AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

INSTYTUT INFORMATYKI

PRACA DYPLOMOWA MAGISTERSKA

Analiza fałszywych wiadomości

Analysis of fake news

Autor: *Piotr Kardaś*
Kierunek studiów: *Informatyka*
Typ studiów: *Stacjonarne*
Opiekun pracy: *dr inż. Anna Zygmunt*

Kraków, 2021

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście, samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

.....

PODPIS

Spis treści

1. Wstęp

Fałszywe wiadomości od zawsze stanowiły duże zagrożenie. Fałszywe informacje [28] (ang. *fake news*) to wiadomości, które przekazują nieprawdziwe lub przeinaczone dane. Celem jest wprowadzenie w błąd, zaszokowanie lub wzbudzenie kontrowersji. Aktualnie fałszywe wiadomości rozpowszechniają się głównie za pomocą mediów społecznościowych, a motywacją do ich tworzenia jest chęć osiągnięcia celów politycznych, manipulacja opinią publiczną czy chęć szybkiego zysku z reklam na stronie internetowej (ang. *clickbait*).

Istnieje kilka metod automatycznego wykrywania takich wiadomości: w oparciu o analizę stylu wypowiedzi, nacechowania, propagacji czy weryfikację prawdziwości informacji. Celem pracy była analiza tych metod i skupienie się na analizie tekstu takich wiadomości przy użyciu technik przetwarzania języka naturalnego, jako podejścia dającego najlepsze rezultaty w klasyfikacji. Druga część pracy skupiła się na sprawdzeniu jak wygląda propagacja fałszywych wiadomości przez porównanie ich z rozprzestrzenianiem się epidemii. Przyrównanie rozprzestrzeniania fałszywych informacji do epidemii było eksperymentem myślowym, który również może posłużyć do wykrywania i eliminowania źródeł, które masowo propagują nieprawdziwe wiadomości.

1.1. Motywacja

Wraz ze wzrostem popularności mediów społecznościowych, fałszywe wiadomości są coraz powszechniejszym zjawiskiem. Łatwość ich tworzenia i rozpowszechniania stwarza realne zagrożenia, takie jak wzbudzanie paniki czy wpływanie na wyniki demokratycznych wyborów. Rozpowszechnianie fałszywych wiadomości może być także częścią wojny informacyjnej, w której obce siły próbują manipulować opinią publiczną przez sianie propagandy i dezinformacji.

Aktualnie, najważniejszą metodą walki z fałszywymi informacjami wciąż pozostają organizacje, które weryfikują wiadomości za pomocą metod manualnych. Takie rozwiązanie choć gwarantuje całkowitą przejrzystość i wysoką dokładność, to nie jest w stanie efektywnie działać w środowiskach z dużym przepływem informacji. Innymi słowy, podejście oparte o ręczną weryfikację nie jest skalowalne.

Na podstawie powodów przytoczonych powyżej, istnieje konieczność budowy nowych metod automatycznych do wykrywania fałszywych informacji. Fałszywe informacje stanowią palący problem w wielu mediach społecznościowych, a obecne mechanizmy w nich stosowane nie są wystarczające.

1.2. Zawartość pracy

W tej części, w punktach została wymieniona zawartość pracy.

1. Opis problemu, zagrożeń jakie niosą fałszywe informacje i ich wpływ na społeczeństwo. Uzasadnienie konieczności budowy nowych rozwiązań.

2. Podsumowanie istniejących osiągnięć w dziedzinie wykrywania fałszywych informacji. Opis kilku klasyfikatorów, które zostały użyte podczas rozwoju pracy, opis metryk używanych do walidacji jakości klasyfikacji. Przegląd istniejących modeli epidemiologicznych oraz omówienie współczynnika R_0. Uzasadnienie potrzeby dalszych badań.

3. Opis przyjętego podejścia w problemie klasyfikacji, opis zaproponowanych cech tekstu. Opis podejścia do analizy propagacji.

4. Opis architektury, zastosowanych mechanizmów, bibliotek, zachowania programu. Wyjaśnienie działania ekstrakcji cech. Przepływ tekstu przez program. Wytłumaczenie działania analizy propagacji.

