



AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Praca dyplomowa

*Rozpoznawanie tzw. „fake newsów”
na wąskim obszarze tematycznym*

*Recognition of so called „fake news”
on specified topic scope*

Autor:
Kierunek studiów:
Opiekun pracy:

Konrad Daniel Golemo
Automatyka i Robotyka
dr hab. inż. Jerzy Baranowski

Kraków, 2022

Spis treści

| | | |
|-----------|---|-----------|
| 1. | WSTĘP | 3 |
| 1.1. | WPROWADZENIE | 3 |
| 1.2. | CEL I ZAKRES PRACY | 6 |
| 1.3. | UKŁAD PRACY | 6 |
| 2. | ANALIZA AKTUALNEGO STANU WIEDZY | 7 |
| 2.1. | UCZENIE MASZYNOWE | 7 |
| 2.2. | PRZETWARZANIE JĘZYKA NATURALNEGO | 7 |
| 2.3. | WYNIKI REALIZACJI PODOBNYCH PROJEKTÓW | 9 |
| 2.3.1. | ROZPOZNAWANIE „FAKE NEWSÓW” W JĘZYKU ANGIELSKIM | 9 |
| 2.3.2. | ROZPOZNAWANIE „FAKE NEWSÓW” W JĘZYKU NIEMIECKIM | 11 |
| 2.3.3. | ROZPOZNAWANIE „FAKE NEWSÓW” W JĘZYKU HISPANISKIM | 12 |
| 3. | REALIZACJA PROJEKTU | 13 |
| 3.1. | WYBÓR OBSZARU TEMATYCZNEGO I TWORZENIE ZBIÓR DANYCH | 13 |
| 3.2. | PRZETWARZANIE ZBIORU DANYCH | 20 |
| 3.2.1. | OCZYSZCZANIE DANYCH | 20 |
| 3.2.2. | EKSTRAKCJA I SELEKCJA CECH TEKSTU | 22 |
| 3.3. | WYBRANE ALGORYTMY KLASYFIKACJI | 23 |
| 3.3.1. | NAIWNY KLASYFIKATOR BAYESOWSKI | 23 |
| 3.3.2. | LINIOWA MASZYNA WEKTORÓW NOŚNYCH | 24 |
| 3.3.3. | LAS LOSOWY | 25 |
| 3.4. | MODEL KLASYFIKATORA | 26 |
| 3.5. | TWORZENIE OPROGRAMOWANIA | 27 |
| 3.6. | DOKUMENTACJA | 28 |
| 3.7. | TWORZENIE APLIKACJI WEBOWEJ | 29 |
| 4. | REZULTATY | 30 |
| 4.1. | TESTY JEDNOSTKOWE FUNKCJI | 30 |
| 4.2. | EWALUACJA ALGORYTMÓW KLASYFIKACJI | 31 |
| 4.2.1. | ZASTOSOWANE METRYKI | 31 |
| 4.2.2. | NAIWNY KLASYFIKATOR BAYESOWSKI | 32 |
| 4.2.3. | LINIOWA MASZYNA WEKTORÓW NOŚNYCH | 33 |
| 4.2.4. | LAS LOSOWY | 34 |
| 4.3. | APLIKACJA WEBOWA | 35 |
| 5. | PODSUMOWANIE I WNIOSKI | 37 |
| 5.1. | ANALIZA WYNIKÓW KLASYFIKACJI | 37 |
| 5.2. | WNIOSKI | 38 |
| 5.3. | KIERUNKI DALSZEGO ROZWOJU | 38 |
| 6. | BIBLIOGRAFIA | 39 |

1. Wstęp

1.1. Wprowadzenie

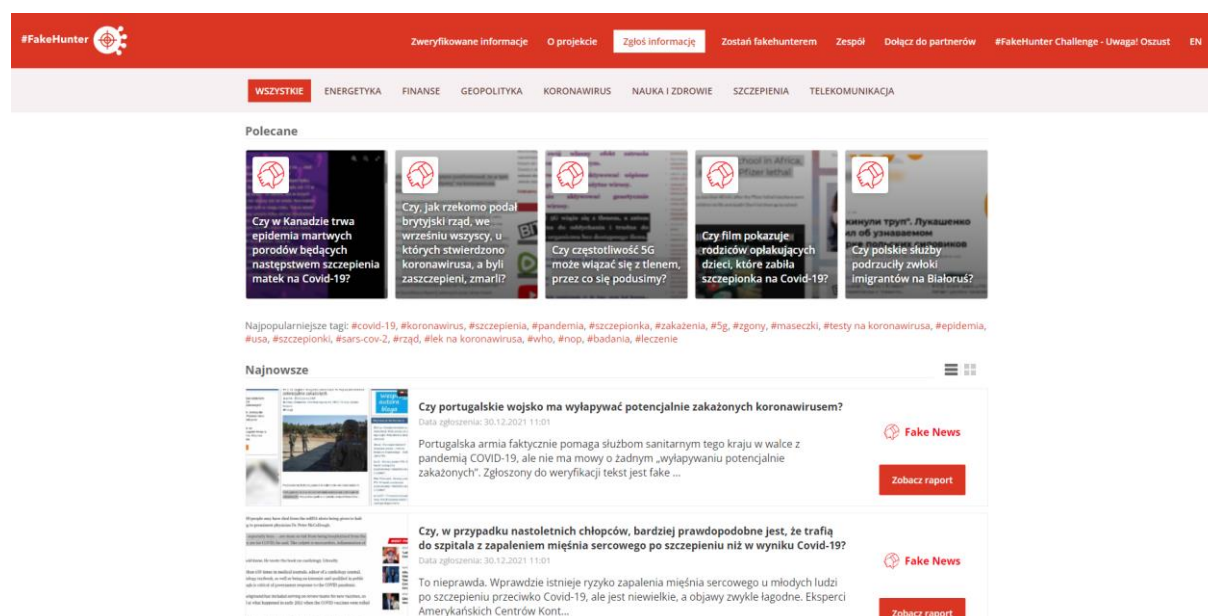
Pojęcie tzw. „fake newsów” odnosi się do fałszywych lub celowo mylących treści publikowanych jako prawdziwe wiadomości. Znaczny wzrost zainteresowania tym zagadnieniem nastąpił w 2016 roku, po wyborach prezydenckich w Stanach Zjednoczonych i referendum w Wielkiej Brytanii w sprawie opuszczenia Unii Europejskiej. [1]

W celu odróżnienia informacji prawdziwych od fałszywych, przeprowadzany jest proces tzw. „fact-checkingu”, którego elementy to np. weryfikacja cytatów czy sprawdzanie wiarygodności źródeł. [2]

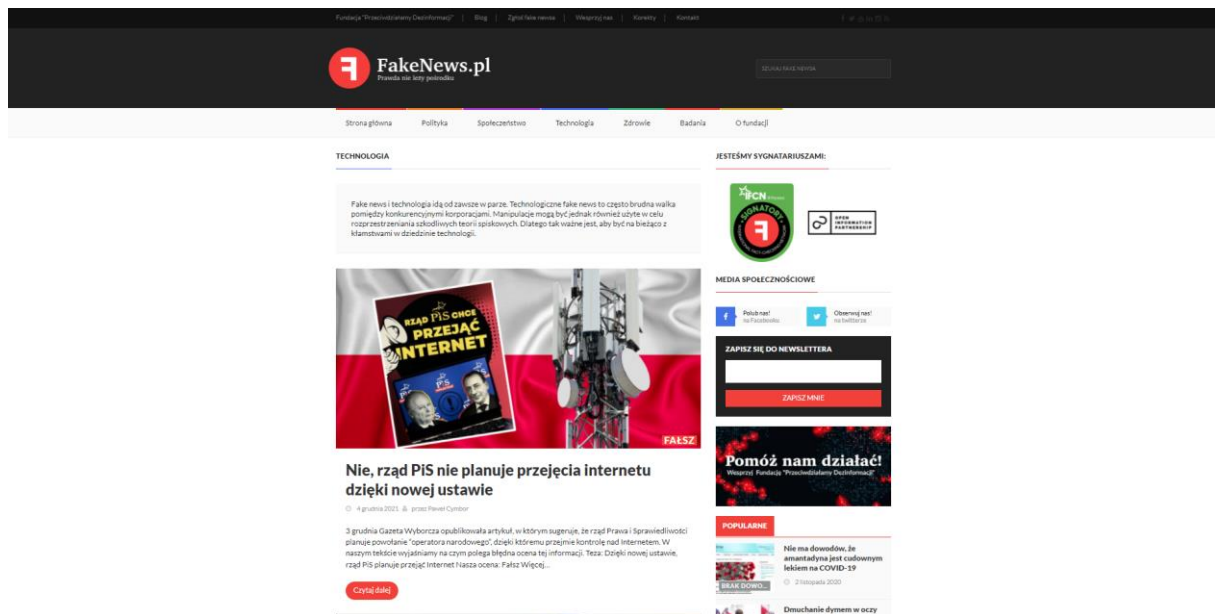
W Polsce funkcjonuje kilka organizacji weryfikujących informacje, m.in.:

- FakeHunter – serwis tworzony przez Polską Agencję Prasową i GovTech Polska [3],
- FakeNews.pl – portal fundacji „Przeciwdziałamy Dezinformacji” [4],
- AntyFAKE – projekt prowadzony przez Grupę Iberion [5],
- Stowarzyszenie Demagog – pierwsza w Polsce organizacja tego typu [6],
- Konkret24 – serwis należący do Grupy TVN [7].

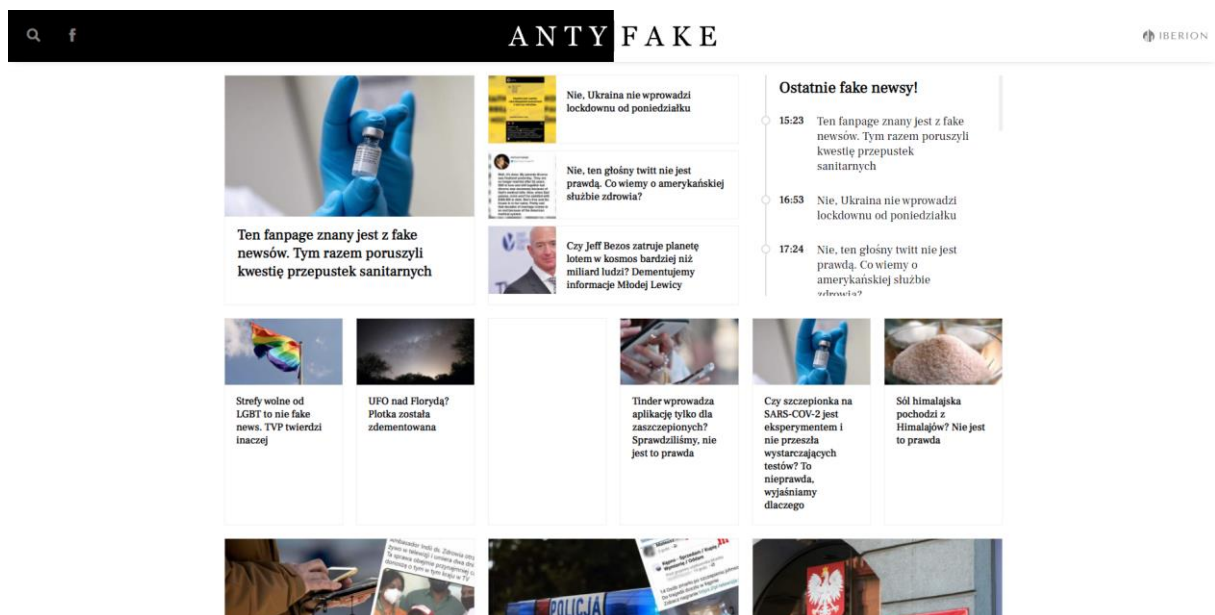
Na Rysunkach 1.1–1.5 przedstawiono strony główne wymienionych portali.



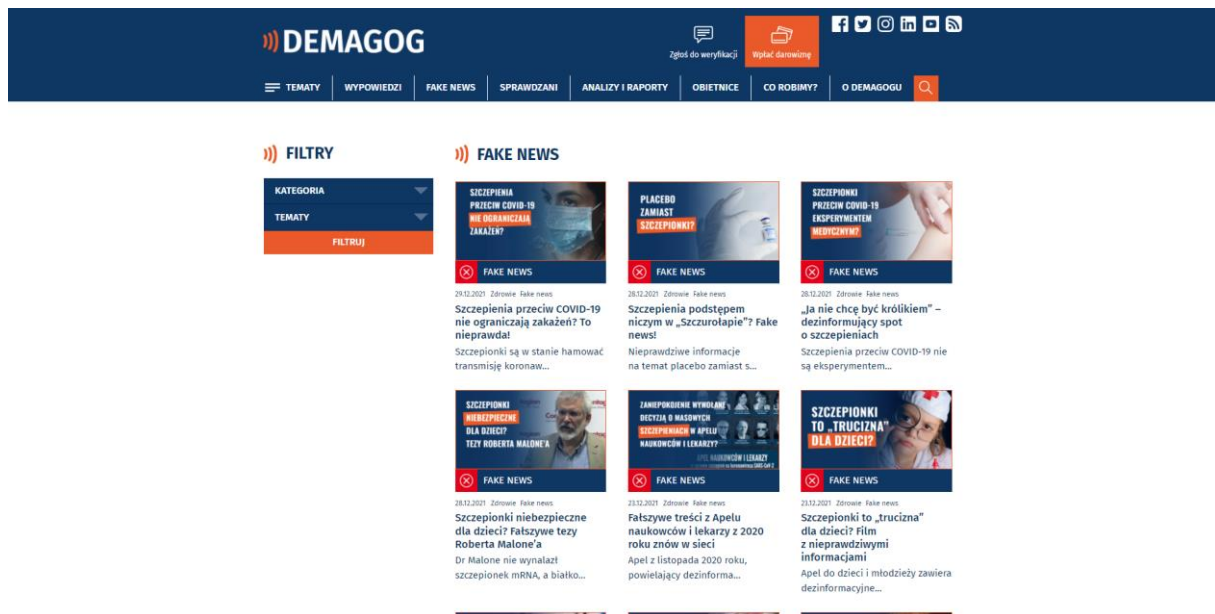
Rysunek 1.1. Strona główna portalu weryfikującego informacje FakeHunter [3]



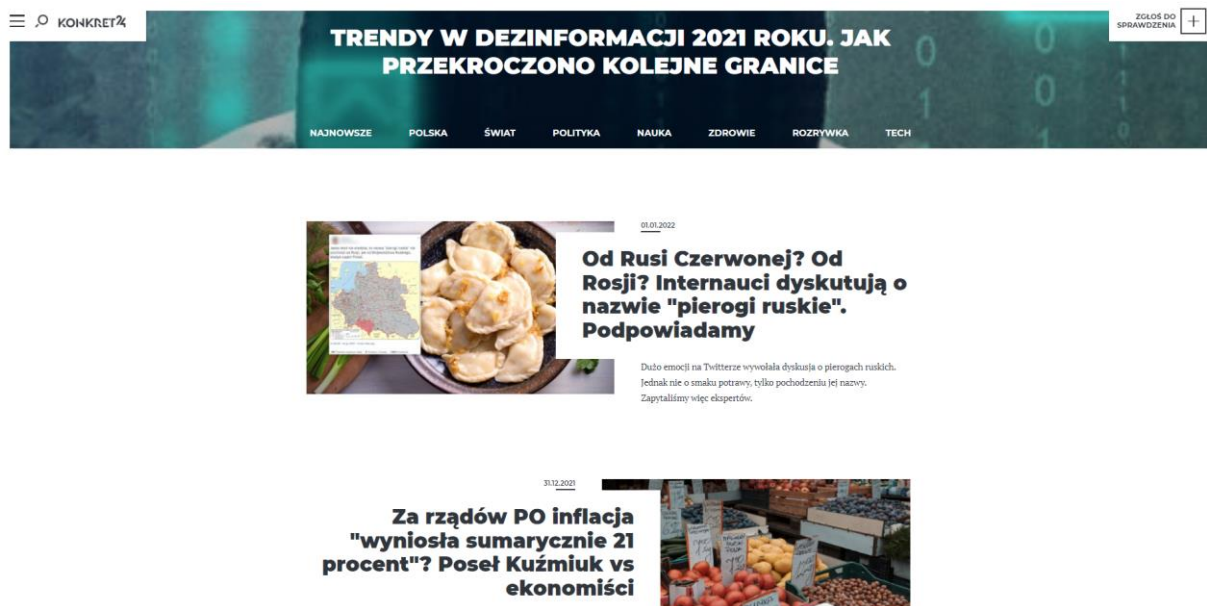
Rysunek 1.2. Strona główna portalu weryfikującego informacje FakeNews.pl [4]



Rysunek 1.3. Strona główna portalu weryfikującego informacje AntyFAKE [5]



Rysunek 1.4. Strona główna portalu weryfikującego informacje Demagog [6]



Rysunek 1.5. Strona główna portalu weryfikującego informacje Konkret24 [7]

1.2. Cel i zakres pracy

Celem niniejszej pracy jest zastosowanie podstawowych metod klasyfikacji tekstu do rozpoznawania „fake newsów”, dotyczących pewnego wybranego obszaru tematycznego, poprzez wykonanie następujących kroków:

- 1) Dokonanie analizy aktualnego stanu wiedzy.
- 2) Wybór obszaru tematycznego.
- 3) Stworzenie korpusu tekstów.
- 4) Wybór i zastosowanie algorytmów uczenia maszynowego.
- 5) Zaprojektowanie aplikacji i przeprowadzenie testów.

