowanie niepotrzebnych elementów (tła), zaznaczenie wszystkiego, przycięcie strony do zaznaczonych (Ctr-Shift-R), zapisanie jako pdf.

Na rysunku 5.3 pokazano przykład dobrze i źle (od strony technicznej) narysowanego diagramu. Na rysunku celowo pokazano ramki, by było widać marginesy. Normalnie ramek tych nie należy stosować.



Rys. 5.3: Przykład diagramu: a) złego, b) w miarę dobrego

Na rysunkach nie powinno stosować się 100% czarnego wypełnienia, bo robią się plamy przebijające się przez kartkę. Zamiast tego wypełnienie powinno być ok. 90% czerni.

Czcionka na rysunkach nie może być większa od czcionki wiodącej tekstu (jedyny wyjątek to np. jakieś nagłówki). Należy stosować czcionkę kroju Arial, Helvetica bądź tego samego kroju co czcionka dokumentu (texgyre-termes).

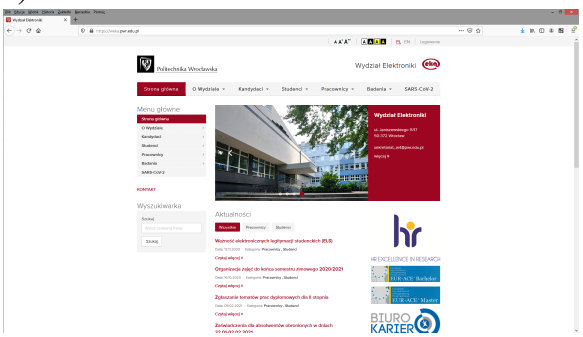
Jeśli na jednym rysunku pojawić się ma kilka grafik, to zamiast stosować subfigure lub inne otoczenia należy wstawić grafiki w tabelę, opisać ją indeksami a) i b), a potem odnieść się do tego w podpisie (rys. 5.2). Czasem pomaga w pozycjonowaniu rysunków użycie komendy: \vtop{\vskip-2ex\hbox{\includegraphics[width=0.475\textwidth]{nazwa}}}

Na rysunkach nie wolno nadużywać kolorów oraz ozdobników (wiele narzędzi do tworzenia diagramów dostarcza grafikę z cieniowaniem, gradacją kolorów itp. co niekoniecznie przekłada się na czytelność rysunku). Jeśli rysunki są kolorowe, to kolory te powinny być rozróżnialne po konwersji do poziomów szarości (chodzi o to, aby na wydrukach wykonanych na drukarkach monochromatycznych można było dostrzec różnice).