5. Opis wykorzystanych zbiorów danych. Wyniki klasyfikacji przy użyciu metryk. Wybór najlepszych klasyfikatorów. Porównanie wyników do istniejących rozwiązań. Eksperymenty dotyczące liczby oraz sensu zastosowania określonej liczby bigramów. Badanie propagacji przez analizę współczynnika R_0, analizę propagacji w czasie i analizę propagacji w przestrzeni. Porównanie do epidemii.

6. Podsumowanie wyników. Propozycja dalszych prac. Odnośnik do kodu źródłowego.

2. Stan badań

W tym rozdziale omówione zostały możliwe podejścia do analizy fałszywych wiadomości [31]. Opisane zostały także algorytmy klasyfikacji wraz z metrykami jakie mogą zostać użyte do badania jakości klasyfikatora. Podsumowany został także model rozprzestrzeniania się fałszywych wiadomości autorstwa Daley'a i Kendall'a, oraz omówiony został współczynnik R_0.

2.1. Manualne wykrywanie fałszywych wiadomości

Podejście zakładające wykorzystanie zdolności ludzi można podzielić na dwie kategorie [30] - weryfikację przez ekspertów oraz na weryfikację przez wykorzystanie zbiorowej inteligencji tłumu.

2.1.1. Weryfikacja przez ekspertów

Istnieją organizacje, które zajmują się weryfikacją fałszywych informacji, na przykład Demagog czy FakeNews.PL. Organizacje te posiadają zespoły pracowników, którzy zajmują się weryfikacją popularnych wiadomości. Praca fact-checkera (osoby weryfikującej) najczęściej polega na przeglądaniu podobnych artykułów, odnośników z głównego artykułu, badaniu reakcji użytkowników czy weryfikacji źródła.

Przykładowo, niedawno został ujawniony raport organizacji FakeNews.PL, który przedstawia skalę rosyjskiej dezinformacji o COVID-19 w Polsce[1]. Obszerny artykuł pojawił się dokładnie rok po wprowadzeniu pierwszych obostrzeń w naszym kraju. Opracowanie choć obszerne, z licznymi odwołaniami do innych opracowań naukowych, jest niestety bardzo spóźnione, ponieważ działania dezinformacyjne zdążyły już wyrządzić szkody. Przygotowanie tego typu dokumentu wymaga czasu i znajomości wielu źródeł.

W przeszłości (przed rozwojem internetu i otwarto-źródłowych zbiorów informacji) praca fact-checkera polegała na konsultacjach z ekspertami, przeglądaniu magazynów i analizie statystycznej w danym temacie [24]. Takie podejście było bardzo drogie i nieefektywne.

2.1.2. Weryfikacja przez wykorzystanie inteligencji tłumu

Drugim sposobem jest wykorzystanie z informacji pochodzących od ludzi nie będących ekspertami w danej dziedzinie.

Facebook wykorzystuje zbiorową inteligencję do oznaczania postów jako fałszywych [27]. Użytkownicy mają możliwość oznaczenia danego postu jako fałszywego. W tym przypadku jednak potrzebne są narzędzia do sprawdzenia wiarygodności i poprawności opinii danego użytkownika poprzez przekazanie podejrzanych artykułów do dalszej analizy

[1] Ujawniamy: skala rosyjskiej dezinformacji o COVID-19 w Polsce, fakenews.pl, dostęp: 29.05.2021

przez ekspertów czy też modelowanie za pomocą mechanizmów prawdopodobieństwa jakie są szanse, że użytkownik się nie myli.

Wykorzystanie zbiorowej inteligencji ma również swoje wady. Zgodnie z mądrością - *kłamstwo powtórzone tysiąc razy staje się prawdą*, często powielane informacje stają się dla nas bardziej naturalne i sprawiają wrażenie prawdziwych. Badanie przeprowadzone w 1977 roku pokazało, że ludzie, którzy mieli do czynienia z nieprawdziwymi stwierdzeniami są bardziej skłonni do uwierzenia w fałszywe informacje, podczas gdy te same informacje przedstawione osobom bez takiej styczności były uznawane przez nie jako fałszywe [11].