1.3. Układ pracy

Praca zorganizowana jest w następujący sposób: Rozdział 1 zawiera wprowadzenie w tematykę, cel i zakres pracy wraz z jej układem. W Rozdziale 2 przeprowadzono analizę aktualnego stanu wiedzy – opisano rozważane zagadnienia i dokonano przeglądu realizacji podobnych projektów. Rozdział 3 dotyczy sposobu wykonania projektu – uzasadniono w nim wybór obszaru tematycznego, omówiono proces gromadzenia i przygotowania danych, przedstawiono wybrane algorytmy klasyfikacji a także model kompletnego klasyfikatora. W Rozdziale 4 zaprezentowano rezultaty testów jednostkowych, stworzonego oprogramowania oraz wyniki ewaluacji zaprojektowanego klasyfikatora. Rozdział 5 zawiera analizę wyników, wnioski oraz propozycję kierunków możliwego rozwoju.

2. Analiza aktualnego stanu wiedzy

2.1. Uczenie maszynowe

Zgodnie z [8] uczenie maszynowe (ang. *Machine Learning, ML*) to ogół metod, za pomocą których, na podstawie dostępnych danych, dokonywana jest pewna predykcja. Wyróżnia się trzy główne grupy uczenia maszynowego:

- uczenie nadzorowane (ang. *supervised learning*) – w przypadku zbioru danych wejściowych ze znanymi oczekiwanymi wyjściami (etykietami w klasyfikacji, wartościami liczbowymi w regresji),
- uczenie nienadzorowane (ang. *unsupervised learning*) – w przypadku zbioru danych wejściowych bez określonych oczekiwanych wyjść,
- uczenie przez wzmacnianie (ang. *reinforcement learning*) – w przypadku, gdy dane wejściowe pobierane są ze środowiska, w którym funkcjonuje model, w celu maksymalizacji pewnej nagrody.

Schemat typowego postępowania w realizacji nadzorowanego uczenia maszynowego, składa się z następujących kroków:

- 1) Gromadzenie danych.
- 2) Wykorzystanie danych w algorytmie uczenia maszynowego.
- 3) Zastosowanie stworzonego modelu do wyznaczenia predykcji w oparciu o nowe dane.

Procedura ta, w kontekście niniejszej pracy, została szerzej opisana w Rozdziale 3.

2.2. Przetwarzanie języka naturalnego

Przetwarzanie języka naturalnego (ang. *Natural Language Processing, NLP*) to dziedzina badań zajmująca się wykorzystaniem komputerów w analizie tekstu i mowy ludzkiej, łącząca elementy m.in. informatyki, lingwistyki i sztucznej inteligencji. [9]

Do zastosowań przetwarzania języka naturalnego można zaliczyć [10] [11]:

- klasyfikację tekstu (np. detekcja „fake newsów”),
- tłumaczenie automatyczne,
- pozyskiwanie informacji z tekstu (np. rozpoznawanie nazw własnych),
- podsumowania tekstów, systemy odpowiadania na pytania,
- systemy dialogowe (tzw. „czatboty”).

Zgodnie z [8] [12] w przypadku klasyfikacji tekstu, a więc zadania nadzorowanego uczenia maszynowego, schemat typowego postępowania składa się z następujących kroków:

- 1) Gromadzenie danych.
- 2) Wstępne przetwarzanie danych.
- 3) Ekstrakcja i selekcja cech tekstu.
- 4) Wykorzystanie algorytmu uczenia maszynowego.

Na etapie wstępnego przetwarzania danych do najczęściej stosowanych metod należą:

- tokenizacja – podział sekwencji znaków na tokeny (poszczególne wyrazy lub zdania),
- filtracja – usunięcie wyrazów należących do listy tzw. „stop-words”, czyli często pojawiających się w tekście, lecz nie przekazujących informacji,
- lematyzacja – sprowadzenie wyrazów do ich podstawowych form (czasowników do formy bezokolicznika, rzeczowników do postaci mianownika liczby pojedynczej),
- tzw. „stemming” – usunięcie formantów fleksyjnych wyrazów.

Ekstrakcja cech tekstu zazwyczaj przeprowadzana jest za pomocą konwersji dokumentów do wektorów w przestrzeni cech, w których z każdym wyrazem (tokenem) związana jest pewna wartość liczbowa (waga tego wyrazu). Wyróżnia się dwa modele doboru wag:

- model boolowski – do każdego wyrazu należącego do dokumentu przypisana jest pewna dodatnia waga, a do każdego nienależącego – wartość 0,
- metoda TF-IDF (ang. *Term Frequency-Inverse Document Frequency*) – wagi wyznaczane są na podstawie liczby wystąpień wyrazów.

Do stosowanych algorytmów klasyfikacji zalicza się:

- naiwny klasyfikator bayesowski (ang. *Naive Bayes classifier, NB*) – klasyfikator probabilistyczny, zakładający niezależność występowania wyrazów w tekście, szerzej opisany w Rozdziale 3.3.1,
- liniową maszynę wektorów nośnych (ang. *Linear Support Vector Machine, LSVM*) – algorytm dokonujący separacji dostępnych klas za pomocą pewnej hiperpłaszczyzny, szerzej opisany w Rozdziale 3.3.2,
- drzewo decyzyjne (ang. *Decision Tree, DT*) – bazujące na rekurencyjnym podziale danych, szerzej opisane w Rozdziale 3.3.3,
- algorytm *K*-najbliższych sąsiadów (ang. *K-Nearest Neighbour classifier, KNN*) – klasa wyznaczana jest w oparciu o podobieństwo dokumentu do jego sąsiedztwa.

Całość wspomnianej procedury, w kontekście niniejszej pracy, została szerzej opisana w Rozdziale 3.

2.3. Wyniki realizacji podobnych projektów

2.3.1. Rozpoznawanie „fake newsów” w języku angielskim

W publikacji [13] wykorzystano zbiór danych zgromadzony przez serwis BuzzFeed News, zawierający 2282 artykułów o tematyce politycznej, przypisanych ręcznie do jednej z czterech kategorii: „głównie prawda”, „głównie fałsz”, „mieszanina prawdy i fałszu” oraz „brak treści merytorycznej”. Podczas implementacji klasyfikatora zredukowano liczebność zestawu poprzez odrzucenie artykułów nienależących do grupy „głównie prawda” lub „głównie fałsz” oraz rekordów niezawierających żadnego tekstu. Finalnie otrzymano zbiór zawierający łącznie 1771 artykułów, spośród których ok. 5% stanowiły „fake newsy”.

Do wykrywania tekstów zawierających nieprawdziwe informacje wykorzystano naiwny klasyfikator bayesowski (Rozdział 2.2 i 3.3.1). Dane podzielono na trzy zestawy: zbiór uczący, walidacyjny i testowy. W zaproponowanej procedurze klasyfikacji iterowano po każdym wyrazie w analizowanym artykule, a o przydzieleniu do jednej z dwóch grup (prawdziwy albo fałszywy) decydowało prawdopodobieństwo warunkowe wyznaczone na podstawie zawartych w tekście pewnych konkretnych wyrazów. Po wytrenowaniu klasyfikatora i dostrojeniu jego parametrów przystąpiono do testów i ewaluacji działania algorytmów za pomocą dokładności klasyfikacji (Rozdział 4.2.1), w wyniku której ogółem uzyskano wynik na poziomie ok. 75% (Tabela 2.1).

W celu poprawy działania klasyfikatora zaproponowano m.in. usunięcie z artykułów wyrazów należących do listy „stop-words” (Rozdział 2.2), wykorzystanie „stemmingu” (Rozdział 2.2), czy zmianę sposobu wyznaczania prawdopodobieństwa.

Tabela 2.1. Rezultaty ewaluacji klasyfikacji artykułów, przeprowadzonej przy pomocy dokładności [%], z podziałem na ich rodzaj, w analizowanej publikacji [13]

| Rodzaj artykułu | Dokładność [%] |
|-----------------|----------------|
| Prawdziwy | 76 |
| Fałszywy | 72 |
| Ogółem | 75 |

W pracy [14] zgromadzono 25 200 artykułów o tematyce politycznej, spośród których połowę stanowią wiadomości prawdziwe, a długość każdego z tekstów jest nie mniejsza niż 200 znaków. Dane podzielono na zbiór treningowy i testowy. Do określenia przestrzeni cech użyto modelu N -gramowego, który na podstawie $N - 1$ wyrazów przewiduje kolejny [11], a następnie zredukowano liczbę tych cech przy pomocy metody TF (Rozdział 3.2.2) oraz TF-IDF (Rozdział 2.2 i 3.2.2).

Klasyfikację przeprowadzono m.in. za pomocą liniowej maszyny wektorów nośnych (Rozdział 2.2 i 3.2.2), algorytmu K -najbliższych sąsiadów (Rozdział 2.2) oraz drzewa decyzyjnego (Rozdział 2.2 i 3.3.3). W trakcie trenowania algorytmów wykorzystano 5-krotny sprawdzian krzyżowy (ang. *k-fold cross-validation*, metoda polegająca na k -krotnym losowym podziale oryginalnego zbioru treningowego na mniejszy podzbiór treningowy i testowy, na których uczony jest algorytm uczenia maszynowego [11]).

W eksperymencie porównano dokładność klasyfikacji (Rozdział 4.2.1) dla obu metod ekstrakcji cech przy zmiennym rozmiarze N -gramu dla każdego z zastosowanych algorytmów. Największą dokładność (92%) osiągnięto dla algorytmu LSVM (Tabela 2.2).

Tabela 2.2. Rezultaty ewaluacji klasyfikacji artykułów z wykorzystaniem algorytmu LSVM, przeprowadzonej przy pomocy dokładności [%], z rozróżnieniem rozmiaru N -gramu, liczby cech i metod ich selekcji, w analizowanej publikacji [14]

| Rozmiar N -gramu | Liczba cech | Dokładność [%] | |
|-----------------------|----------------|----------------|----|
| | | TF-IDF | TF |
| Unigram | 1000 | 89 | 87 |
| | 5000 | 89 | 87 |
| | 10000 | 89 | 87 |
| | 50000 | 92 | 87 |
| Bigram | 1000 | 87 | 86 |
| | 5000 | 87 | 83 |
| | 10000 | 88 | 82 |
| | 50000 | 89 | 82 |

W artykule [15] wykorzystano ten sam zbiór danych, co w pozycji [13] (2282 tekstów o tematyce politycznej), lecz zaproponowano inną metodę podziału na treści wiarygodne i nieprawdziwe. Odrzucono wiadomości oznaczone etykietą „brak treści merytorycznej”, a te należące do grup „głównie fałsz” i „mieszanina prawdy i fałszu” połączono i potraktowano jako teksty fałszywe (342 pozycje). Pozostałe, należące do klasy „głównie prawda”, przydzielono do kategorii artykułów prawdziwych (1666 pozycje).

Klasyfikacji dokonano poprzez zastosowanie m.in: naiwnego klasyfikatora bayesowskiego (Rozdział 2.2 i 3.3.1), lasu losowego (ang. *Random Forest*, *RF*) (Rozdział 3.3.3), oraz XGBoost (metoda wzmacniająca gradient, bazująca na drzewach [16]). Cechy artykułów wyznaczono ręcznie, wyróżniając trzy kategorie: cechy tekstu (m.in. właściwości językowe), cechy źródła danych (m.in. uprzedzenia polityczne i wiarygodność) oraz cechy otoczenia (liczba interakcji użytkowników z postem).

Ewaluację klasyfikacji przeprowadzono za pomocą współczynnika F_1 -score (Rozdział 4.2.1) oraz obliczonego pola pod krzywą ROC (ang. *Receiver Operating Characteristic*) (Rozdział 4.2.1). Najlepsze rezultaty uzyskano dla algorytmu RF i XGBoost (Tabela 2.3).

Tabela 2.3. Rezultaty ewaluacji klasyfikacji artykułów, przeprowadzonej przy pomocy wielkości pola pod krzywą ROC oraz współczynnika F_1 -score [%], z rozróżnieniem rodzaju algorytmu klasyfikacji, w analizowanej publikacji [15]

| Algorytm klasyfikacji | Pole pod krzywą ROC | Współczynnik F_1 -score [%] |
|-----------------------|---------------------|-------------------------------|
| NB | 0,72 | 75 |
| RF | 0,85 | 81 |
| XGB | 0,86 | 81 |

2.3.2. Rozpoznawanie „fake newsów” w języku niemieckim

W artykule [17] zgromadzono zbiór danych zawierający 490 artykułów przekazujących nieprawdziwe informacje oraz 4500 wiadomości opublikowanych przez wiarygodnych wydawców głównego nurtu. Zastosowano dwie metody klasyfikacji: jedną opierającą się na maszynie wektorów nośnych (Rozdział 2.2 i 3.3.2), drugą na konwolucyjnej sieci neuronowej (ang. *Convolutional Neural Network*, *CNN*, wielowarstwowa sieć neuronowa wykorzystująca operacje konwolucji, często stosowana w przypadku dużej ilości danych [18]). Do wytrenowania klasyfikatorów wykorzystano teksty zawierające co najmniej 200 słów, co zredukowało liczbę „fake newsów” do 300.

W przypadku klasyfikacji z wykorzystaniem maszyny wektorów nośnych połowę zbioru artykułów wykorzystano do treningu, drugą część przeznaczając na testy. Na podstawie zestawu danych, za pomocą metody TF-IDF (Rozdział 2.2 i 3.3.2), określono wagę słów w tekście i wyznaczono odpowiadające im wektory cech. W ramach klasyfikatora opierającego się na CNN dane podzielono na trzy zestawy – treningowy (50% zbioru), walidacyjny (25%) i testowy (25%), a w celu wytrenowania architektury sieci neuronowej wykorzystano wcześniej przygotowane tzw. „word embeddings” (pewną wektorową reprezentację słów).

W wyniku ewaluacji za pomocą współczynnika F_1 -score (Rozdział 4.2.1), dla algorytmu SVM otrzymano wynik równy 74%, a dla metody wykorzystującej CNN 90% (Tabela 2.4).

Tabela 2.4. Rezultaty ewaluacji klasyfikacji artykułów, przeprowadzonej przy pomocy dokładności [%] oraz współczynnika F_1 -score [%], z rozróżnieniem rodzaju algorytmu klasyfikacji, w analizowanej publikacji [17]

| Algorytm klasyfikacji | Dokładność [%] | Współczynnik F_1 -score [%] |
|-----------------------|----------------|-------------------------------|
| SVM | 96 | 74 |
| CNN | 98 | 90 |

2.3.3. Rozpoznawanie „fake newsów” w języku hiszpańskim

W pracy [19] zebrano łącznie 971 artykułów z 9 kategorii tematycznych, w tym 491 oznaczonych jako prawdziwe i 480 jako fałszywe, które następnie podzielono na zestawy treningowe (70% zbioru) i testowe (30%). Jako reprezentację cech wykorzystano m.in.:

- tzw. „bag-of-words” (BOW) – metoda reprezentowania dokumentów za pomocą wektora, w którym ciąg 0 i 1 określa występowanie wyrazów z pewnego słownika [11],
- tzw. „part-of-speech tagging” (POS) – metoda przypisywania do każdego wyrazu odpowiadającej mu części mowy [11],
- znakowe N -gramy – metoda podziału każdego wyrazu na N -znakowe części [20],

a do klasyfikacji wykorzystano algorytmy: maszynę wektorów nośnych (Rozdział 2.2 i 3.3.2), regresję logistyczną (ang. *Logistic Regression*, *LR*, metoda regresji umożliwiająca wyznaczenie prawdopodobieństwa należenia do klasy [11]), las losowy (Rozdział 3.3.3) oraz tzw. „boosting” (metoda polegająca na stworzeniu dobrego algorytmu na podstawie słabych [21]).