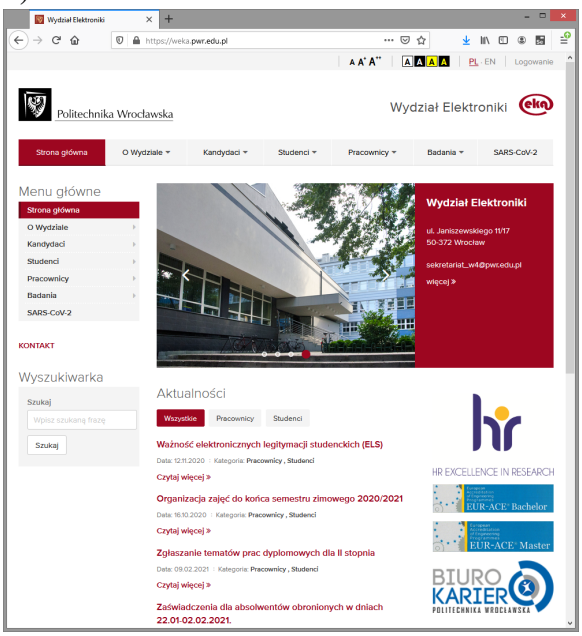
Podczas robienia zrzutów z ekranu należy zadbać o to, by taki zrzut był czytelny po wydrukowaniu. Czyli aby pojawiające się literki były wystarczająco duże, a przestrzenie bez treści – relatywnie małe. Przystępując do robienia zrzutu trzeba odpowiednio wyskalować elementy na ekranie. Na przykład robiąc zrzut z przeglądarki FF najpierw należy wcisnąć CTR–0 (domyślne skalowanie), potem CTR–- (zmniejszenie skali o stopień). Potem dobrze jest zawęzić okno przeglądarki tak, by interesująca treść wypełniła je w całości. Jeśli na obserwowanej stronie jest zbyt dużo pustych obszarów, to należy je jakoś zawęzić (sterując wielkością okna przeglądarki lub aktywnymi elementami interfejsu użytkownika). Zrzut bowiem wcale nie musi być odzwierciedleniem 1:1 domyślnego układu obserwowanych elementów. Ważne jest, by na zrzucie pokazać interesujący, opisywany fragment i żeby ten fragment był czytelny. Nie trzeba też zawsze robić zrzutów w układzie 16:9 (lub innym panoramicznym). Czasem lepiej jest zrobić zrzuty okien niemal kwadratowych, bo lepiej się układają w wynikowym dokumencie. Poza tym można je przeskalować (powiększyć) by zajmowały całą szerokość strony, a wtedy czcionka na wydrukach będzie większa. Takie skalowanie dla zrzutów panoramicznych zwykle się nie udaje. Na rysunku 5.4 pokazano przykłady dobrze i źle zrobionych zrzutów (w celu oszczędzenia miejsca zrzuty umieszczono obok siebie).

Czasem problemem jest tworzenie zrzutów z ekranu, gdy występują na nim dane wrażliwe. Istnieją dwa sposoby na radzenie sobie z tym problemem. Pierwszy polega na zastąpieniu w systemie danych danych rzeczywistych danymi testowymi – wygenerowanymi tylko do celów prezentacji. Zrzut robi się wtedy na bazie danych testowych. Drugi polega na wykonaniu zrzutu z ekranu, na którym pokazano dane rzeczywiste, i następnie zamianie tych danych już w pliku graficznym za pomocą odpowiedniego edytora (np. gimp). Czyli oryginalny zrzut z ekranu należy otworzyć w edytorze, a potem nadpisać oryginalny tekst własnym tekstem. Konieczne jest wtedy dobranie odpowiednich czcionek aby nie było widać wprowadzonych zmian.

a)



b)



Rys. 5.4: Przykłady zrzutów z ekranu: a) zły (nieczytelny, zrobiony przy zbyt szerokim oknie, z niepotrzebnymi marginesami, niepotrzebnym paskiem menu), b) w miarę dobry (w miarę czytelny, zrobiony przy zawężonym oknie, byłoby wskazane jeszcze usunięcie z niego beleczki z faviconem (jeśli nic nie wnosi) oraz przycięcie od dołu (jeśli treści tam pokazywane nie są istotne))

> Uwaga: takie manipulowanie zrzutami jest usprawiedliwione jedynie w przypadku konieczności ochrony danych wrażliwych czy też lepszego pokazania wybranych elementów. Nie może to prowadzić generowania fałszywych rezultatów!!!

5.3. Wstawianie kodu źródłowego

Kod źródłowy można wstawiać jako blok tekstu pisany czcionką maszynową. Używa się do tego otoczenie \lstlisting. W atrybutach otoczenia można zdefiniować tekst podpisu wstawianego wraz z numerem nad blokiem, etykietę do tworzenia odwołań, sposób formatowania i inne ustawienia. Zaleca się stosowanie w tym otoczeniu następujących parametrów:

```
\begin{lstlisting}[label=list:req1,caption=Initial HTTP Request,
                   basicstyle=\footnotesize\ttfamily]
```

Szczególnie przydatne podczas wstawiania większej ilości kodu źródłowego jest zastosowanie parametru basicstyle=\footnotesize\ttfamily. Dzięki niemu zmniejsza się czcionka, a przez to na stronie można zmieścić dłuższe linijki kodu. Użycie tak zdefiniowanego parametru nie jest jednak sztywnym zaleceniem. Wielkość czcionki można dobierać do potrzeb.