Problematyczny okazuje się także mechanizm działania mediów społecznościowych oraz algorytmów w nich zawartych [19]. Giganci technologiczni tacy jak Facebook czy Google zbudowali mechanizmy rekomendujące użytkownikom treści w obrębie ich zainteresowań. Okazuje się, że z tego powodu użytkownicy żyją w bańkach informacyjnych [4], osoby o prawicowych poglądach nie otrzymują treści lewicowych, a osoby o lewicowych poglądach, nie otrzymują treści prawicowych. Takie zachowanie doprowadza do dużej polaryzacji i radykalizacji poglądów. Dla problemu klasyfikacji fałszywych informacji oznacza to, że osoby będące wystawione jedynie na treści zgodne z ich poglądami mogą nie być w stanie obiektywnie ocenić prawdziwości otrzymywanych informacji niezgodnych z ich poglądami.

2.1.3. Metody manualne, a komputerowe

Manualne oznaczanie fałszywych wiadomości może być podstawą do budowania podejść opartych na sztucznej inteligencji. Zbiory danych oznaczone przez człowieka mogą służyć do trenowania nadzorowanych algorytmów uczenia maszynowego [31].

Powstało wiele opisanych zbiorów fałszywych i prawdziwych informacji. Najpopularniejszym typem zbiorów są zbiory binarne - artykuł prawdziwy lub fałszywy. Znacznie mniej popularnymi zbiorami są zbiory opisane kilkustopniową skalą prawdziwości (całkowicie prawdziwe, częściowo prawdziwe, częściowo nieprawdziwe, całkowicie nieprawdziwe). Opisane zbiory posiadają najczęściej treść i tytuł artykułu, kategorię oraz datę powstania[2], bogatsze zbiory czasami zwierają linki do użytych obrazków, autorów czy link do oryginalnego tekstu[3].

Metody manualne nadal pozostają więc bardzo ważną techniką walki z dezinformacją. Gromadzenie danych jest bardzo drogim i czasochłonnym procesem, dlatego szeroka dostępność darmowych opisanych zbiorów danych znacznie przyśpiesza badania nad automatycznymi metodami rozpoznawania fałszywych informacji.

[2]Fake and real news dataset, kaggle.com, dostęp: 01.05.2021
[3]FakeNewsNet, kaggle.com, dostęp: 01.05.2021

2.2. Komputerowe oznaczanie fałszywych wiadomości

Dziennie do internetu trafiają olbrzymie ilości danych, ręczna weryfikacja każdej informacji jest fizycznie niemożliwa. Stąd istnieje konieczność automatyzacji wykrywania fałszywych informacji. Istnieje kilka możliwych podejść analizy informacji, które mogą zostać wykorzystane w algorytmach uczenia maszynowego [31].

2.2.1. Analiza stylu

Twórców fałszywych informacji łączy podobny styl pisania [21] [24], na przykład: używanie silnie nacechowanych słów, ułożenie znaków interpunkcyjnych czy długość zdań. Fałszywe informacje często pisane są przez amatorów, dlatego w porównaniu z profesjonalnymi pisarzami łatwo je odróżnić. Artykuł napisany przez profesjonalnego twórcę powinien przedstawiać jedynie fakty, natomiast artykuły, które pisane są po to, aby wprowadzać w błąd, zawierają dużo opinii, często negatywnych oraz używają potocznego języka.

Analiza sentymentu [6] [17]

Analiza sentymentu, znana również jako eksploracja opinii, bada odczucia ludzi wobec określonych podmiotów.

Prostą metodą na uzyskanie tego współczynnika jest wykorzystanie słów pozytywnych do zwiększania wartości współczynnika o *+1*, a każde negatywne słowo jako zmniejszającego o *-1*. Do obliczenia współczynnika wykorzystuje się, zbiory słów uznanych za pozytywne lub negatywne (podejście regułowe), a także wykorzystuje się uczenie maszynowe.

Metody oparte na analizie stylu są uznawane za relatywnie dobre podejście do klasyfikacji wiadomości [31]. Niektórzy naukowcy sugerują użycie różnych cech tekstu jakie mogą być brane pod uwagę, są to m.in.: wszelkie liczniki - liczby zdań, słów, części mowy, itd., nacechowanie tekstu, obiektywność i wiele innych.

Analiza obiektywności [13]

Analiza subiektywności rozpoznaje kontekstową biegunowość uczuć / opinii w kierunku różnych obiektów. Klasyfikacja subiektywności kategoryzuje dany tekst jako subiektywny lub obiektywny. Tekst obiektywny zawiera jeden lub więcej faktów na temat danego zagadnienia, a tekst subiektywny wyraża opinie autora.