Korzystając z dokładności (Rozdział 4.2.1) porównano działanie klasyfikatorów trenowanych na różnych zbiorach cech tekstu – BOW, POS oraz połącznie obu (Tabela 2.5).

Tabela 2.5. Rezultaty ewaluacji klasyfikacji artykułów, przeprowadzonej przy pomocy dokładności [%], z rozróżnieniem sposobu reprezentacji cech i zastosowanego algorytmu, w analizowanej publikacji [19]

| Reprezentacja cech | Dokładność klasyfikacji [%] | | | |
|--------------------|-----------------------------|----|-----------|----|
| | SVM | LR | RF | BO |
| BOW | 72 | 72 | 76 | 73 |
| POS | 68 | 67 | 64 | 61 |
| BOW+POS | 71 | 74 | 77 | 72 |

Zestawiono także dokładność klasyfikacji modeli bazujących na znakowych N -gramach z usuwaniem lub nie, wyrazów należących do listy „stop-words” (Tabela 2.6).

Tabela 2.6. Rezultaty ewaluacji klasyfikacji artykułów, przeprowadzonej przy pomocy dokładności [%], z rozróżnieniem rozmiaru N -gramu i zastosowanego algorytmu, w analizowanej publikacji [19]

| Rozmiar znakowego N -gramu | Dokładność klasyfikacji [%] | | | | | | | |
|------------------------------------|-----------------------------|----|----|----|---------------------|----|----|-----------|
| | usuwanie wyrazów | | | | nieusuwanie wyrazów | | | |
| | SVM | LR | RF | BO | SVM | LR | RF | BO |
| 3-gram | 69 | 67 | 70 | 70 | 73 | 71 | 75 | 75 |
| 4-gram | 72 | 70 | 71 | 72 | 76 | 77 | 75 | 77 |
| 5-gram | 74 | 70 | 74 | 73 | 76 | 76 | 76 | 76 |

W publikacji lepsze rezultaty uzyskano w przypadku korzystania z reprezentacji N -gramowej – 77% dokładności dla znakowego N -gramu o rozmiarze 4 i metody „boosting” oraz 77% dokładności dla kombinacji BOW z POS i algorytmu RF.

3. Realizacja projektu

3.1. Wybór obszaru tematycznego i tworzenie zbiorów danych

Na początku roku 2020 w związku z pandemią COVID-19 zaczęto obserwować znaczną liczbę „fake newsów”, podważających istnienie wirusa czy jego śmiertelność [22]. Z uwagi na mnogość wiadomości dotyczących koronawirusa, zarówno prawdziwych, jak i fałszywych, w niniejszej pracy zdecydowano się skupić właśnie na tym obszarze tematycznym. Do stworzenia korpusu, mogącego posłużyć jako dane w procesie klasyfikacji, konieczne było zgromadzenie odpowiedniej liczby tekstów, zawierających prawdziwe albo fałszywe informacje. W tym celu wykorzystano tzw. „web scraping”, czyli technikę pozyskiwania informacji z witryn internetowych [23], używając bibliotek Pythona: *Requests* [24] oraz *Beautiful Soup* [25].

Nieprawdziwe wiadomości pobrano z portalu FakeHunter [3], na którego łamach, na podstawie wiarygodnych źródeł, weryfikowana jest poprawność informacji przesłanych przez użytkowników. Przeanalizowane doniesienia są następnie publikowane w formie artykułów, zawierających werdykt („Fake News” albo „Prawda”) oraz raport eksperta wraz ze zdjęciem całości zgłoszonego tekstu lub jego fragmentu, a także adres jego źródła. Przykładowa analiza opublikowana w serwisie została przedstawiona na Rysunku 3.1.

Zgłoszony news

Główny Inspektorat Sanitarny zaprzecza, że istnieją „przykazy z góry”, by kary nakładane podczas pandemii przez państwowych inspektorów sanitarnych były jak najwyższe. Jak wyjaśnia jego rzecznik, było...

<https://warszawa.onet.pl/koronawirus-polska-sanepid-wystawil-juz-kary-na-5-mln-zi-ktu-decyduje-o-ich-wysokosci/s6xvqh>

Werdykt

Fake News

Raport eksperta

Data werdyktu: 20.04.2020 15:24

Główny Inspektorat Sanitarny zaprzecza, że istnieją „przykazy z góry”, by kary nakładane podczas pandemii przez państwowych inspektorów sanitarnych były jak najwyższe. Jak wyjaśnia jego rzecznik, byłoby to wręcz niezgodne z przepisami prawa obowiązującymi inspektorów podlegającej GIS Państwowej Inspekcji Sanitarnej.

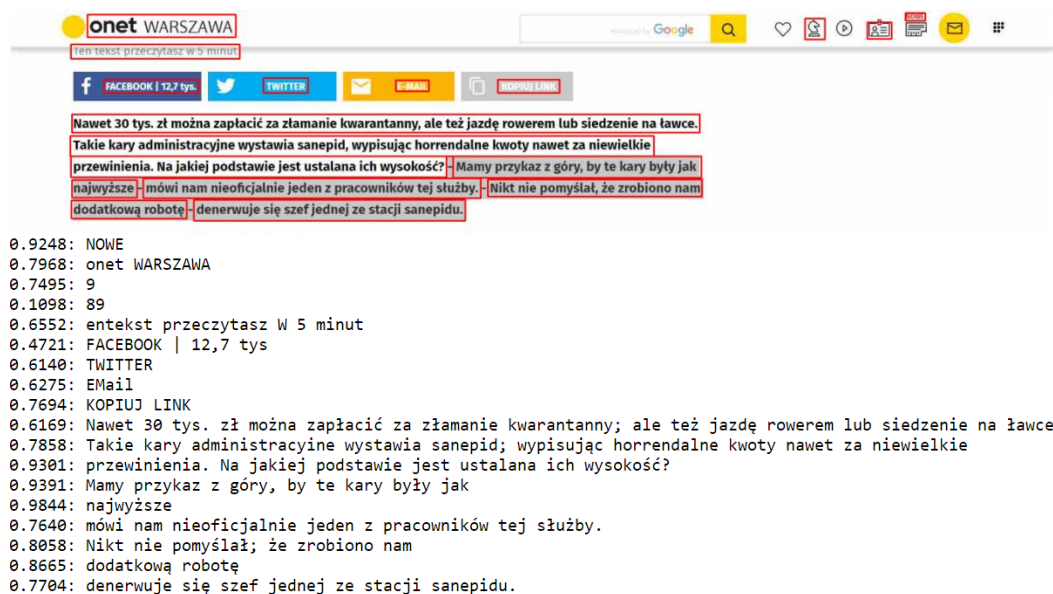
Rysunek 3.1. Przykładowy raport z weryfikacji opublikowany na łamach portalu FakeHunter [26] z widocznymi wykorzystywanymi w pracy elementami – zdjęciem wiadomości oraz werdyktem

W celu pobrania weryfikacji wykorzystano tzw. „API” (ang. *Application Programming Interface*) portalu, umożliwiające akwizycję zawartości strony do postaci słownika, składającego się z par klucz-wartość (np. do klucza „werdykt” przypisana jest wartość „fałszywy”). W ten sposób uzyskano dostęp do elementów strony widocznych na Rysunku 3.2.

```
"domains": [
  {
    "created_at": "2020-06-13T19:25:45.252751Z",
    "id": "a315ebf4-fb9f-42a1-bcc4-723eb61e7c64",
    "name": "koronawirus"
  }
],
"expert_opinion": {
  "title": "Czy inspektorzy sanepidu mogą dowolnie nakładać najwyższe kary za złamanie przepisów obowiązujących",
  "confirmation_sources": "Email od Jana Bodnara do redakcji #FakeHunter, Data: pon., 20 kwietnia 2020 roku,",
  "comment": "Główny Inspektorat Sanitarny zaprzecza, że istnieją „przykazy z góry”, by kary nakładane podcza",
  "date": "2020-04-20T13:24:42.367597Z",
  "verdict": "false"
},
"fact_checker_opinions": [
  {
    "title": "Wysokość kary zależy od interpretacji stanu faktycznego przez funkcjonariuszy publicznych.",
    "confirmation_sources": "https://www.gazetaprawna.pl/artykuly/1467609,gis-kara-lamanie-przepisow-zapobi",
    "comment": "W sytuacji naruszenia obowiązujących restrykcji, obywatel jest bezwarunkowo zdany na interp",
    "date": "2020-04-17T15:11:10.659753Z",
    "verdict": "false"
  }
],
"id": "2143bb38-fa97-4a76-a9f1-40a322d5e29b",
"is_pinned": false,
"reported_at": "2020-04-17T13:54:32.070705Z",
"screenshot_url": "https://sfnf-collector-prod.s3.amazonaws.com/ad0932bc-d807-4995-85ab-854e47c4d885.jpg",
"tags": [
  {
    "id": "91be111d-4dc7-44ef-b176-268e1a8f6a2b",
    "name": "sanepid",
    "created_at": "2020-06-09T18:20:49.812051Z"
  }
],
"text": "Główny Inspektorat Sanitarny zaprzecza, że istnieją „przykazy z góry”, by kary nakładane podczas pande",
"title": "Czy inspektorzy sanepidu mogą dowolnie nakładać najwyższe kary za złamanie przepisów obowiązujących",
"url": "https://warszawa.onet.pl/koronawirus-polska-sanepid-wystawil-juz-kary-na-5-mln-zl-kto-decyduje-o-ich-wy",
"verdict": "false",
"verified_by_expert": true
```

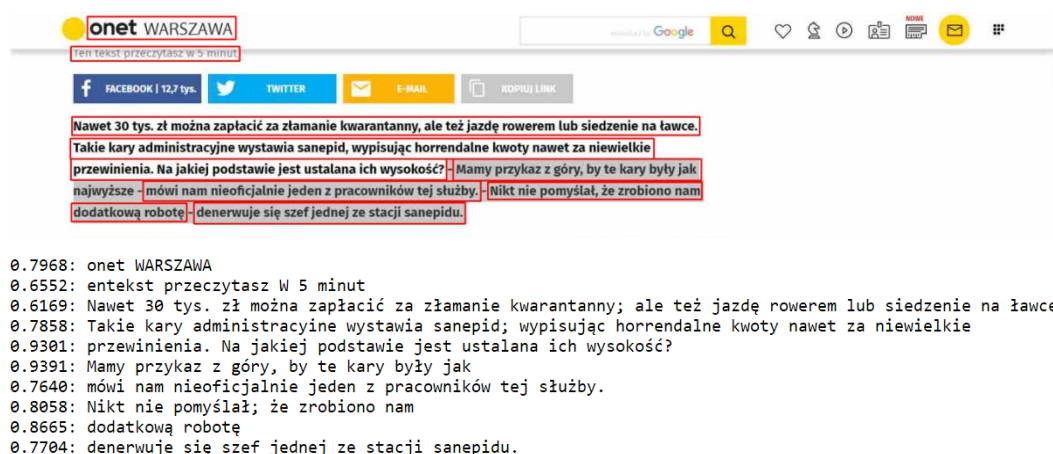
Rysunek 3.2. Raport z weryfikacji [26] widoczny w tzw. „API”, z widocznym polem „opinia eksperta”, zawierającym m.in. werdykt (niebieska ramka), a także elementami zawierającymi adres, pod którym znajduje się zdjęcie weryfikowanego artykułu (zielona ramka) oraz adres źródłowy tego artykułu (pomarańczowa ramka)

Mimo dostępu do witryn, z których pochodzą badane artykuły, niemożliwe było zaprojektowanie mechanizmu dokonującego akwizycji ich treści – każda strona projektowana jest w inny sposób, więc tekst może znajdować się w różnych jej elementach. Do wyodrębnienia artykułów stanowiących „fake newsy” użyto zamieszczonych w analizie zdjęć, zawierających część lub całość ich treści. W tym celu wykorzystano oprogramowanie rozpoznające znaki, tzw. „OCR” (ang. *Optical Character Recognition*), z biblioteki *EasyOCR* [27], którego przykład działania dla domyślnych parametrów i dekodera typu „beam-search” (algorytm przeszukiwania grafu, w którym w każdej iteracji pozostawiana jest jedynie grupa najbardziej obiecujących ścieżek [11]), widoczny jest na Rysunku 3.3.



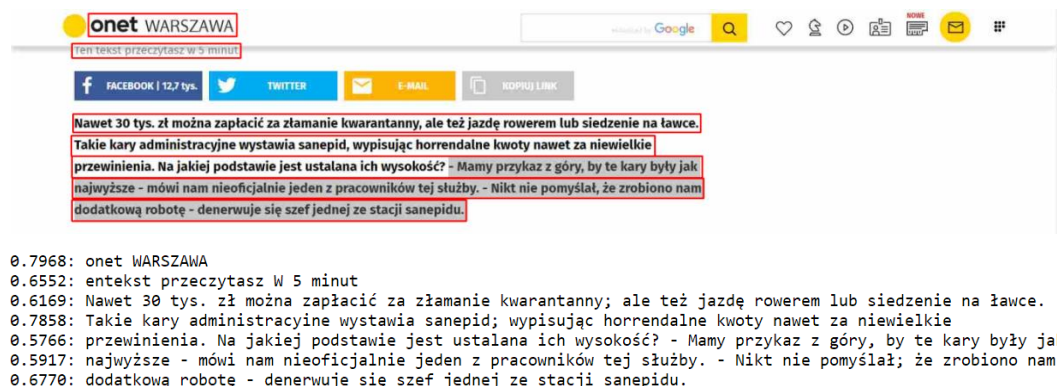
Rysunek 3.3. Rezultat ekstrakcji tekstu za pomocą wykorzystanego w pracy oprogramowania rozpoznającego znaki z przykładowego zdjęcia artykułu dla domyślnych parametrów detektora z zaznaczonymi czerwona ramką wykrytymi fragmentami wiadomości

Elementy zwracane przez to oprogramowanie to m.in. poziom pewności detekcji obiektu oraz jego treść. W uzyskanym rezultacie problem stanowiło jednak traktowanie jako tekst elementów, które w rzeczywistości nim nie były (np. elementy znajdujące się w prawym górnym rogu Rysunku 3.3), czy detekcja fragmentów odnośników do innych portali lub przycisków. W celu eliminacji tego typu zakłóceń określono minimalny rozmiar prostokąta otaczającego tekst, poprzez ustalenie wartości parametru *min_size* na 200 (domyślnie równego 10), w wyniku otrzymując detekcje przedstawione na Rysunku 3.4.



Rysunek 3.4. Rezultat ekstrakcji tekstu za pomocą wykorzystanego w pracy oprogramowania rozpoznającego znaki z przykładowego zdjęcia artykułu po zwiększeniu wartości parametru *min_size*, odpowiadającego za minimalny rozmiar detekcji. Wśród wykrytych elementów nie znajdują się już odnośniki do innych portali czy symbole graficzne

Kolejną przeszkodę stanowiło traktowanie wyrazów rozdzielonych myślnikiem jako osobne obiekty. Za łączenie prostokątów otaczających teksty odpowiada parametr *width_ths*, określający minimalną poziomą odległość między łączonymi elementami, którego wartość ustalono na 1 (domyślnie przyjmował 0,5). Rezultat przedstawiono na Rysunku 3.5.