Listing 5.2: Initial HTTP Request

```
GET /script/Articles/Latest.aspx HTTP/1.1
Host: www.codeproject.com
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml
User-Agent: Mozilla/5.0 ...
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US...
Accept-Charset: windows-1251,utf-8...
```

Można też sformatować kod bez stosowania numerowanego podpisu (wtedy nie zamieszcza się caption na liście atrybutów).

```
GET /script/Articles/Latest.aspx HTTP/1.1
Host: www.codeproject.com
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml
User-Agent: Mozilla/5.0 ...
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US...
Accept-Charset: windows-1251,utf-8...
```

Istnieje możliwość wstawiania kodu źródłowego w bieżącej linijce tekstu. Można to zrobić na kilka sposobów:

- korzystając z polecenia \texttt ustawiającego czcionkę maszynową, jak w przykładzie tutaj (efekt zastosowania komendy \texttt{tutaj}). Problemem jednak mogą okazać się znaki podkreślenia i inne znaki kontrolne.
- korzystają z otoczenia \verb zapewniającego wypisanie kodu czcionką maszynową jak w przykładzie tutaj (efekt zastosowania komendy \verb|tutaj|). Problemem jest to, że polecenie \verb nie potrafi łamać dłuższego tekstu.
- korzystając z polecenia \lstin umożliwiającego wypisanie kodu czcionką ustawianą w opcjach jak w przykładzie tutaj (efekt komendy \lstset{basicstyle=\ttfamily}\lstinline{tutaj}) lub tutaj (efekt komendy \lstinline[basicstyle=\ttfamily]=tutaj=).

Poniżej zamieszczono przykłady kodów źródłowych z podświetleniem składni.

Listing 5.3: Opis 1

```
package pl.mrbarozoit.backend;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BackendApplication {

    public static void main(String[] args) {
        SpringApplication.run(BackendApplication.class, args);
    }

}
```

Jeśli w kodzie źródłowym jest jakiś nieistotny fragment względem omawianego problemu, to można go wykropkować (patrz listing 5.4).

Listing 5.4: Opis 2

```
// Karma configuration file, see link for more information
// https://karma-runner.github.io/1.0/config/configuration-file.html

module.exports = function (config) {
  config.set({
    basePath: '',
    frameworks: ['jasmine', '@angular-devkit/build-angular'],
    plugins: [
      require('karma-jasmine'),
      require('karma-chrome-launcher'),
      require('karma-jasmine-html-reporter'),
      require('karma-coverage-istanbul-reporter'),
      require('@angular-devkit/build-angular/plugins/karma')
```

```
    ],
    ... // opuszczony kod
    autoWatch: true,
    browsers: ['Chrome'],
    singleRun: false,
    restartOnFileChange: true
  });
};
```

Jeśli kod jest wąski, to można taki kod wrzucić w dwie kolumny. Przykład zamieszczono na listingu 5.5.

Listing 5.5: Kontrakt na model wejściowy endpointu /api/v1/chess/board/move.

```
{
    "lastPosition": {
        "fenDescription": "string"
    },
    "image": {
        ...
    },
    "positions": {
        "chessboardCorners": [
            ...
        ],
        "tilesCornerPoints": [
            ...
        ]
    },
    "referenceColors": {
        ...
    },
}
```

5.4. Wykaz literatury oraz cytowania

Cytowania powinny być zamieszczane w tekście z użyciem komendy \cite{}. Jej argumentem powinien być klucz cytowanej pozycji (lub lista kluczy rozdzielonych przecinkiem bez spacji, jeśli takich pozycji w danym miejscu cytuje się więcej) jaki jest używany w bazie danych bibliograficznych (plik dokumentacja.bib). Po kompilacji bibtex i pdflatex w tekście pojawia się właściwy odsyłacz do pozycji w wykazie literatury (ujęty w kwadratowe nawiasy – zgodnie z tym, co definiuje styl plabbrv.bst), zaś w samym wykazie (rozdział Literatura) – zacytowana pozycja. Przykładem cytowania jest: „dobrze to opisano w pracach [**?**, **?**]" (gdzie zastosowano komendę \cite{JS07,SQL2}).