Do działania takiej analizy wykorzystuje się uczenie maszynowe wytrenowane na opisanych zbiorach danych (obiektywnych i nieobiektywnych wypowiedziach).

Wykorzystując liczniki do klasyfikacji, przydatne jest pojęcie bigramu [25].

Bigram [26]

Jest to sekwencja dwóch sąsiadujących tokenów (słów, sylab, liter). Bigram jest *n-gramem*, gdzie *n=2*. Bigramy są często wykorzystywane do budowania statystyk tekstu.

Przykład: "Ala ma czarnego kota."
Bigramy: (Ala, ma), (ma, czarnego), (czarnego, kota)

Bigramy mogą zostać wykorzystane do zliczenia najczęstszych wyrażeń, ponieważ okazuje się, że pewne stwierdzenia są charakterystyczne dla wypowiedzi prawdziwych, a inne są charakterystyczne dla stwierdzeń fałszywych [25] [31].

Aby ograniczyć liczbę bigramów oraz uzyskać lepszą generalizację, np. aby nie traktować osobno następujących bigramów: *(Ala, miała)* oraz *(Ala, ma)*, konieczne jest wykonanie lemmatyzacji.

Lemmatyzacja

Proces sprowadzania wyrazu do formy podstawowej w obszarze części mowy, którą reprezentuje.

Przykład ze słowem *goli*[a]:

- Piłkarz strzelił 7 **goli**, *goli* → *gol*
- Pewien mężczyzna co rano się **goli**, *goli* → *golić*
- Mężczyźni na plaży nudystów byli **goli**, *goli* → *goły*

Proces lemmatyzacji jest procesem trudnym w kontekście metod komputerowych, wykorzystuje się podejścia regułowe lub uczenie maszynowe.

[a]Przykład na podstawie wykładów Przetwarzania Języka Naturalnego, dr inż. Smywiński-Pohl Aleksander, AGH 2021

Autorzy podkreślają, że metody oparte na analizie stylu mogą efektywnie wykryć fałszywą informację już na wstępnym etapie propagacji, jednak mogą zostać łatwo oszukane jeśli autorzy fałszywych wiadomości nauczą się jak oszukać klasyfikator (tzw. zabawa w *kotka i myszkę*). Dodatkowo naukowcy zwracają uwagę na to, że styl wiadomości może mocno zależeć od okresu, w którym powstała oraz od środowiska / języka. Aby wyeliminować przytoczone problemy, twórcy algorytmów sugerują, aby klasyfikator miał także wiedzę o kontekście w jakim powstała dana wiadomość.

2.2.2. Analiza treści

Istnieją metody, które skupiają się na analizie informacji niesionej przez badane publikacje, zamiast analizować tekst, analizie poddawane są stwierdzenia i oceniana jest ich

prawdziwość [31].

Do implementacji takiego rozwiązania wykorzystuje się tzw. knowledge graphs (grafy wiedzy). Grafy zawierają ogromne ilości faktów, które są połączone ze sobą za pomocą pewnej relacji, np. Elon Musk połączony z firmą Tesla jako jej założyciel.

Knowledge Graphs [5]

Jest to baza wiedzy, która wykorzystuje grafowy model danych w celu zintegrowania danych. Grafy wiedzy są często używane do przechowywania wzajemnie powiązanych opisów podmiotów - obiektów, zdarzeń, sytuacji lub abstrakcyjnych pojęć - z dowolną semantyką.

Knowledge Graphs są wykorzystywane np. przez Google do wyświetlania informacji o wyszukiwanej frazie. Do budowy struktury zawierającej ponad 70 miliardów faktów[4], wykorzystuje się dane m.in. z Wikipedii czy baz danych CIA[5]. Google nie udostępnia swojej technologii, ale istnieją otwarte grafy wiedzy, np. KBPedia [6].

Metody oparte na sprawdzaniu tez, aktualnie nie są na radarze naukowców ze względu na ich niską skuteczność [31]. Dużym problemem jest brak uniwersalnych źródeł prawdy, które działałyby efektywnie w dynamicznych środowiskach. Oznacza to, że takie metody nie są w stanie poprawnie ocenić wiadomości, które niosą nowe fakty. Przykładowo, informacja o potencjalnej katastrofie nuklearnej nie może zostać zweryfikowana używając dotychczasowej wiedzy pochodzącej z bazy danych, taka informacja musiałaby najpierw zostać tam dodana, co znacznie ogranicza sens istnienia takiego podejścia jeśli poszukujemy metody całkowicie automatycznej.