Rysunek 3.5. Rezultat ekstrakcji tekstu za pomocą wykorzystanego w pracy oprogramowania rozpoznającego znaki z przykładowego zdjęcia artykułu po zwiększeniu wartości parametru *width_ths*, odpowiadającego za minimalną poziomą odległość między łączonymi fragmentami detekcji. Poszczególne wiersze zostały połączone w większe elementy

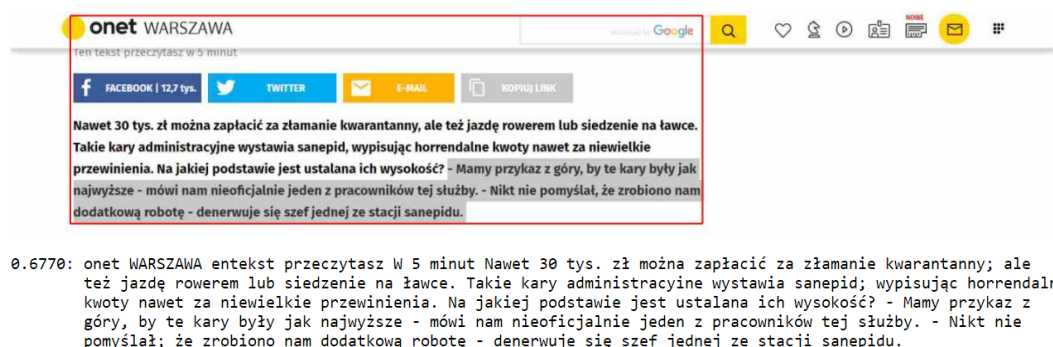
W następnym kroku połączono poszczególne wiersze wiadomości w większe obiekty poprzez ustawienie flagi *paragraph*, w wyniku czego uzyskano detekcję ukazaną na Rysunku 3.6.



Rysunek 3.6. Rezultat ekstrakcji tekstu za pomocą wykorzystanego w pracy oprogramowania rozpoznającego znaki z przykładowego zdjęcia artykułu po ustawieniu flagi *paragraph*, odpowiadającej za łączenie wykrytych wierszy w akapity. Poszczególne obiekty zostały scalone w większe fragmenty

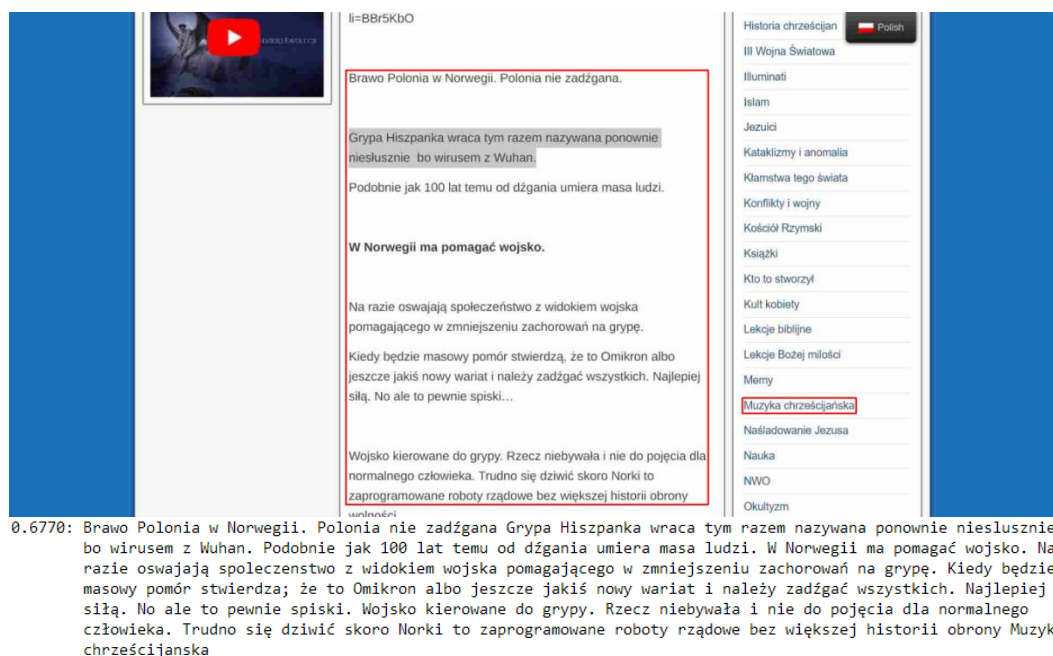
Uznano, że większe obiekty także powinny zostać połączone – w tym przykładzie osobny element stanowi nazwa serwisu, z którego pochodzi artykuł, co nie jest informacją bezużyteczną, a w niektórych przypadkach treść wiadomości może być podzielona większymi prze-
 rwami na akapity, które powinny zostać połączone. W tym celu ustalono wartość parametru

y_{ths} , określającego maksymalną pionową odległość między łączonymi obiektami, na 4 (domyślnie wartość parametru wynosiła 0,5), uzyskując rezultat widoczny na Rysunku 3.7.

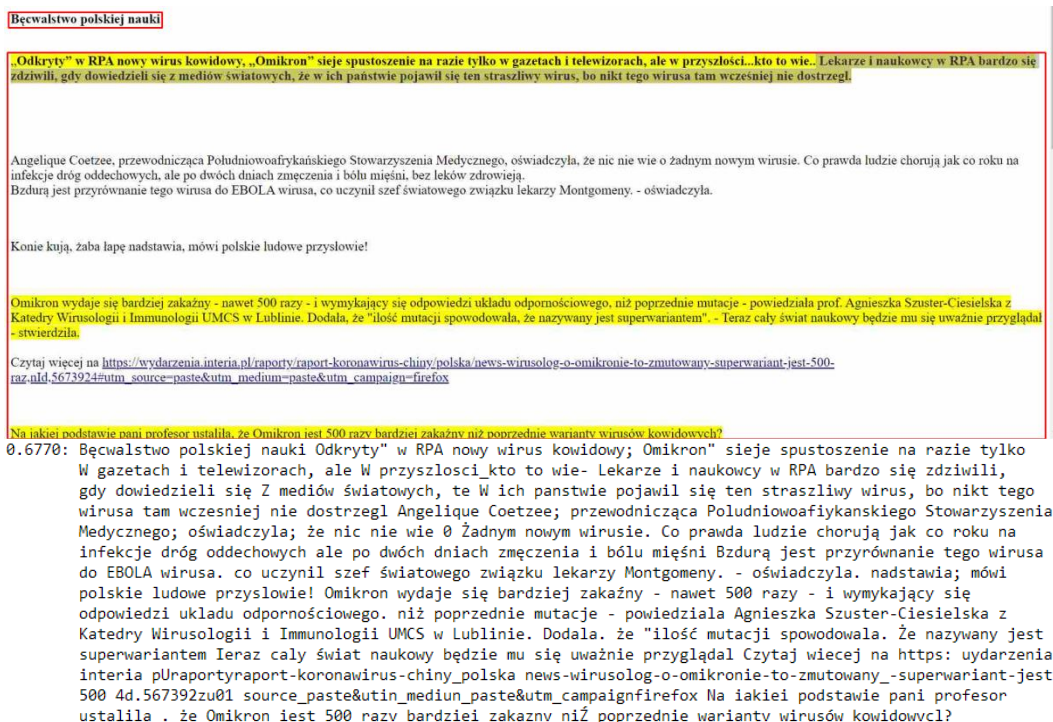


Rysunek 3.7. Rezultat ekstrakcji tekstu za pomocą wykorzystanego w pracy oprogramowania rozpoznającego znaki z przykładowego zdjęcia artykułu po zwiększeniu wartości parametru y_{ths} , odpowiadającego za minimalną pionową odległość między łączonymi fragmentami detekcji. Wykryte fragmenty zostały scalone w jeden obiekt

Poniżej, na Rysunkach 3.8 i 3.9, zamieszczono przykłady ekstrakcji cech z innych artykułów, na których widoczne są połączone w większe elementy poszczególne akapity tekstu. Na Rysunku 3.8 widoczny jest także element nienależący do treści wiadomości, który będąc częścią detekcji stanowi szum.



Rysunek 3.8. Rezultat ekstrakcji tekstu za pomocą wykorzystanego w pracy oprogramowania rozpoznającego znaki ze zdjęcia artykułu zamieszczonego w raporcie z weryfikacji opublikowanej na portalu FakeHunter [28]

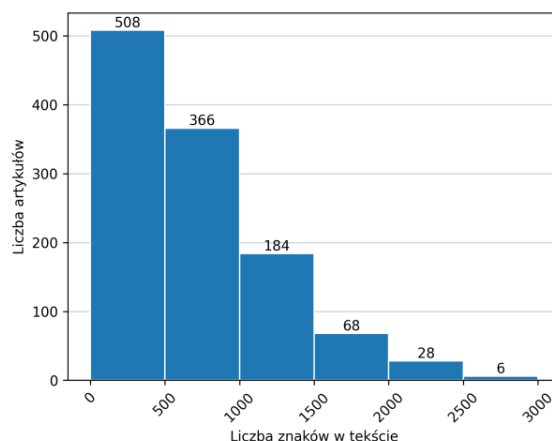


Rysunek 3.9. Rezultat ekstrakcji tekstu za pomocą wykorzystanego w pracy oprogramowania rozpoznającego znaki ze zdjęcia artykułu zamieszczonego w raporcie z weryfikacji opublikowanej na portalu FakeHunter [29]

Pseudokod procedury pobierania danych z portalu FakeHunter przedstawiono poniżej:

```
for strona in zakres_stron:
    weryfikacje = pobierz_weryfikacje(strona)
    for w in weryfikacje:
        tekst = pobierz_tekst_z_obrazu(weryfikacja[adres_obrazu])
        zapisz_do_tabeli(w[werdykt], w[tytuł], tekst, w[adres_artykułu])
```

Z portalu wyodrębniono 939 tekstów określonych jako „fake news”, 151 prawdziwych i 74 bez werdyktu. Artykuły te umieszczono w pliku *fakehunter_dataset.xlsx*. Na Rysunku 3.10 przedstawiono rozkład ich długości.



Rysunek 3.10. Histogram długości artykułów pobranych z portalu FakeHunter. Widoczna jest lewostronna asymetria rozkładu – długość większości zgromadzonych tekstów nie przekracza 500 znaków

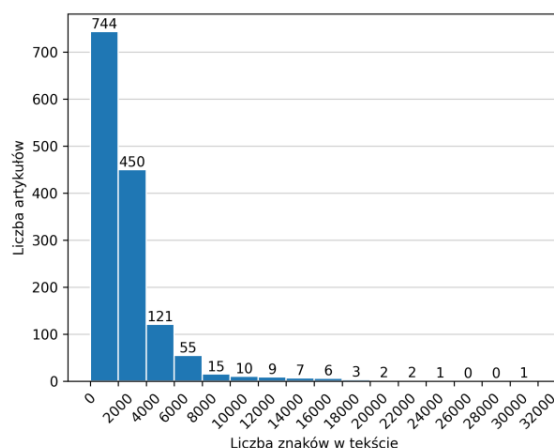
Jako bazę wiarygodnych wiadomości przyjęto serwis informacyjny wydawnictwa Termedia [30], publikujący artykuły o tematyce medycznej, w tym dotyczące wirusa SARS-CoV-2. Z uwagi na brak możliwości skorzystania z „API” serwisu, zawartość została wyodrębniona ze strony z wykorzystaniem parsowania kodu HTML witryny internetowej.

Pseudokod procedury pobierania danych z portalu Termedia przedstawiono poniżej:

```
for strona in zakres_stron:
    adresy_artykułow = pobierz_adresy(strona)
    for adres in adresy_artykułow:
        tytuł, tekst, adres_artykułu = pobierz_tekst_z_artykułu(adres)
        zapisz_do_tabeli('prawda', tytuł, tekst, adres_artykułu)
```

Z portalu wyodrębniono 1429 artykułów, które umieszczono w pliku *termedia_dataset.xlsx*.

Na Rysunku 3.11 przedstawiono rozkład długości tekstów.



Rysunek 3.11. Fragment histogramu długości artykułów pobranych z portalu Termedia. Widoczna jest lewostronna asymetria rozkładu – długość zdecydowanej większości zgromadzonych tekstów nie przekracza 4000 znaków

3.2. Przetwarzanie zbioru danych

3.2.1. Oczyszczanie danych

W ramach przygotowania zbioru danych z portalu FakeHunter do konwersji do przestrzeni cech tekstu, konieczne było wykonanie poniższych operacji:

- 1) Usunięcie rekordów niezawierających treści artykułu – przypadki, w których oprogramowanie wykrywające znaki nie odnalazło tekstu na zdjęciu.
- 2) Usunięcie tekstów niepolskojęzycznych – przypadki, w których artykuł pochodził z zagranicznego źródła, do ich wykrycia wykorzystano bibliotekę polyglot [31].
- 3) Usunięcie tekstów bez werdyktu – przypadki, w których zgłoszona wiadomość przekazywała informacje nieweryfikowalne.
- 4) Usunięcie tekstów błędnie wyodrębnionych z serwisu Twitter – przypadki, w których oprogramowanie wykrywające znaki niepoprawnie wyodrębniło niektóre teksty pochodzące z serwisu Twitter.

W przypadku artykułów z portalu Termedia, należało jedynie usunąć puste rekordy (przypadki, w których treść artykułu była niedostępna).

Następnie na obu wstępnie przetworzonych zestawach przeprowadzono kolejne operacje:

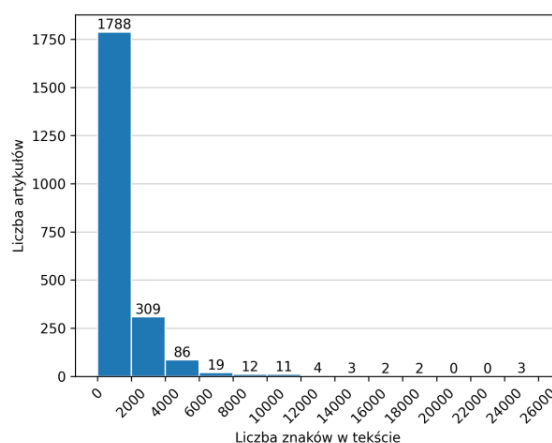
- 1) Usunięcie znaków ucieczki – znaków tabulacji, nowej linii itp. stanowiących jedynie utrudnienie analizy tekstu.
- 2) Usunięcie znaków interpunkcyjnych i nienależących do polskiego alfabetu – w przetwarzaniu języka naturalnego stanowiących szum, nieprzekazujący żadnej informacji.
- 3) Zastąpienie wielokrotnych spacji pojedynczą – stanowiących utrudnienie w procesie tokenizacji przeprowadzanym na podstawie pojedynczych spacji między wyrazami.
- 4) Zamiana wielkich liter w wyrazach na małe – wielkość liter nie ma znaczenia w przeprowadzanej analizie tekstów, nie ma więc potrzeby jej rozróżniania.
- 5) Podział tekstów na tokeny – rozdzielenie tekstu na pojedyncze wyrazy.
- 6) Usunięcie wyrazów należących do listy „stop-words” – są wyrazy nieprzekazujące informacji, często występujące w języku polskim lub będące spójnikami (Rozdział 2.2).
- 7) Przeprowadzenie lematyzacji – w tym celu wykorzystano analizator i generator fleksyjny dla języka polskiego *Morfeusz2* [32], którego przykład działania (z pominięciem ww. kroków przygotowawczych) przedstawiono na Rysunku 3.12.

| Zasięg | Segment | Lemat | Znacznik | Pospolitość | Kwalifikatory |
|--------|-----------------|-------------|-----------------------------|-----------------|---------------|
| 0-1 | [Rozpoznawanie] | rozpoznawać | ger:sg:nom.acc:nimperf:aff | | |
| 1-2 | [tzw] | tak_zwany | brev:pun | | |
| 2-3 | [,] | , | interp | | |
| 3-4 | [,] | , | interp | | |
| 4-5 | [fake] | fake | ign | | |
| 5-6 | [newsów] | news | subst:pl:gen:m2 | nazwa_pospolita | pot. |
| | | | subst:pl:gen:m3 | nazwa_pospolita | pot. |
| 6-7 | ["] | " | interp | | |
| 7-8 | [na] | na:P | prep:acc | | |
| | | | prep:loc | | |
| | | na:I | interp | | |
| 8-9 | [wąskim] | wąski | adj:pl:dat:m1.m2.m3.f.n:pos | | |
| | | | adj:sg:inst:m1.m2.m3.n:pos | | |
| | | | adj:sg:loc:m1.m2.m3.n:pos | | |
| 9-10 | [obszarze] | obszar | subst:sg:loc:m3 | nazwa_pospolita | |
| | | | subst:sg:voc:m3 | nazwa_pospolita | |
| 10-11 | [tematycznym] | tematyczny | adj:pl:dat:m1.m2.m3.f.n:pos | | |

Rysunek 3.12. Demonstracja analizy morfologicznej wykorzystanego do lematyzacji zgromadzonych w zbiorze danych tekstów programu *Morfeusz2* przeprowadzonej dla zdania „Rozpoznawanie tzw. »fake newsów« na wąskim obszarze tematycznym” [33]

- 8) Usunięcie rekordów niezawierających treści artykułu – w rezultacie ww. kroków pojawiły się do tej pory niewykryte rekordy bez użytecznego tekstu.
- 9) Usunięcie tekstów zbyt krótkich – w celu zmniejszenia szumu, zdecydowano się na analizowanie jedynie tekstów zawierających 30 lub więcej znaków.