Co do zawartości rekordów bibliograficznych - style bibtexowe potrafią „skracać" imiona (czyli wstawiać, jeśli taka wola, inicjały zamiast pełnych imion). Niemniej dobrze jest od razu przyjąć jakąś konwencję. Proponuje się, aby w rekordach od razu wstawiane były inicjały zamiast pełnych imion.

Niekiedy tytuły prac zawierają wyrazy z dużymi i małymi literami. Takie tytuły należy brać w podwójne nawiasy klamrowe, aby bibtex nie zamienił ich na postać, w której poza pierwszą literą pozostałe są małe.

Jeśli jakiś cytowany zasób pochodzi z Internetu, to jego rekord w pliku bib powinien wyglądać jak niżej.

```
@INPROCEEDINGS{SQL2,
  title={{A MySQL-based data archiver: preliminary results}},
  author={Bickley, M. and Slominski, Ch.},
  booktitle = {{Proceedings of ICALEPCS07}},
        month = oct,
        day = {15--19},
        year={2007},
  note={\url{http://www.osti.gov/scitech/servlets/purl/922267}
        [dostęp dnia 20 czerwca 2015]}
}
```

A to inny przykład rekordu danych bibliograficznych:

```
@TechReport{JS07,
        author = {Jędrzejczyk, J. and Środka, B.},
        title  ={Segmentacja obrazów metodą drzew decyzyjnych},
        year = {2007},
        institution = {Politechnika Wrocławska, Wydział Elektroniki}
}
```

5.5. Indeks rzeczowy

Generowanie indeksu po trosze wygląda jak generowanie wykazu literatury – wymaga kilku kroków. Podczas pierwszej kompilacji pdflatex generowany jest plik z rozszerzeniem *.idx (zawierający „surowy indeks"). Następnie, bazując na tym pliku, generowany jest plik z rozszerzeniem *.ind zawierający sformatowane dane. Ten krok wymaga uruchomienia odpowiedniego narzędzia oraz zastosowania plik z definicją stylu Dyplom.ist. W kroku ostatnim dokonuje się kolejnej kompilacji pdflatex (dzięki niej w wynikowym dokumencie pojawi się Indeks rzeczowy). Domyślnie Indeks rzeczowy zostanie sformatowany w układzie dwukolumnowym.

Oczywiście aby to wszystko zadziałało w kodzie szablonu należy umieścić odpowiednie komendy definiujące elementy indeksu rzeczowego (\index) oraz wstawiające sformatowany Indeks rzeczowy do dokumentu wynikowego (\printindex). Więcej informacji o tworzeniu indeksu rzeczowego można znaleźć na stronie https://en.wikibooks.org/wiki/LaTeX/Indexing. Poniżej przedstawiono przykłady komend użytych w szablonie do zdefiniowania elementów indeksu rzeczowego:

- \index{linia komend} – pozycji główna.
- \index{generowanie!-- indeksu} – podpozycja.

Generowanie pliku *.ind można inicjować na kilka sposobów:

- poprzez wydanie odpowiedniego polecenia bezpośrednio w linii komend

  ```
  makeindex Dyplom.idx -t Dyplom.ilg -o Dyplom.ind -s Dyplom.ist
  ```