2.2.3. Analiza propagacji

Kolejnym, podejściem jest analiza propagacji. Zamiast badania zawartości (analiza stylu lub tekstu), analizuje się powiązania z innymi informacjami oraz to jak dany artykuł rozprzestrzeniał się po sieci. Czyli analizuje się, jak informacja była przekazywana, na przykład z jaką dynamiką Tweet był przekazywany dalej (*retweetowany*).

Okazuje się, że informacje fałszywe są chętniej przekazywane dalej [31], co oznacza, że propagują się szybciej i osiągają większy zasięg. W tym momencie można dostrzec wadę tego podejścia. Do klasyfikacji potrzebny jest pełen obraz propagacji wiadomości (jeśli nie ma historii propagacji, nie ma na czym bazować klasyfikacji). Jeśli potrzebny jest pełen obraz propagacji, konieczne jest pozwolenie każdej wiadomości na nieograniczone rozpowszechnianie się, w tym informacjom fałszywym. Więc, zanim

[4]Google Knowledge Graph, en.wikipedia.org, dostęp: 01.05.2021
[5]The World Factbook, cia.gov, dostęp: 01.05.2021
[6]KBpedia - an open-source knowledge graph, kbpedia.org, dostęp: 01.05.2021

zostanie osiągnięty pełen obraz propagacji, fałszywa informacja rozprzestrzenia się po sieci dokonując ciężkich do odwrócenia zniszczeń. Z tego powodu, okazuje się, że takie metody mają trudności w poprawnej analizie przypadków będących we wczesnej fazie propagacji [29] co znacznie ogranicza ich użyteczność.

2.2.4. Analiza postawy

Analiza postawy polega na automatycznym wykrywaniu, czy autor tekstu opowiada się za danym stwierdzeniem, czy jest przeciw niemu. Analiza postawy może zostać wykorzystana do wykrywania fałszywych informacji, przez sprawdzanie czy nagłówek mówi o tym samym co treść artykułu.

W 2017 szereg uniwersytetów oraz prywatnych organizacji zorganizowało konkurs *Fake News Challenge*. Autorzy dostarczyli uczestnikom zbiór 50-u tysięcy opisanych artykułów i nagłówków. Podejście, które zajęło drugie miejsce uzyskało dokładność na poziomie 81% [10]. Autorzy zaproponowali użycie cech: *Term Frequency Inverse Document Frequency, Latent Semantic Indexing, Latent Semantic Analysis* oraz *Paraphrase Detection based on Word Embeddings*, które stanowiły wejście do sieci neuronowej.

- *Term Frequency Inverse Document Frequency* [14] [22] - jest to narzędzie statystyczne pozwalające na opisanie ważności danego słowa w dokumencie. Miara składa się z dwóch części: TF oraz IDF.

  TF - liczba wystąpień frazy w dokumencie przez liczbę znajdujących się w nim wszystkich słów.

  IDF - logarytm ogólnej liczby dokumentów w badanym korpusie językowym przez liczbę dokumentów, które zawierają co najmniej jedno wystąpienie słowa.

- *Latent Semantic Analysis / Latent Semantic Indexing* [15] [23] - jest to narzędzie pozwalające na identyfikację elementów mających podobne znaczenie w niestrukturyzowanej kolekcji tekstów. Słowa odnajdywane przez LSA/LSI, to słowa, które często występują razem, na przykład *Apple* i *iPhone*, często występują w tym samym kontekście, mają bliskie znaczenie, ale nie są synonimami.

- *Paraphrase Detection based on Word Embeddings* [8] - detekcja parafrazowania polega na porównaniu dwóch tekstów i określeniu czy opisują to samo równymi słowami. Może być realizowana przy użyciu osadzeń słów (słowa zamieniane na specjalną postać wektorową) przekazywanych do sieci neuronowej.

Naukowcy dogłębnie przeanalizowali zachowanie narzędzia i doszli do wniosku, że nie jest ono podejściem wystarczającym, głównie ze względu na niedostateczną generalizację modelu. Przytoczyli kilka przykładów na dowód, że detekcja fałszywych informacji przy pomocy analizy nastawienia jest problemem trudnym i ani ich podejście ani podejście, które zajęło pierwsze miejsce nie radzi sobie z najtrudniejszymi przypadkami. Autorzy sugerują, że do rozwiązania problemu konieczne mogą być bardziej zaawansowane techniki uczenia maszynowego oraz inny zbiór cech wykorzystywanych do treningu i klasyfikacji.