Finalnie uzyskano zbiór składający się z 1497 artykułów przekazujących prawdziwe informacje i 753 fałszywych wiadomości – łącznie 2240 pozycji, które umieszczono w pliku *complete_dataset.xlsx*. Na Rysunku 3.13 przedstawiono rozkład długości pobranych tekstów, na którym widoczny jest wzrost liczby artykułów o liczbie znaków nieprzekraczającej 2 tysięcy, związany z dołączeniem do zbioru fałszywych wiadomości.



Rysunek 3.13. Histogram długości artykułów w kompletnym zbiorze danych. Widoczna jest lewostronna asymetria rozkładu – długość większości zgromadzonych tekstów nie przekracza 2000 znaków

3.2.2. Ekstrakcja i selekcja cech tekstu

W celu konwersji zgromadzonych dokumentów do wektorów cech, wykorzystano metodę TF-IDF (ang. *Term Frequency-Inverse Document Frequency*), która zgodnie z [11] przypisuje tokenom wagi według następującej zależności:

$$w_{w,d} = tf_{w,d} \cdot idf_w, \quad (3.1.1)$$

gdzie $tf_{w,d}$ to częstość występowania wyrazu w dokumencie określona jako stosunek liczby wystąpień wyrazu w dokumencie $n_{w,d}$ do liczby wszystkich wyrazów w dokumencie $\sum_i n_{i,d}$:

$$tf_{w,d} = \frac{n_{w,d}}{\sum_i n_{i,d}}, \quad (3.1.2)$$

a idf_w to logarytm dziesiętny ze stosunku liczby wszystkich dokumentów N do liczby dokumentów, w których występuje wyraz n_w :

$$idf_w = \log\left(\frac{N}{n_w}\right). \quad (3.1.3)$$

W realizacji projektu wykorzystano metodę TF-IDF z biblioteki *scikit-learn*, w której składnik IDF wyznaczany jest w następujący sposób [34]:

$$idf_w = \log\left(\frac{N + 1}{n_w + 1}\right) + 1. \quad (3.1.4)$$

Wyznaczone wektory TF-IDF są następnie normalizowane z wykorzystaniem normy euklidesowej [34]. Funkcja tworząca wektory cech umożliwia także wyznaczanie ich w oparciu o wyrazowe N -gramy, w analizie wykorzystano więc uni- i bigramy.

Zestaw danych został podzielony na zbiór treningowy (70% danych, 1568 tekstów) i testowy (30% danych, 672 teksty), a następnie dokonano wektoryzacji, w wyniku której uzyskano 27 416 cech w przypadku unigramów i 193 552 dla reprezentacji bigramowej. Selekcję cech przeprowadzono w oparciu o wybór jedynie K najczęściej występujących cech pod względem współczynnika $tf_{w,d}$ w korpusie. W kolejnym kroku zastosowano wymienione w Rozdziale 3.3 algorytmy klasyfikacji.

3.3. Wybrane algorytmy klasyfikacji

3.3.1. Naiwny klasyfikator bayesowski

Naiwny klasyfikator bayesowski (ang. *Naive Bayes classifier*, *NB*) to probabilistyczna metoda klasyfikacji, której zadaniem jest wyznaczenie zgodnie z twierdzeniem Bayesa najbardziej prawdopodobnej klasy dla danego wektora zmiennych. Istotnym uproszczeniem, znacznie usprawniającym proces uczenia (zwłaszcza przy dużej liczbie zmiennych), jest założenie o wzajemnej niezależności predyktorów, umożliwiające wyznaczanie parametrów dla każdej ze zmiennych oddzielnie. [35]

Zgodnie z [36] prawdopodobieństwo przynależności wektora cech tekstu X do klasy y jest równe:

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)} = \frac{P(x_1, x_2, \dots, x_n|y) \cdot P(y)}{P(x_1, x_2, \dots, x_n)}. \quad (3.2.1)$$

Uwzględniając założenie o niezależności predyktorów:

$$P(y|X) = \frac{P(y) \cdot \prod_{i=1}^n P(x_i|y)}{P(X)}. \quad (3.2.2)$$

Zgodnie z [37], z uwagi na fakt, że mianownik $P(X)$ jest stały, reguła klasyfikacji określona jest zależnością:

$$P(y|X) \propto P(y) \cdot \prod_{i=1}^n P(x_i|y), \quad (3.2.3)$$

a więc najbardziej odpowiednią klasę dla danego wektora cech tekstu stanowi:

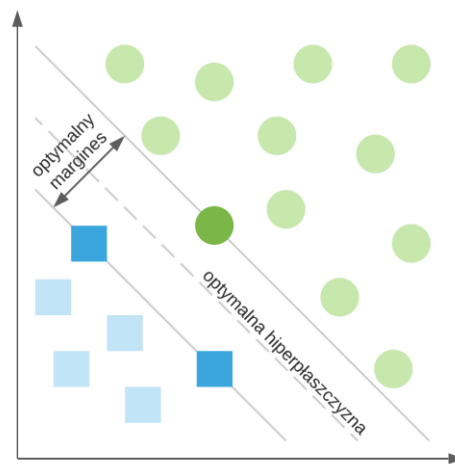
$$\hat{y} = \arg \max_y \hat{P}(y) \cdot \prod_{i=1}^n \hat{P}(x_i|y), \quad (3.2.4)$$

gdzie wartości częstości występowania klasy y w zbiorze danych ($\hat{P}(y)$) oraz częstości występowania cechy x_i w tekstach należących do klasy y ($\hat{P}(x_i|y)$) są szacowane na podstawie zestawu danych uczących.

W naiwnym klasyfikatorze bayesowskim z biblioteki *scikit-learn*, wykorzystanym w realizacji projektu, wartość $\hat{P}(x_i|y)$ jest estymowana z uwzględnieniem wygładzania addytywnego, co ma na celu wyeliminowanie zerowania się tego składnika w przypadkach, gdy dana para cecha-klasa nie wystąpiła w zbiorze treningowym. [38]

3.3.2. Liniowa maszyna wektorów nośnych

Liniowa maszyna wektorów nośnych (ang. *Linear Support Vector Machine, LSVM*) to algorytm stosowany w problemach klasyfikacji binarnej, którego celem jest określenie hiperpłaszczyzny rozdzielającej zestaw danych pomiędzy dwie klasy. Wymiar tej hiperpłaszczyzny zależy od liczebności zbioru cech, na podstawie którego będzie ona wyznaczana. Margines wokół granicy (funkcji) decyzyjnej, przedstawiony na Rysunku 3.14, maksymalizowany jest na podstawie par wektorów cech x_i i odpowiadających im etykiet y_i . [39].



Rysunek 3.14. Przykład funkcji decyzyjnej dla danych liniowo separowalnych z oznaczonymi ciemniejszym odcieniem wektorami nośnymi [opracowanie własne na podstawie (40)]

Zgodnie z [40] zestaw danych uczących $(x_1, y_1), \dots, (x_N, y_N)$, $y_i \in \{-1, 1\}$ jest liniowo separowalny, jeśli nierówność:

$$y_i(w \cdot x_i + b) \geq 1, \quad (3.3.1)$$

jest spełniona dla wszystkich jego elementów.

Wektory, dla których $y_i(w \cdot x_i + b) = 1$ to tzw. wektory nośne, na podstawie których w przestrzeni cech konstruowana jest optymalna hiperpłaszczyzna określona zależnością:

$$w_{opt} \cdot x + b_{opt} = 0, \quad (3.3.2)$$

rozdzielająca dane uczące z największym możliwym marginesem, opisanym wzorem:

$$d(w_{opt}, b_{opt}) = \frac{2}{|w_{opt}|} = \frac{2}{\sqrt{w_{opt} \cdot w_{opt}}}. \quad (3.3.3)$$

Wyznaczanie hiperpłaszczyzny (3.3.2) sprowadza się do rozwiązania problemu programowania kwadratowego – minimalizacji iloczynu $w \cdot w$ przy ograniczeniach (3.3.1).

Maszyna wektorów nośnych z biblioteki *scikit-learn*, wykorzystana w realizacji projektu, przy obliczaniu współczynników hiperpłaszczyzny uwzględnia dodatkowo funkcję straty typu hinge loss, kontrolującą wpływ predykcji ze zbyt małym marginesem lub błędnych. [41]

3.3.3. Las losowy

Las losowy (ang. *Random Forest, RF*) to metoda uczenia maszynowego działająca w oparciu o grupę drzew decyzyjnych (ang. *Decision Tree, DT*).

Zgodnie z [42], algorytm drzewa decyzyjnego dokonuje rekurencyjnego, binarnego podziału przestrzeni cech według poszczególnych zmiennych objaśniających (predyktorów), aż do momentu spełnienia kryterium zatrzymania. W rezultacie otrzymywane są niepodzielne węzły, zwane końcowymi, na podstawie których określana jest predykcja algorytmu. Do zalet drzew decyzyjnych należą m.in.:

- możliwość stosowania w problemach klasyfikacji i regresji,
- możliwość opisanie zależności między zmiennymi objaśniającymi,
- odporność na braki danych, odstające wartości i predyktory bez znaczenia.

Istotną wadę drzew decyzyjnych stanowi m.in. niestabilność (nieznaczna zmiana w przestrzeni zmiennych skutkuje innym drzewem), a także mniejsza dokładność względem nowszych metod (np. sieci neuronowych).

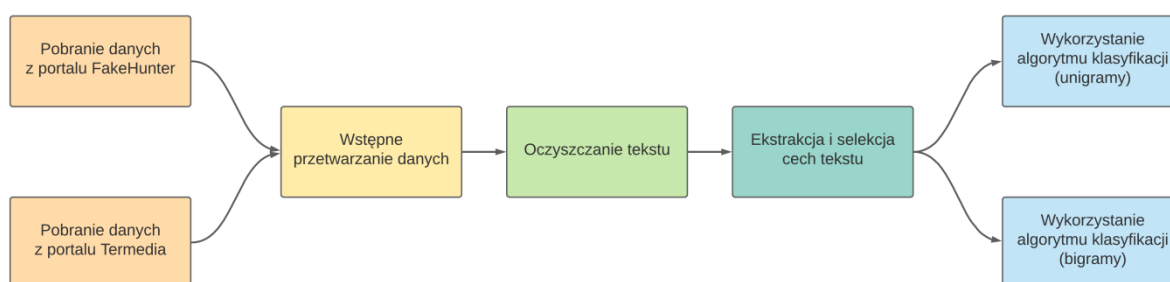
W celu poprawienia rezultatów, drzewa, z których składa się las losowy, tworzone są na podstawie wylosowanego ze zwracaniem podzbioru dostępnych danych, a podział w węzłach tych drzew odbywa się z wykorzystaniem pewnej wylosowanej liczby predyktorów – poprzez wybór najlepszego podziału dla tego podzbioru zmiennych objaśniających. W przypadku klasyfikacji prognozę algorytmu stanowi wynik głosowania wśród drzew (klasa wybrana przez większość), a przy wyznaczaniu wartości liczbowej zmiennej objaśnianej w regresji, średnia wyników uzyskanych przez poszczególne drzewa. [43]

W modelu lasu losowego z biblioteki *scikit-learn*, wykorzystanym w realizacji projektu, prognozę stanowi średnia predykcji wszystkich drzew, na podstawie której określana jest przynależność do klasy, a nie klasa wybrana przez większość. [44]

3.4. Model klasyfikatora

Kompletny model klasyfikatora wykorzystany w niniejszej pracy, przedstawiony na Rysunku 3.15, składa się z następujących elementów:

- 1) Pobranie danych z portalu FakeHunter i Termedia (Rozdział 3.1).
- 2) Wstępne przetwarzanie danych (Rozdział 3.2.1).
- 3) Oczyszczanie tekstu (Rozdział 3.2.1).
- 4) Ekstrakcja i selekcja cech tekstu (Rozdział 3.2.2).
- 5) Wykorzystanie algorytmu klasyfikacji (Rozdział 3.3).



Rysunek 3.15. Model klasyfikatora wykorzystanego w niniejszej pracy do detekcji tzw. „fake newsów”, obrazujący wszystkie fazy związane z przygotowaniem danych i użyciem ich do wytrenowania oraz przetestowania algorytmów klasyfikacji

Parametry algorytmów klasyfikacji dobierano za pomocą metody tzw. „grid search” z biblioteki *scikit-learn*, która dokonuje optymalizacji modelu na podstawie przeglądu siatki parametrów i wykorzystania mechanizmu sprawdzianu krzyżowego [45].

W przypadku naiwnego klasyfikatora bayesowskiego manipulowano wartością współczynnika α , odpowiadającego za stopień wygładzania addytywnego (0 oznacza brak wygładzania) [38], wybierając go ze zbioru {0,001; 0,003; 0,01; 0,03; 0,1; 0,3; 1}.

W przypadku liniowej maszyny wektorów nośnych modyfikowano stałą regularyzacji C (stopień regularyzacji jest odwrotnie proporcjonalny do C) [41], wybierając jej wartość ze zbioru {0,01; 0,03; 0,1; 0,3; 1}.

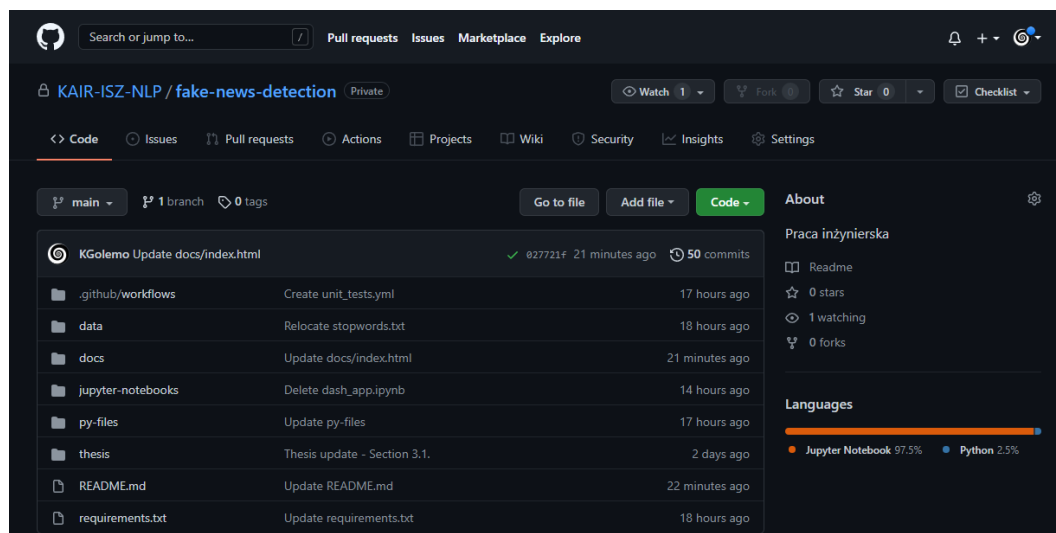
W przypadku lasu losowego zmieniano parametr n (liczbę dostępnych drzew w lesie) [44], wybierając go ze zbioru {10; 30; 100; 300; 1000}.

Rezultaty ewaluacji modeli klasyfikatora zostały omówione w Rozdziale 4.2.