- poprzez odpalenie odpowiedniego narzędzia środowiska. Na przykład w TeXnicCenter definiuje się tzw. output profiles:

  ```
  makeindex "%tm.idx" -t "%tm.ilg" -o "%tm.ind" -s "%tm.ist"
  ```

  a samo generowanie pliku *.ind zapewni wybranie pozycji menu Build/Makeindex.
- korzystając z odpowiednio sparametryzowanych pakietów i komend wewnątrz kompilowanego dokumentu (czyli od razu przy okazji jego kompilacji).

  ```
  \DisemulatePackage{imakeidx}
  \usepackage[noautomatic]{imakeidx}
  % jeśli chcemy, by indeks by generowany automatycznie programem makeindex
      ↪ :
  %\usepackage[makeindex]{imakeidx}
  % a tak ponoć można przekazać opcje do programu generującego indeks:
  %\makeindex[options=-s podrecznik -L polish -M lang/polish/utf8]
  %\makeindex[options=-s podrecznik]
  \makeindex
  ```

  Niestety, makeindex jest narzędziem, które umieszcza część pozycji w grupie Symbols, a nie w grupach związanych z literkami alfabetu. W związku z czym indeksowany element zaczynający się od polskiej literki trafia do grupy Symbols, jak np. \index{Światło}. Jeśli chce się zamieszczać w indeksie symbole matematyczne, to dobrze jest to robić jak w następującym przykładzie: \index{$asterisk@$\ast$} czy też \index{c@$\mathcal{C}$}, tj. dostarczając przy okazji klucz do sortowania. Lepiej w tym względzie radzą sobie inne narzędzia,

jak texindy lub xindy dostępne pod linuxem. Korzystając z nich uzyskuje się grupy polskich literek w indeksie rzeczowym (hasła zaczynające się od polskich literek już nie trafiają do grupy Symbols). Przykład polecenia wydanego z linii komend, w którym wykorzystano texindy zamieszczono poniżej (zakładamy kodowanie plików w UTF8, można dla niniejszego szablonu zmienić na cp1250):

```
texindy -L polish -M lang/polish/utf8 Dyplom.idx
```

To polecenie wygeneruje Dyplom.ind o zawartości:

```
\begin{theindex}
  \providecommand*\lettergroupDefault[1]{}
  \providecommand*\lettergroup[1]{%
      \par\textbf{#1}\par
      \nopagebreak
  }

  \lettergroup{G}
  \item generowanie
    \subitem -- indeksu, 27
    \subitem -- wykazu literatury, 27

  \indexspace

  \lettergroup{L}
  \item linia komend, 27

  \indexspace

  \lettergroup{Ś}
  \item \'Swiat\IeC {\l }o, 28

\end{theindex}
```

Aby mieć większą kontrolę automatyczne generowanie indeksu zostało w niniejszym szablonie wyłączone (indeks trzeba wygenerować samemu, wydając polecenie makeindex lub zalecane texindy).

5.6. Inne uwagi

Dobrym sposobem na kontrolę błędów występujących podczas kompilacji jest wstawianie linijki \end{document} w wybranym miejscu dokumentu. Jest to szczególnie przydatne w przypadkach, gdy błędy te są trudne do zidentyfikowania (gdy wygenerowane przez kompilator numery linii z błędami nie są tymi, w których błędy występują). Wystarczy wtedy przestawić wspomnianą linijkę do kolejnych miejsc, aż znajduję to miejsce, gdzie występuje problem.

Aby osiągnąć apostrofy maszynowe (czyli takie złożone z samych kresek) należy użyć polecenia "{}jak tutaj{}" (podwójny apostrof i podwójny apostrof z na wszelki wypadek umieszczonymi nawiasami klamrowymi, nawiasy są potrzebne z tej racji, iż podwójny apostrof przed niektórymi literkami zamienia je na literki z akcentami). W efekcie otrzymamy "jak tutaj". Jeśli natomiast apostrofy mają być drukarskie (czyli złożone z kropek i kresek), to należy użyć polecenia ,,jak tutaj'' (dwa pojedyncze przecinki i dwa pojedyncze apostrofy). W efekcie otrzymamy „jak tutaj”. Można też użyć znaków apostrofów odpowiednio zakodowanych „jak tutaj”, tylko że czasem trudno pisze się takie apostrofy w środowiskach kompilacji projektów latexowych.