2.2.5. Analiza reputacji źródła

Sama analiza tego skąd pochodzi dana informacja może stanowić dobry punkt początkowy dla klasyfikacji [29]. Naukowcy twierdzą, że fałszywe informacje pochodzą zazwyczaj ze źródeł, które specjalizują się w produkcji takich wiadomości. Jednak istnieją też źródła, które dla niepoznaki publikują czasami prawdziwe wiadomości w celu zmylenia czytelnika i w celu budowania wizerunku rzetelnego źródła [21]. Dodatkowo kolejnym problemem z poleganiem na samych źródłach jest to, że strony publikujące fałszywe informacje często istnieją tylko przez kilka miesięcy [1]. Przykładowo w 2016, w roku, w którym odbywały się wybory w USA, powstało wiele stron publikujących fałszywe informacje, które zaraz po wyborach zniknęło z internetu. Był to element wojny informacyjnej.

Zatem w sytuacji, w której źródła wiadomości pojawiają się i znikają, trudna jest budowa klasyfikatora o wysokiej skuteczności bazującego tylko na reputacji strony, z której wiadomości pochodzą.

2.2.6. Analiza cech wizualnych

Okazuje się, że analiza cech wizualnych również może posłużyć do wykrywania fałszywych informacji [25]. Taka analiza polega na badaniu liczy obrazów czy stosunku liczby obrazów do całości tekstu, ale może także polegać na analizie samych obrazów. Fałszywe informacje wykorzystują ludzką podatność na wiarę w obrazy przedstawiające sensacyjne czy oburzające treści. Okazuje się, że można zbudować klasyfikator, który jest w stanie uchwycić elementy powodujące silną reakcję emocjonalną. Do treningu klasyfikatora wykorzystuje się przede wszystkim cechy statystyczne obrazów, takie jak *clarity score, coherence score, diversity score* czy *clustering score* [3].

- *clarity score* - mierzy różnicę w rozkładzie pomiędzy dwoma zbiorami obrazów: jeden to zbiór obrazów w określonym zdarzeniu (zbiór zdarzeń), a drugi to zbiór obrazów zawierający obrazy ze wszystkich zdarzeń (zbiór kolekcji). Ta cecha jest zdefiniowana jako dywergencja Kullbacka-Leiblera[7] (metryka, określająca rozbieżności między dwoma rozkładami prawdopodobieństwa).

- *cohenrence score* - mierzy spójność obrazów. Cecha ta jest obliczana na podstawie wizualnego podobieństwa pomiędzy dowolną parą obrazów przedstawiające wydarzenie. Do określania podobieństw wykorzystuje się zbiór cech GIST.

  Ekstrakcja cech GIST polega m.in. na filtrowaniu obrazu wejściowego na szereg niskopoziomowych kanałów cech wizualnych, takich jak intensywność, kolor, orientacja, ruch i migotanie w wielu skalach przestrzennych.

[7] Kullback–Leibler divergence, en.wikipedia.org, dostęp: 01.05.2021

- *diversity score* - mierzy różnorodność obrazów w artykule, bierze pod uwagę najpopularniejsze obrazy pochodzące z mediów społecznościowych do weryfikacji pokrycia dwóch zbiorów.

- *clustering score* - obrazy w wiadomości są kalsteryzowane, następnie oceniana jest liczba klastrów. Do klasteryzacji wykorzystuje się algorytm Hierarchical Agglomerative Clustering (HAC).

Inne cechy, które zalicza się do analizy wizualnej to popularność danego artykułu / obrazu mierzona jako ogólna liczba wyświetleń oraz liczba udostępnień.

Metoda wykorzystująca analizę wizualną jest stosunkowo młodą metodą. Istnieje zaledwie kilka opracowań traktujących o tym temacie, dlatego dalsze badania nad tym tematem są wymagane, aby metoda mogła być używana szerzej.