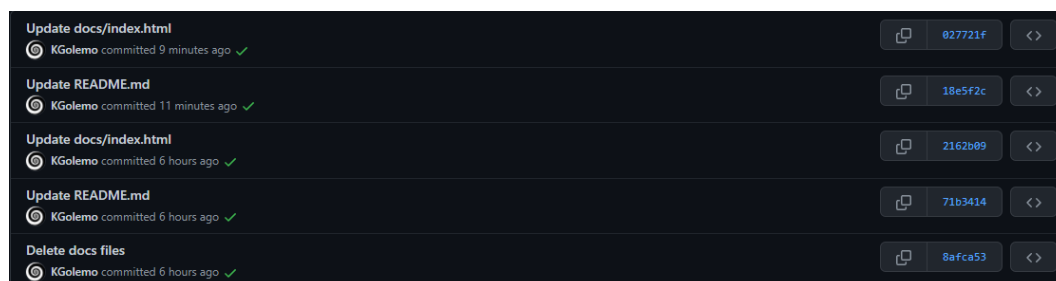
3.5. Tworzenie oprogramowania

Projekt został zrealizowany z wykorzystaniem notatników Jupyter (ang. *Jupyter Notebooks*) [46], umożliwiających tworzenie kodu w języku Python w formie uruchamialnych dokumentów. Proces rozwoju oprogramowania przeprowadzony został w oparciu o system kontroli wersji Git, z wykorzystaniem założonego w serwisie GitHub repozytorium projektu [47]. Umożliwiło to m.in. śledzenie zmian w plikach, czy wykorzystanie mechanizmu ciągłej integracji (Rozdział 4.1).

Na Rysunku 3.16 przedstawiono panel główny repozytorium, z widocznym m.in. katalogiem *jupyter-notebooks* zawierającym wspomniane notatniki, czy *data*, w którym przechowywany jest zbiór danych. Lokalne zmiany, nanoszone w trakcie tworzenia oprogramowania, były każdorazowo przesyłane do repozytorium za pomocą tzw. „commitów”, których historia widoczna jest na Rysunku 3.17.



Rysunek 3.16. Panel główny repozytorium projektu z widocznym m.in. katalogiem *thesis* zawierającym niniejszą pracę oraz *data*, w którym przechowywany jest zbiór danych



Rysunek 3.17. Fragment historii tzw. „commitów”, czyli operacji przesłania zmian do repozytorium, z widocznymi zielonym znacznikami związanymi z mechanizmem ciągłej integracji (Rozdział 4.1)

3.6. Dokumentacja

Dokumentacja opracowanego oprogramowania w formie interaktywnej witryny internetowej [48] została wygenerowana za pomocą narzędzia *pdoc3* [49] na podstawie dodanych do każdej zaimplementowanej funkcji przetwarzającej zbiór danych (Rozdział 3.2) tzw. „docstringów” [50]. Elementy te, widoczne na Rysunku 3.18, stanowią opis realizowanego przez funkcję zadania, jej argumentów oraz zwracanych przez nią obiektów. Na Rysunku 3.19 przedstawiono fragment dokumentacji z widocznym panelem nawigacyjnym, umożliwiającym przemieszczanie się między opisami wszystkich funkcji.

```
def extract_text_from_article(url: str) -> Tuple[str, str]:
    """Gets the article's title and content.

    Args:
        url (str): URL of the article.

    Returns:
        (Tuple [str, str]): Tuple containing:
            extract_text_from_image (str): Article's title.
            text (str): Article's content.
    """
```

Rysunek 3.18. Przykładowy opis za pomocą tzw. „docstringa” funkcji odpowiadającej za wyodrębnienie tekstu z artykułu z portalu Termedia

Module `web_scraping_functions`

```
extract_text_from_article
extract_text_from_image
get_articles_links
get_verifications
```

Module `preprocessing_functions`

```
change_verdict_dtype
delete_escape_chars
delete_stop_words
drop_empty
drop_non_polish
drop_short
drop_title_and_url
drop_twitter
drop_unidentified
lemmatize
lowercase_all
replace_whitespace
strip_non_polish
tokenize
```

Module `web_scraping_functions`

► EXPAND SOURCE CODE

```
def extract_text_from_article(url: str) -> Tuple[str, str]
```

Gets the article's title and content.

Args

```
url : str
    URL of the article.
```

Returns

```
(Tuple [str, str]): Tuple containing: title (str): Article's title. text (str): Article's content.
```

► EXPAND SOURCE CODE

```
def extract_text_from_image(url: str) -> str
```

Performs OCR on the screenshot of an article.

Args

```
url : str
    URL of the screenshot.
```

Returns

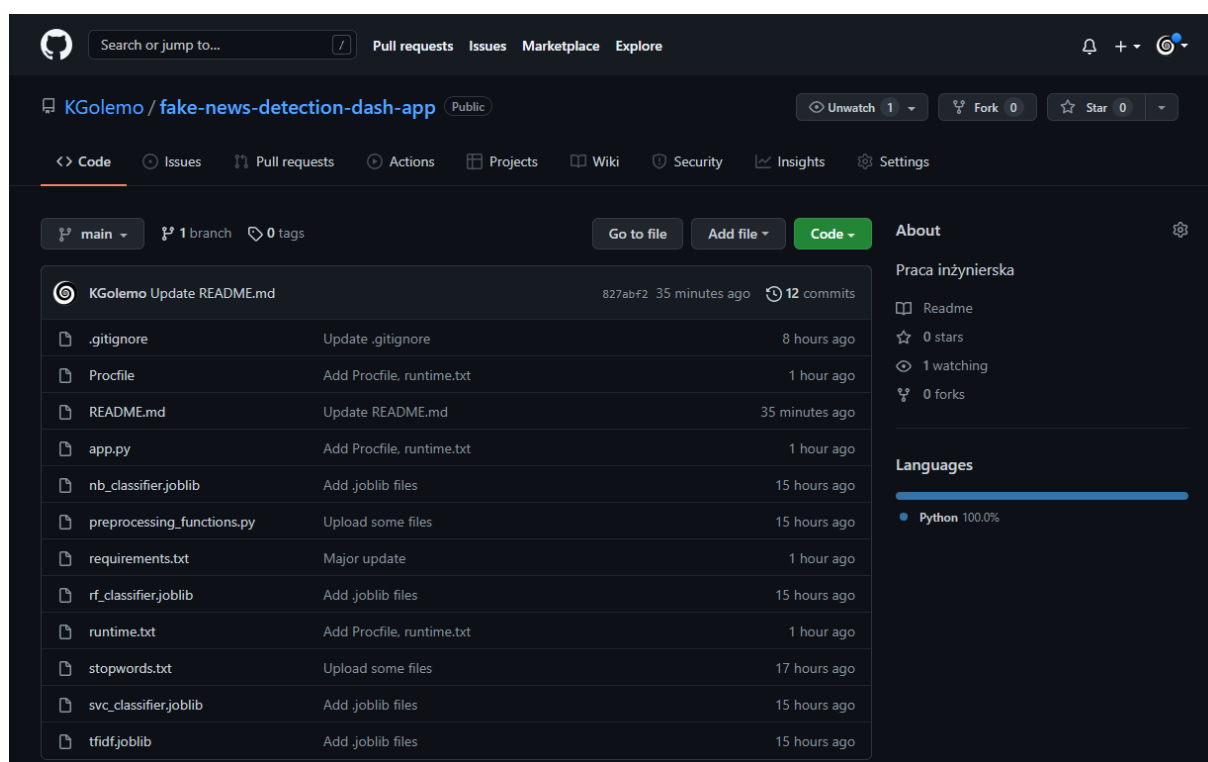
```
text (str): Extracted text.
```

► EXPAND SOURCE CODE

Rysunek 3.19. Fragment dokumentacji projektu z widocznym po lewej stronie panelem nawigacyjnym, umożliwiającym przemieszczanie się pomiędzy opisami wszystkich zaimplementowanych funkcji wykorzystywanych do tworzenia i przetwarzania zbioru danych

3.7. Tworzenie aplikacji webowej

W celu stworzenia aplikacji, umożliwiającej użytkownikowi zweryfikowanie wybranej wiadomości, konieczne było wyeksportowanie za pomocą narzędzia *Joblib* [51] obiektów związanych z ekstrakcją cech tekstu oraz wytrenowanych algorytmów klasyfikacji. Wszelkie pliki konieczne do działania aplikacji i modelu klasyfikatora zostały umieszczone w osobnym repozytorium [52], którego panel główny widoczny jest na Rysunku 3.20. Do zaprojektowania interfejsu wykorzystano pakiet *Dash* [53], umożliwiający tworzenie prostych witryn internetowych za pomocą języka Python, a w celu opublikowania aplikacji skorzystano z bezpłatnej platformy chmurowej *Heroku* [54].

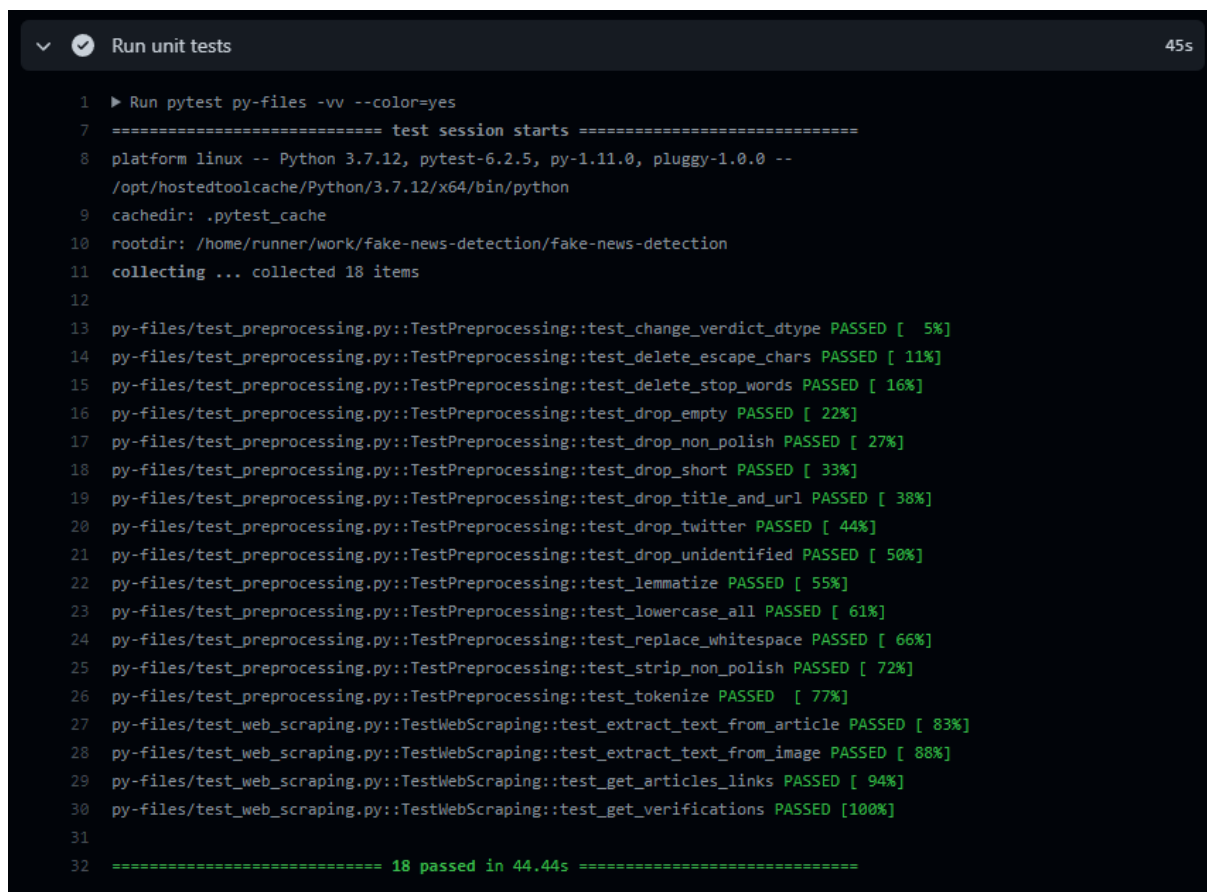


Rysunek 3.20. Panel główny repozytorium aplikacji z widocznymi m.in. plikami *.joblib* zawierającymi obiekty związane z modelem klasyfikatora

4. Rezultaty

4.1. Testy jednostkowe funkcji

Stworzone oprogramowanie zostało pokryte testami jednostkowymi *unittest* [55], weryfikującymi działanie wszystkich zaimplementowanych funkcji odpowiadających za przetwarzanie zbioru danych (Rozdział 3.2). Wykorzystano także mechanizm ciągłej integracji (ang. *Continuous Integration, CI*), polegający na weryfikowaniu za pomocą testów każdej aktualizacji projektu w repozytorium. Mechanizm ten zaimplementowano za pomocą dostępnych GitHub Actions, które korzystając z zamieszczonego w katalogu *workflows* pliku, przeprowadzają określone w nim operacje związane z ciągłą integracją. Rezultaty przykładowej weryfikacji przedstawiono na Rysunku 4.1.



```
Run unit tests 45s

1 ▶ Run pytest py-files -vv --color=yes
7 ===== test session starts =====
8 platform linux -- Python 3.7.12, pytest-6.2.5, py-1.11.0, pluggy-1.0.0 --
  /opt/hostedtoolcache/Python/3.7.12/x64/bin/python
9 cachedir: .pytest_cache
10 rootdir: /home/runner/work/fake-news-detection/fake-news-detection
11 collecting ... collected 18 items
12
13 py-files/test_preprocessing.py::TestPreprocessing::test_change_verdict_dtype PASSED [ 5%]
14 py-files/test_preprocessing.py::TestPreprocessing::test_delete_escape_chars PASSED [ 11%]
15 py-files/test_preprocessing.py::TestPreprocessing::test_delete_stop_words PASSED [ 16%]
16 py-files/test_preprocessing.py::TestPreprocessing::test_drop_empty PASSED [ 22%]
17 py-files/test_preprocessing.py::TestPreprocessing::test_drop_non_polish PASSED [ 27%]
18 py-files/test_preprocessing.py::TestPreprocessing::test_drop_short PASSED [ 33%]
19 py-files/test_preprocessing.py::TestPreprocessing::test_drop_title_and_url PASSED [ 38%]
20 py-files/test_preprocessing.py::TestPreprocessing::test_drop_twitter PASSED [ 44%]
21 py-files/test_preprocessing.py::TestPreprocessing::test_drop_unidentified PASSED [ 50%]
22 py-files/test_preprocessing.py::TestPreprocessing::test_lemmatize PASSED [ 55%]
23 py-files/test_preprocessing.py::TestPreprocessing::test_lowercase_all PASSED [ 61%]
24 py-files/test_preprocessing.py::TestPreprocessing::test_replace_whitespace PASSED [ 66%]
25 py-files/test_preprocessing.py::TestPreprocessing::test_strip_non_polish PASSED [ 72%]
26 py-files/test_preprocessing.py::TestPreprocessing::test_tokenize PASSED [ 77%]
27 py-files/test_web_scraping.py::TestWebScraping::test_extract_text_from_article PASSED [ 83%]
28 py-files/test_web_scraping.py::TestWebScraping::test_extract_text_from_image PASSED [ 88%]
29 py-files/test_web_scraping.py::TestWebScraping::test_get_articles_links PASSED [ 94%]
30 py-files/test_web_scraping.py::TestWebScraping::test_get_verifications PASSED [100%]
31
32 ===== 18 passed in 44.44s =====
```

Rysunek 4.1. Rezultat przykładowej weryfikacji za pomocą testów jednostkowych przeprowadzonej przez GitHub Actions w ramach mechanizmu ciągłej integracji

4.2. Ewaluacja algorytmów klasyfikacji

4.2.1. Zastosowane metryki

W celu ewaluacji klasyfikacji przeprowadzonych z użyciem algorytmów wymienionych w Rozdziale 3.3, wykorzystano następujące metryki z biblioteki *scikit-learn* [56]:

- precyzję (ang. *precision*),
- czułość (ang. *recall*),
- współczynnik F_1 -score,
- dokładność (ang. *accuracy*),
- pole pod krzywą ROC (ang. *Receiver Operating Characteristic*).