Oto sposoby ustawienia odstępów między liniami:

- używając komendy \linespread{...} (akceptowalne), przy czym atrybutem tej metody jest współczynnik zależny od wielkości czcionki. Dla czcionki wiodącej 12pt odstęp półtora linii osiągnie się komendą \linespread{1.241}. Dla innych czcionek wiodących wartości tego parametru są jak w poniższym zestawieniu.

```
10pt 1.25 dla \onehalfspacing
     1.667 for \doublespacing,
                ponieważ ,,basic ratio'' = 1.2
               (\normalfont posiada \baselineskip rozmiaru 12pt)
11pt 1.213 dla \onehalfspacing oraz 1.618 dla \doublespacing,
     ponieważ ,,basic ratio'' = 1.236
               (\normalfont posiada \baselineskip rozmiaru 13.6pt)
12pt 1.241 dla \onehalfspacing oraz 1.655 dla \doublespacing,
     ponieważsince ''basic ratio'' is 1.208
               (\normalfont has a \baselineskip of 14.5pt)
```

  Kłopot w tym, że raz ustawiony odstęp będzie obowiązywał do wszystkich czcionek (nie działa tu żadem mechanizm zmiany współczynnika w zależności od wielkości czcionki akapitu).
- używając pakietu setspace (niezalecane). Ponieważ klasa memoir emuluje pakiet setspace, w preambule dokumentu należałoby umieścić:

```
\DisemulatePackage{setspace}
\usepackage{setspace}
```

  a potem można już sterować odstęp komendami:

```
\singlespacing
\onehalfspacing
\doubelspacing
```

  Ten sposób pozwala na korzystanie z mechanizmu automatycznej zmiany odległości linii w zależności od wielkości czcionki danego akapitu.
- korzystając bezpośrednio z komend dostarczonych w klasie memoir (zalecane):

```
\SingleSpacing
\OnehalfSpacing
\DoubleSpacing
```

  Ten sposób również pozwala na korzystanie z mechanizmu automatycznej zmiany odległości linii w zależności od wielkości czcionki danego akapitu.

Na koniec jeszcze uwaga o rozmiarze pliku wynikowego. Otóż pdflatex generuje pliki pdf, które zazwyczaj mogłyby być nieco lepiej skompresowane. Do lepszego skompresowania tych plików można użyć programu ghostscript. Wystarczy w tym celu wydać komendę (pod windowsami):

```
gswin64 -sDEVICE=pdfwrite -dCompatibilityLevel=1.4 -dNOPAUSE -dQUIET \
-dSAFER -dBATCH -sOutputFile=Dyplom-compressed.pdf Dyplom.pdf
```

W poleceniu tym można również wstawić opcję -dPDFSETTINGS=/prepress (zapewniającą uzyskanie wysokiej jakości, zachowanie kolorów, uzyskanie obrazków w rozdzielczości 300 dpi). Ze względów licencyjnych ghostscript używa domyślnie algorytmów z kompresją stratną. Przy kompresji może więc dojść do utraty jakości bitmap.

Rozdział 6

Podsumowanie

Podsumowanie jest miejscem, w którym należy zamieścić syntetyczny opis tego, o czym jest dokument. W szczególności w pracach dyplomowych w podsumowaniu powinno znaleźć się jawnie podane stwierdzenie dotyczące stopnia realizacji celu. Czyli powinny pojawić się w niej akapity ze zdaniami typu: „Podczas realizacji pracy udało się zrealizować wszystkie postawione cele". Ponadto powinna pojawić się dyskusja na temat napotkanych przeszkód i sposobów ich pokonania, perspektyw dalszego rozwoju, możliwych zastosowań wyników pracy itp.

6.1. Sekcja poziomu 1

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.

6.1.1. Sekcja poziomu 2

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.

Sekcja poziomu 3

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.

Paragraf 4 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.

6.2. Sekcja poziomu 1

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.

Literatura

[1] 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.

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

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

[4] 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.

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

[6] 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).

[7] 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).

[8] 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.

[9] 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.

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

[11] 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.

[12] 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).

[13] 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.