2.3. Klasyfikatory dla problemu wykrywania fałszywych wiadomości

Do problemu klasyfikacji binarnej, najczęściej wykorzystywanymi klasyfikatorami są [9]:

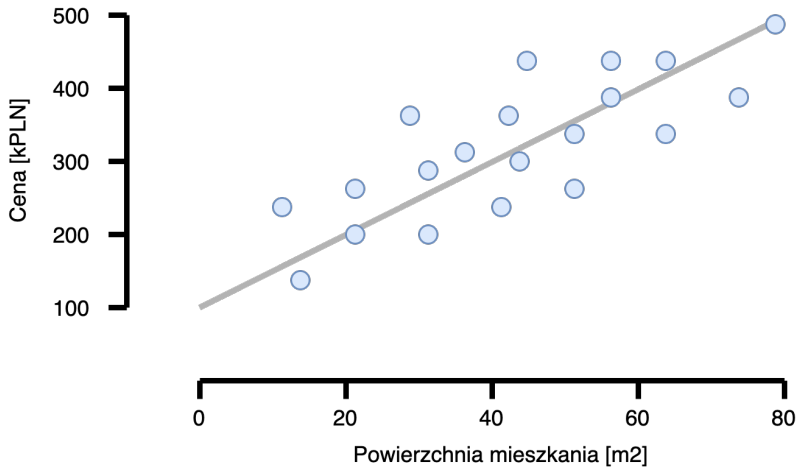
- Logistic Regression
- Passive Aggressive Classifier
- SGD Classifier
- Random Forest Classifier
- Gradient Boosting Classifier
- Extra Trees Classifier
- Hard Voting Classifier (Logistic Regression + Random Forest Classifier + Gradient Boosting Classifier + Extra Trees Classifier)
- Soft Voting Classifier (Logistic Regression + Random Forest Classifier + Gradient Boosting Classifier + Extra Trees Classifier)

Klasyfikacja

Proces podziału obiektów na klasy. W klasyfikacji można wyróżnić problem klasyfikacji binarnej, tj. przypisywania obiektów do jednej z dwóch klas. Klasyfikacja binarna była głównym problemem rozpatrywanym w tej pracy.

2.3.1. Logistic Regression

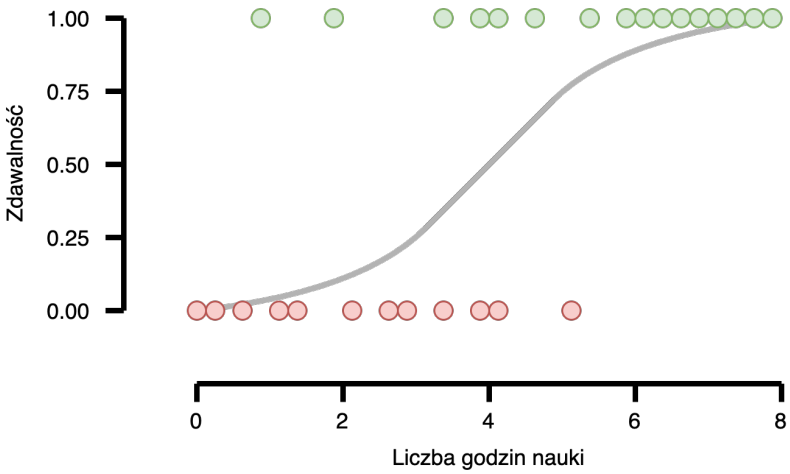
Regresja modeluje docelową wartość predykcyjną w oparciu o zmienne niezależne. Najczęściej wykorzystywana do ustalania zależności między zmiennymi i prognozowaniem.



Rysunek 1: Regresja liniowa na przykładzie cen mieszkań.

Powyższy przykład obrazuje jak liniowo zmienia się cena mieszkań w zależności od powierzchni (regresja liniowa). Wykorzystując dostępne obserwacje, można dopasować linię, która pozwoli na oszacowanie ceny mieszkań, o powierzchni nie występującej w zbiorze danych. Innymi słowami, można *wyinterpolować* taką wartość na podstawie obserwacji.

Niektóre algorytmy regresji mogą zostać użyte do problemu klasyfikacji. *Regresja Logistyczna* należy do jednych z nich. Służy ona do szacowania prawdopodobieństwa przynależności do danej klasy. Jeżeli szacowane prawdopodobieństwo wynosi powyżej 50%, instancja jest przypisywana do danej klasy pozytywnej (np. jest fałszywą wiadomością), w przeciwnym przypadku przypisywana jest do klasy negatywnej (np. nie jest fałszywą wiadomością).