Zgodnie z [57] w ewaluacji binarnych klasyfikacji wyróżnia się cztery rodzaje predykcji:

- przypadki prawdziwie pozytywne (ang. *True Positive, TP*) – poprawne klasyfikacje pozytywnych przypadków (prawdziwe teksty zidentyfikowane jako prawdziwe),
- przypadki prawdziwie negatywne (ang. *True Negative, TN*) – poprawne klasyfikacje negatywnych przypadków (fałszywe teksty zidentyfikowane jako fałszywe),
- przypadki fałszywie pozytywne (ang. *False Positive, FP*) – niepoprawne klasyfikacje negatywnych przypadków (fałszywe teksty zidentyfikowane jako prawdziwe),
- przypadki fałszywie negatywne (ang. *False Negative, FN*) – niepoprawne klasyfikacje pozytywnych przypadków (prawdziwe teksty zidentyfikowane jako fałszywe).

Precyzja określa jaką część wszystkich pozytywnych przypadków stanowią predykcje prawdziwie pozytywne [56]:

$$\text{precyzja} = \frac{TP}{TP + FP}. \quad (4.1.1)$$

Czułość stanowi stosunek prawdziwie pozytywnych predykcji do wszystkich pozytywnych przypadków [56]:

$$\text{czułość} = \frac{TP}{TP + FN}. \quad (4.1.2)$$

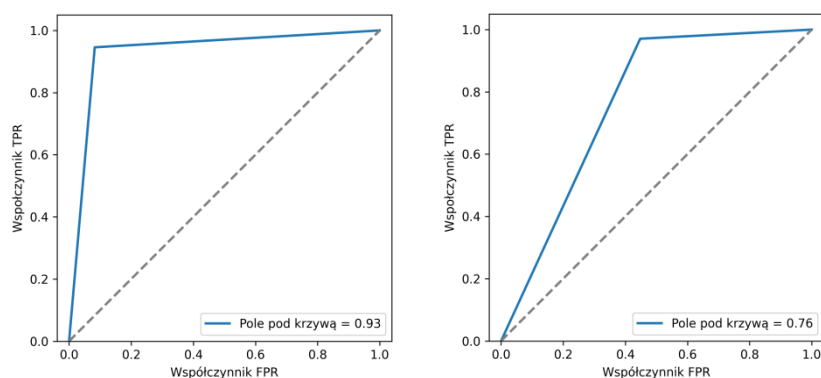
F_1 -score stanowi harmoniczną średnią precyzji i czułości [56]:

$$F_1\text{-score} = \frac{\text{precyzja} \cdot \text{czułość}}{\text{precyzja} + \text{czułość}}. \quad (4.1.3)$$

Dokładność określa jaką część wszystkich predykcji stanowią poprawne klasyfikacje [56]:

$$\text{dokładność} = \frac{TP + TN}{TP + TN + FP + FN}. \quad (4.1.4)$$

Krzywa ROC obrazuje zależność między tzw. współczynnikiem TPR (ang. *True Positive Rate*) – stosunkiem predykcji prawdziwie pozytywnych do wszystkich pozytywnych przypadków, a tzw. współczynnikiem FPR (ang. *False Positive Rate*) – stosunkiem predykcji fałszywie pozytywnych do wszystkich negatywnych przypadków. Jeżeli konieczne jest ocenienie jakości klasyfikacji za pomocą jednej metryki, pole pod tą krzywą jest jedną z najlepszych możliwości, ponieważ wskazuje ono jak dobrze klasy są separowane [57]. Na Rysunku 4.2 przedstawiono przykładowe krzywe ROC wraz z obliczonym polem pod nimi.



Rysunek 4.2. Przykładowe krzywe ROC z obliczonymi pod nimi polami

4.2.2. Naiwny klasyfikator bayesowski

Tabela 4.1. Rezultaty ewaluacji klasyfikacji algorytmem NB dla unigramów, przeprowadzonej przy pomocy wartości precyzji [%], czułości [%], współczynnika F_1 -score [%] i dokładności [%] oraz wielkości pola pod krzywą ROC, z rozróżnieniem liczby cech i wartości współczynnika α algorytmu. Najlepsze wyniki uzyskano przy ograniczeniu liczby cech do 10 tysięcy

| Liczba cech | α | Precyzja [%] | Czułość [%] | Współczynnik F_1 -score [%] | Dokładność [%] | Pole pod krzywą ROC |
|-------------|----------|--------------|-------------|-------------------------------|----------------|---------------------|
| 1000 | 0,1 | 85 | 96 | 90 | 87 | 0,82 |
| 3000 | 0,1 | 88 | 96 | 92 | 90 | 0,86 |
| 10000 | 0,03 | 93 | 95 | 94 | 92 | 0,90 |
| 27416 | 0,01 | 88 | 96 | 92 | 89 | 0,86 |

Tabela 4.2. Rezultaty ewaluacji klasyfikacji algorytmem NB dla bigramów, przeprowadzonej przy pomocy wartości precyzji [%], czułości [%], współczynnika F_1 -score [%] i dokładności [%] oraz wielkości pola pod krzywą ROC, z rozróżnieniem liczby cech i wartości współczynnika α algorytmu. Najlepsze wyniki uzyskano przy ograniczeniu liczby cech 10 tysięcy

| Liczba cech | α | Precyzja [%] | Czułość [%] | Współczynnik F_1 -score [%] | Dokładność [%] | Pole pod krzywą ROC |
|-------------|----------|--------------|-------------|-------------------------------|----------------|---------------------|
| 1000 | 0,1 | 79 | 97 | 87 | 81 | 0,74 |
| 3000 | 0,1 | 80 | 97 | 88 | 83 | 0,76 |
| 10000 | 0,03 | 81 | 97 | 88 | 83 | 0,77 |
| 30000 | 0,01 | 81 | 97 | 88 | 83 | 0,76 |
| 100000 | 0,001 | 80 | 97 | 88 | 82 | 0,75 |
| 193552 | 0,3 | 76 | 98 | 86 | 78 | 0,69 |

W przypadku unigramów (Tabela 4.1) największą wartość współczynnika F_1 -score, dokładności i pola pod krzywą ROC uzyskano przy ograniczeniu liczby cech do 10 tysięcy.

W przypadku bigramów (Tabela 4.2) największą wartość współczynnika F_1 -score osiągnięto przy ograniczeniu liczby cech do 3 tysięcy, 10 tysięcy, 30 tysięcy i 100 tysięcy, największą dokładność przy ograniczeniu do 3, 10 i 30 tysięcy, a największe pole pod krzywą ROC przy ograniczeniu do 10 tysięcy.

Ogółem dla naiwnego klasyfikatora bayesowskiego najlepsze rezultaty uzyskano przy ograniczeniu liczby cech do 10 tysięcy w przestrzeni opartej o unigramy. Dodatkowa analiza bigramów nie poprawiła jakości klasyfikacji, zmniejszając wartość pola pod krzywą ROC o ok. 17%.

4.2.3. Liniowa maszyna wektorów nośnych

Tabela 4.3. Rezultaty ewaluacji klasyfikacji algorytmem LSVM dla unigramów, przeprowadzonej przy pomocy wartości precyzji [%], czułości [%], współczynnika F_1 -score [%] i dokładności [%] oraz wielkości pola pod krzywą ROC, z rozróżnieniem liczby cech i wartości stałej C algorytmu. Najlepsze wyniki uzyskano przy ograniczeniu liczby cech 10 tysięcy

| Liczba cech | C | Precyzja [%] | Czułość [%] | Współczynnik F_1 -score [%] | Dokładność [%] | Pole pod krzywą ROC |
|-------------|-----|--------------|-------------|-------------------------------|----------------|---------------------|
| 1000 | 0,1 | 91 | 94 | 93 | 90 | 0,88 |
| 3000 | 0,3 | 94 | 94 | 94 | 92 | 0,91 |
| 10000 | 0,3 | 96 | 94 | 95 | 94 | 0,94 |
| 27416 | 1 | 96 | 94 | 95 | 94 | 0,93 |

Tabela 4.4. Rezultaty ewaluacji klasyfikacji algorytmem LSVM dla bigramów, przeprowadzonej przy pomocy wartości precyzji [%], czułości [%], współczynnika F_1 -score [%] i dokładności [%] oraz wielkości pola pod krzywą ROC, z rozróżnieniem liczby cech i wartości stałej C algorytmu. Najlepsze wyniki uzyskano przy ograniczeniu liczby cech 10 tysięcy

| Liczba cech | C | Precyzja [%] | Czułość [%] | Współczynnik F_1 -score [%] | Dokładność [%] | Pole pod krzywą ROC |
|-------------|-----|--------------|-------------|-------------------------------|----------------|---------------------|
| 1000 | 0,3 | 90 | 92 | 91 | 88 | 0,86 |
| 3000 | 0,3 | 90 | 93 | 91 | 88 | 0,85 |
| 10000 | 1 | 91 | 94 | 92 | 90 | 0,88 |
| 30000 | 1 | 89 | 95 | 92 | 89 | 0,86 |
| 100000 | 1 | 85 | 96 | 91 | 87 | 0,82 |
| 193552 | 1 | 81 | 97 | 88 | 83 | 0,76 |

W przypadku unigramów (Tabela 4.3) największe wartości współczynnika F_1 -score i dokładności osiągnięto przy ograniczeniu liczby cech do 10 tysięcy i pozostawieniu ich domyślnej liczby, a największe pole pod krzywą ROC przy ograniczeniu liczby cech do 10 tysięcy.

W przypadku bigramów (Tabela 4.4) największą wartość współczynnika F_1 -score osiągnięto przy ograniczeniu liczby cech do 10 i 30 tysięcy, a największą dokładność i pole pod krzywą ROC przy ograniczeniu do 10 tysięcy.

Ogółem dla maszyny wektorów nośnych najlepsze rezultaty uzyskano przy ograniczeniu liczby cech do 10 tysięcy w przestrzeni opartej o unigramy. Dodatkowa analiza bigramów nie poprawiła jakości klasyfikacji, zmniejszając wartość pola pod krzywą ROC o ok. 7%.

4.2.4. Las losowy

Tabela 4.5. Rezultaty ewaluacji klasyfikacji algorytmem RF dla unigramów, przeprowadzonej przy pomocy precyzji wartości [%], czułości [%], współczynnika F_1 -score [%] i dokładności [%] oraz wielkości pola pod krzywą ROC, z rozróżnieniem liczby cech i wartości parametru n algorytmu. Najlepsze wyniki uzyskano przy ograniczeniu liczby cech do 3 i 10 tysięcy

| Liczba cech | n | Precyzja [%] | Czułość [%] | Współczynnik F_1 -score [%] | Dokładność [%] | Pole pod krzywą ROC |
|-------------|------|--------------|-------------|-------------------------------|----------------|---------------------|
| 1000 | 1000 | 91 | 92 | 92 | 89 | 0,87 |
| 3000 | 1000 | 91 | 93 | 92 | 89 | 0,88 |
| 10000 | 300 | 91 | 93 | 92 | 89 | 0,88 |
| 27416 | 300 | 91 | 93 | 92 | 89 | 0,87 |

Tabela 4.6. Rezultaty ewaluacji klasyfikacji algorytmem RF dla bigramów, przeprowadzonej przy pomocy wartości precyzji [%], czułości [%], współczynnika F_1 -score [%] i dokładności [%] oraz wielkości pola pod krzywą ROC, z rozróżnieniem liczby cech i wartości parametru n algorytmu. Najlepsze wyniki uzyskano przy ograniczeniu liczby cech 10 tysięcy

| Liczba cech | n | Precyzja [%] | Czułość [%] | Współczynnik F_1 -score [%] | Dokładność [%] | Pole pod krzywą ROC |
|-------------|------|--------------|-------------|-------------------------------|----------------|---------------------|
| 1000 | 1000 | 93 | 88 | 90 | 88 | 0,87 |
| 3000 | 300 | 94 | 87 | 90 | 88 | 0,88 |
| 10000 | 1000 | 95 | 88 | 91 | 89 | 0,90 |
| 30000 | 100 | 97 | 85 | 91 | 88 | 0,90 |
| 100000 | 300 | 99 | 83 | 91 | 89 | 0,91 |
| 193552 | 100 | 98 | 86 | 92 | 90 | 0,91 |

W przypadku unigramów (Tabela 4.5) współczynnik F_1 -score i dokładność przyjmowały te same wartości dla każdej liczby cech, a największe pole pod krzywą ROC uzyskano przy ograniczeniu liczby cech do 3 i 10 tysięcy.

W przypadku bigramów (Tabela 4.6) największe wartości współczynnika F_1 -score i dokładności osiągnięto dla domyślnej liczby cech, a największe pole pod krzywą ROC przy ograniczeniu do 100 tysięcy i pozostawieniu ich domyślnej liczby.

Ogółem dla lasu losowego najlepsze rezultaty uzyskano dla pełnego zbioru cech w przestrzeni opartej o bigramy. Dodatkowa analiza bigramów nieznacznie poprawiła jakość klasyfikacji, zwiększając wartość pola pod krzywą ROC o ok. 3%.

4.3. Aplikacja webowa

Stworzona aplikacja [58] umożliwia użytkownikowi zweryfikowanie wybranej wiadomości poprzez wklejenie jej treści do przeznaczonego do tego pola, a następnie wybranie przycisku rozpoczynającego proces. Elementy te przedstawiono na Rysunku 4.3.

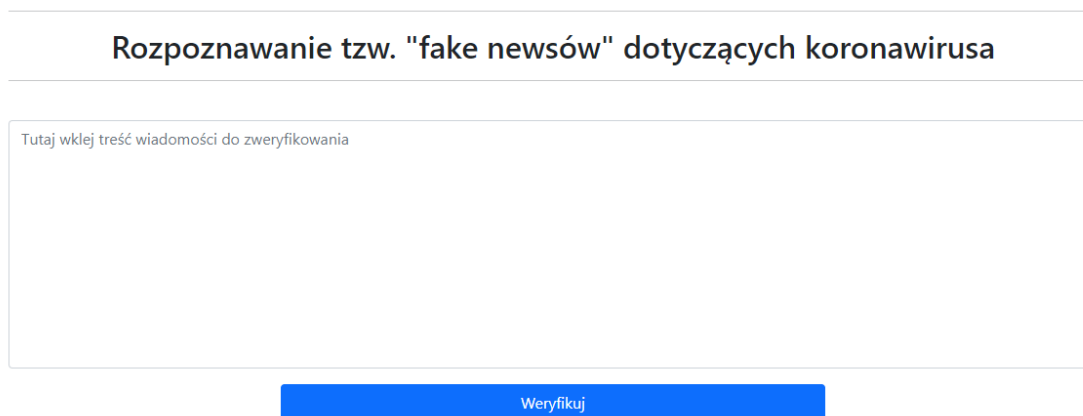
Po uruchomieniu analizy przeprowadzane są następujące operacje:

- 1) Oczyszczenie tekstu za pomocą funkcji przetwarzających dane (Rozdział 3.2).
- 2) Przeniesienie oczyszczonego tekstu do przestrzeni cech (Rozdział 3.2) za pomocą obiektu wektoryzującego odtworzonego z pliku *.joblib*.
- 3) Wyznaczenie predykcji za pomocą wytrenowanych algorytmów klasyfikacyjnych (Rozdział 3.3) odtworzonych z pliku *.joblib*.

Końcowy werdykt wyznaczany jest w następujący sposób:

- informacja jest uznawana za fałszywą, jeśli przynajmniej jeden z klasyfikatorów uznał ją za fałszywą,
- informacja jest uznawana za prawdziwą, jeśli wszystkie klasyfikatory uznały ją za prawdziwą.

Raport z analizy zawiera dodatkowo informacje o wynikach predykcji poszczególnych algorytmów. Przykładowe weryfikacje zostały przedstawione na Rysunkach 4.4 i 4.5.



Rozpoznawanie tzw. "fake newsów" dotyczących koronawirusa

Tutaj wklej treść wiadomości do zweryfikowania

Weryfikuj

Rysunek 4.3. Domyślny wygląd stworzonej aplikacji z widocznym polem do wklejenia treści wiadomości oraz przyciskiem rozpoczynającym proces weryfikacji

Rozpoznawanie tzw. "fake newsów" dotyczących koronawirusa

Norwegia była do niedawna rajem nie tylko dla Polaków szukających godziwego zarobku, ale i oazą wolności w czasach kowidyzmu. Jednakże sataneria dobiera taktyki do mentalności konkretnego narodu. W związku z tym, że Norwedzy są posłuszni rządowi z natury, to ze szczepieniami nie było problemu. Ponad 70% Norwegów uwierzyło propagandzie. Jaki jest tego skutek? Teraz już nie mogą zganiać na antyszczepionkowców przy takim wskaźniku zakażeń. Jest to klęska idei budowania odporności produktem mRNA. Brawo Polonia w Norwegii. Polonia nie zadżgana. Grypa Hiszpanka wraca tym razem nazywana ponownie niesłusznie bo wirusem z Wuhan. Podobnie jak 100 lat temu od dżgania umiera masa ludzi.

Weryfikuj

Werdykt:

To prawdopodobnie fałszywa informacja

Predykcje poszczególnych klasyfikatorów (True - wiadomość prawdziwa, False - fałszywa):

| Algorytm | Predykcja |
|----------------------------------|-----------|
| Naiwny klasyfikator bayesowski | False |
| Liniowa maszyna wektorów nośnych | False |
| Las losowy | False |

Rysunek 4.4. Rezultat weryfikacji fałszywej informacji w stworzonej aplikacji

Rozpoznawanie tzw. "fake newsów" dotyczących koronawirusa

O odkryciu poinformował Leondios Kostrikis, profesor nauk biologicznych na Uniwersytecie Cypryjskim. Zespół badaczy pod kierunkiem Kostrikisa nazwał ten szczep deltakronem ze względu na jego sygnatury genetyczne, podobne do wariantów omikron i delta. Jak podaje laboratorium Kostrikisa, jego zespół odkrył do tej pory 25 przypadków zakażenia nowym wariantem. Badacze podkreślają, że jest jeszcze za wcześnie, aby stwierdzić, jakie mogą być skutki takich zakażeń. - W przyszłości zobaczymy, czy ten szczep jest bardziej patologiczny lub bardziej zaraźliwy i czy wypchnie dominujące obecnie deltę i omikrona - powiedział Kostrikis w wywiadzie dla telewizji Sigma. Według Bloombera naukowcy przesłali już swoje odkrycie do GISAI, międzynarodowej bazy danych, która śledzi wirusy.

Weryfikuj

Werdykt:

To prawdopodobnie prawdziwa informacja

Predykcje poszczególnych klasyfikatorów (True - wiadomość prawdziwa, False - fałszywa):

| Algorytm | Predykcja |
|----------------------------------|-----------|
| Naiwny klasyfikator bayesowski | True |
| Liniowa maszyna wektorów nośnych | True |
| Las losowy | True |

Rysunek 4.5. Rezultat weryfikacji prawdziwej informacji w stworzonej aplikacji

5. Podsumowanie i wnioski

5.1. Analiza wyników klasyfikacji

Poniżej zamieszczono najlepsze pod względem zastosowanych metryk wyniki uzyskane przez wszystkie ewaluowane algorytmy dla uni- i bigramowych przestrzeni cech (Rozdział 4.2). Porównano wartości współczynników F_1 -score, poziomy dokładności oraz pola pod krzywymi ROC.

Tabela 5.1. Rezultaty ewaluacji klasyfikacji dla unigramów, przeprowadzonej przy pomocy wartości współczynnika F_1 -score [%] i dokładności [%] oraz wielkości pola pod krzywą ROC, z rozróżnieniem liczby cech i rodzaju zastosowanego algorytmu. Najlepsze wyniki uzyskano dla klasyfikatora LSVM

| Algorytm klasyfikacji | Liczba cech | Współczynnik F_1 -score [%] | Dokładność [%] | Pole pod krzywą ROC |
|-----------------------|-------------|-------------------------------|----------------|---------------------|
| NB | 10000 | 94 | 92 | 0,90 |
| LSVM | 10000 | 95 | 94 | 0,94 |
| RF | 10000 | 92 | 89 | 0,88 |

Tabela 5.2. Rezultaty ewaluacji klasyfikacji dla bigramów, przeprowadzonej przy pomocy wartości współczynnika F_1 -score [%] i dokładności [%] oraz wielkości pola pod krzywą ROC, z rozróżnieniem liczby cech i rodzaju zastosowanego algorytmu. Najlepsze wyniki uzyskano dla klasyfikatora RF

| Algorytm klasyfikacji | Liczba cech | Współczynnik F_1 -score [%] | Dokładność [%] | Pole pod krzywą ROC |
|-----------------------|-------------|-------------------------------|----------------|---------------------|
| NB | 10000 | 88 | 83 | 0,77 |
| LSVM | 10000 | 92 | 90 | 0,88 |
| RF | 193552 | 92 | 90 | 0,91 |

W przypadku unigramów (Tabela 5.1) dla wszystkich algorytmów najlepsze rezultaty uzyskano przy ograniczeniu liczby cech do 10 tysięcy. Największe wartości współczynnika F_1 -score, dokładności oraz pola pod krzywą ROC otrzymano dla klasyfikatora LSVM (kolejno 95%, 94% i 0,94), a najmniejsze dla RF (kolejno 92%, 89%, 0,88).

W przypadku bigramów (Tabela 5.2) dla algorytmów NB i LSVM najlepsze rezultaty uzyskano przy ograniczeniu liczby cech do 10 tysięcy, a dla klasyfikatora RF przy ich pełnym zbiorze. Największe wartości współczynnika F_1 -score, dokładności oraz pola pod krzywą ROC otrzymano dla klasyfikatora RF (kolejno 92%, 90% i 0,91), a najmniejsze dla NB (kolejno 88%, 83%, 0,77).

Ogółem najlepsze rezultaty uzyskano przy zastosowaniu ograniczonej do 10 tysięcy cech przestrzeni unigramów i algorytmu LSVM – wartości współczynnika F_1 -score, dokładności oraz pola pod krzywą ROC wyniosły odpowiednio 95%, 94% i 0,94.

5.2. Wnioski

Przeprowadzone badanie literaturowe umożliwiło wybór metod ekstrakcji cech tekstu oraz algorytmów odpowiednich do tego typu problemu klasyfikacji. Zaproponowany sposób akwizycji danych pozwolił na stworzenie właściwego korpusu tekstów, który został użyty w modelu klasyfikatora. Stworzone oprogramowanie rozpoznaje „fake newsy” dotyczące koronawirusa, a aplikacja umożliwia użytkownikowi zweryfikowanie wybranej wiadomości.

5.3. Kierunki dalszego rozwoju

W zbiorze danych, stworzonym z wykorzystaniem zaproponowanych technik, artykuły stanowiące „fake newsy” charakteryzują się mniejszą, wynoszącą najczęściej kilkaset znaków, długością względem prawdziwych wiadomości, których zdecydowana większość liczy ponad tysiąc znaków. Wynika to z faktu, że algorytm pobierania fałszywych artykułów dokonuje ekstrakcji ich treści ze zdjęcia, co uniemożliwia wprowadzenie do korpusu dłuższych tekstów przekazujących nieprawdziwe informacje (w przypadku takich tekstów zdjęcie zawiera najczęściej jedynie krótki fragment). Potencjalny kierunek dalszego rozwoju stanowi więc rozszerzenie zbioru danych poprzez pobranie fałszywych wiadomości z większej liczby serwisów weryfikujących informacje, co zrekompensowałoby mniejszą długość tych tekstów, poprzez powiększenie ich liczebności. Znacznie bardziej czasochłonną alternatywą jest ręczne przetwarzanie kodu stron wskazanych jako źródła „fake newsów”, które umożliwiłoby pobranie pełnych tekstów, a nie jedynie fragmentów, a także uniezależnienie się od jakości zdjęć artykułów, czy doboru kadru. Inny kierunek stanowi zastosowanie złożonych metod ekstrakcji i selekcji cech, czy zastosowanie bardziej zaawansowanych klasyfikatorów, np. konwulucyjnych sieci neuronowych.

6. Bibliografia

1. Gelfert A. Fake News: A Definition. *Informal Logic*. 2018 marzec: 84-117.
2. Molina MD, Sundar SS, Quang Le T, Lee D. “Fake News” Is Not Simply False Information: A Concept Explication and Taxonomy of Online Content. *American Behavioral Scientist*. 2019 październik.
3. FakeHunter. [Online]. [Dostęp 1 stycznia 2022]. <https://fakehunter.pap.pl>.
4. FakeNews.pl. [Online]. [Dostęp 1 stycznia 2022]. <https://fakenews.pl/>.
5. AntyFAKE. [Online]. [Dostęp 1 stycznia 2022]. <https://www.antyfake.pl/>.
6. Demagog. [Online]. [Dostęp 1 stycznia 2022]. <https://demagog.org.pl/>.
7. Konkret24. [Online]. [Dostęp 1 stycznia 2022]. <https://konkret24.tvn24.pl/>.
8. Molnar C. Interpretable machine learning: lulu.com; 2020.
9. Chowdhury G. Natural language processing. *Annual Review of Information Science and Technology*. 2005 styczeń: 51-89.
10. Khurana D, Koli A, Khatter K, Singh S. Natural Language Processing: State of Art, Current Trends and Challenges. 2017.
11. Jurafsky D, Martin J. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2nd ed.: Prentice Hall; 2008.
12. Allahyari M, Pouriyeh S, Assefi M, Safaei S. A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques. 2017 lipiec.
13. Granik M, Mesyura V. Fake news detection using naive Bayes classifier. [W:] *IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*; 2017; Kijów. 900-903.
14. Ahmed H, Traore I, Saad S. Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. [W:] *International Conference on Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments*; 2017; Vancouver. 127-138.
15. Reis JCS, Correia A, Murai F, Veloso A, Benevenuto F. Supervised Learning for Fake News Detection. *IEEE Intelligent Systems*. 2019 marzec-kwiecień: 76-81.
16. Dokumentacja algorytmu XGBoost. [Online]. [Dostęp 5 stycznia 2022]. <https://xgboost.readthedocs.io/en/latest/index.html>.

17. Vogel I, Jiang P. Fake News Detection with the New German Dataset “GermanFakeNC”. [W:] Digital Libraries for Open Knowledge; 2019; Oslo. 288-295.
18. Albawi S, Mohammed TA, Al-Zawi S. Understanding of a convolutional neural network. [W:] International Conference on Engineering and Technology (ICET); 2017. 1-6.
19. Sidorov G, Moreno J, Adorno HG, Posadas-Durán JP. Detection of fake news in a new corpus for the Spanish language. *Journal of Intelligent & Fuzzy Systems*. 2019 maj; 36(5): p. 4869-4876.
20. Cavnar WB, Trenkle JM. N-gram-based text categorization. [W:] Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval; 1994.
21. Schapire RE. The Boosting Approach to Machine Learning: An Overview. In Denison DD, Holmes CC, Hansen MH, Mallick B, Yu B. *Nonlinear Estimation and Classification. Lecture Notes in Statistics*. Nowy Jork: Springer; 2003. p. 149-171.
22. van der Linden S, Roozenbeek J, Compton J. Inoculating Against Fake News About COVID-19. *Frontiers in Psychology*. 2020 październik.
23. Diouf R, Sall O, Birregah B, Bousso M, Sarr EN, Bousso M, et al. Web Scraping: State-of-the-Art and Areas of Application. [W:] IEEE International Conference on Big Data; 2019; Los Angeles. 6040-6042.
24. Dokumentacja biblioteki Requests. [Online]. [Dostęp 5 stycznia 2022].
25. Dokumentacja biblioteki BeautifulSoup. [Online]. [Dostęp 5 stycznia 2022].
<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>.
26. Raport 1 z portalu FakeHunter. [Online]. [Dostęp 3 stycznia 2022].
<https://fakehunter.pap.pl/raport/2143bb38-fa97-4a76-a9f1-40a322d5e29b>.
27. Dokumentacja biblioteki EasyOCR. [Online]. [Dostęp 5 stycznia 2022].
<https://jaided.ai/easyocr/documentation/>.
28. Raport 2 z portalu FakeHunter. [Online]. [Dostęp 5 stycznia 2022].
<https://fakehunter.pap.pl/raport/eafff780-4b0e-441a-8329-cc26845c5279>.
29. Raport 3 z portalu FakeHunter. [Online]. [Dostęp 5 stycznia 2022].
<https://fakehunter.pap.pl/raport/fa241a92-7327-49c6-8b63-621264a80956>.
30. Termedia. [Online]. [Dostęp 12 grudnia 2021]. <https://www.termedia.pl>.
31. Dokumentacja biblioteki polyglot. [Online]. [Dostęp 5 stycznia 2022].
<https://polyglot.readthedocs.io/en/latest/>.

-
32. Kieraś W, Woliński M. Morfeusz 2 – analizator i generator fleksyjny dla języka polskiego. *Język Polski*. 2017: 75-83.
 33. Wersja demonstracyjna programu Morfeusz2. [Online]. [Dostęp 5 stycznia 2022].
<http://morfeusz.sgjp.pl/demo/>.
 34. Dokumentacja metody TF-IDF (scikit-learn). [Online]. [Dostęp 5 stycznia 2022].
https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction.
 35. McCallum A, Nigam K. A Comparison of Event Models for Naive Bayes Text Classification. 2001 maj.
 36. Zhang H. The Optimality of Naive Bayes. [W:] *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*; 2004; Floryda.
 37. Raghavan P, Schütze H, Manning C. *Introduction to Information Retrieval* Cambridge: Cambridge University Press; 2009.
 38. Dokumentacja naiwnego klasyfikatora bayesowskiego (scikit-learn). [Online]. [Dostęp 18 grudnia 2021].
https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes.
 39. Boser B, Guyon I, Vapnik V. A Training Algorithm for Optimal Margin Classifier. *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*. 1992: 144-152.
 40. Cortes C, Vapnik V. Support-vector networks. *Machine Learning*. 1995: 273-297.
 41. Dokumentacja liniowej maszyny wektorów nośnych (scikit-learn). [Online]. [Dostęp 20 grudnia 2021]. <https://scikit-learn.org/stable/modules/svm.html#id15>.
 42. Cutler A, Cutler DR, Stevens JR. Random Forests. *Machine Learning*. 2011 styczeń: 157-176.
 43. Breiman L. Random Forests. *Machine Learning*. 2001 październik: 5-32.
 44. Dokumentacja algorytmu lasu losowego (scikit-learn). [Online]. [Dostęp 24 grudnia 2021]. <https://scikit-learn.org/stable/modules/ensemble.html#forest>.
 45. Dokumentacja metody grid search (scikit-learn). [Online]. [Dostęp 5 stycznia 2022].
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.
 46. Dokumentacja notatników Jupyter. [Online]. [Dostęp 8 stycznia 2022].
<https://docs.jupyter.org/en/latest/>.

-
47. Repozytorium GitHub projektu. [Online]. [Dostęp 8 stycznia 2022].
<https://github.com/KAIR-ISZ-NLP/fake-news-detection>.
 48. Dokumentacja projektu. [Online]. [Dostęp 8 stycznia 2022].
<https://kair-isz-nlp.github.io/fake-news-detection/>.
 49. Dokumentacja generatora pdoc3. [Online]. [Dostęp 8 stycznia 2022].
<https://pdoc3.github.io/pdoc/>.
 50. Dokumentacja PEP 257. [Online]. [Dostęp 8 stycznia 2022].
<https://www.python.org/dev/peps/pep-0257/>.
 51. Dokumentacja pakietu Joblib. [Online]. [Dostęp 8 stycznia 2022].
<https://joblib.readthedocs.io/en/latest/>.
 52. Repozytorium aplikacji. [Online]. [Dostęp 8 stycznia 2022].
<https://github.com/KGolemo/fake-news-detection-dash-app>.
 53. Dokumentacja pakietu Dash. [Online]. [Dostęp 8 stycznia 2022]. <https://dash.plotly.com/>.
 54. Dokumentacja platformy Heroku. [Online]. [Dostęp 8 stycznia 2022].
<https://devcenter.heroku.com/categories/reference>.
 55. Dokumentacja unittest. [Online]. [Dostęp 8 stycznia 2022].
<https://docs.python.org/3/library/unittest.html>.
 56. Dokumentacja metryk (scikit-learn). [Online]. [Dostęp 5 stycznia 2022].
https://scikit-learn.org/stable/modules/model_evaluation.html.
 57. Bradley AP. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern recognition. 1997: 1145-1159.
 58. Aplikacja. [Online]. [Dostęp 8 stycznia 2022].
<https://fake-news-detection-dash-app.herokuapp.com/>.