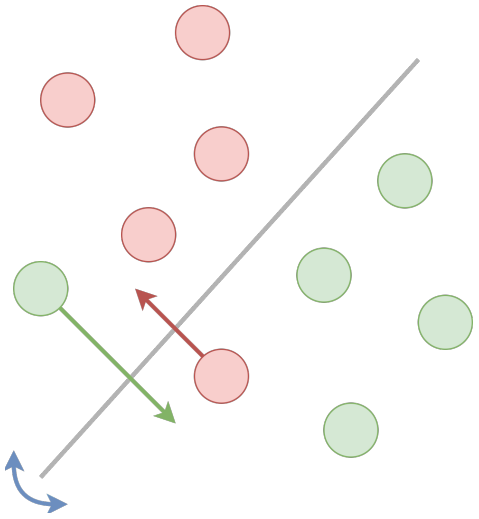


Rysunek 2: Regresja logistyczna na przykładzie zdawalności egzaminu.

Powyższy przykład obrazuje zdawalność w zależności od liczby przestudiowanych godzin. Im więcej godzin nauki, tym większa szansa na zdanie (przypisanie do jednej z dwóch klas - zdał / nie zdał).

2.3.2. Passive Aggressive Classifier

Jest to algorytm służący do klasyfikacji wykorzystujący 2 tryby, *pasywny* i *agresywny*. Podczas nauki, jeżeli algorytm dokonał poprawnego przewidywania, znajdzie się w trybie *pasywnym*, oznacza to, że parametry modelu nie ulegną zmianie. Jeżeli zostanie dokonana nieprawidłowa predykcja, algorytm znajdzie się w stanie *agresywnym*, który spowoduje aktualizacje parametrów modelu. Na rysunku poniżej przedstawiony została przedstawiona wysokopoziomowa koncepcja klasyfikatora w stanie agresywnym, który wymaga aktualizacji parametrów modelu.



Rysunek 3: Klasyfikator w stanie agresywnym, wymagana zmiana parametrów.

Klasyfikator Pasywno-Agresywny zaliczany jest do grupy algorytmów *online*. Algorytmy *online* w uczeniu maszynowym to algorytmy, które są w stanie uczyć się inkrementacyjnie. Oznacza to, że taki algorytm jest w stanie przyjmować partie danych treningowych zamiast oczekiwać całego zbioru na etapie treningu. Takie modele mogą być wykorzystywane w środowiskach produkcyjnych i być na bieżąco *dotrenowywane* przy użyciu świeżych danych.

2.3.3. SGD Classifier

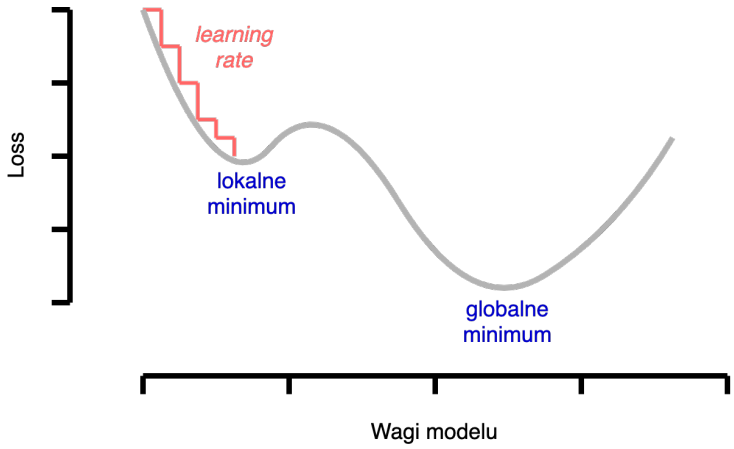
Pełna nazwa to *Stochastic Gradient Descent Classifier*. Nazwa jest w pewien sposób myląca, ponieważ SDG sam w sobie nie jest klasyfikatorem. W rzeczywistości jest to klasyfikator regresyjny, wykorzystujący optymalizację SDG - metodę spadku gradientowego, który przyspiesza czas odnajdowania optymalnego rozwiązania.

Spadek gradientowy

Generyczny algorytm optymalizujący, będący w stanie znaleźć optymalne rozwiązania dla szerokiej gamy problemów.

Służy do odnajdowania minimum zadanej funkcji.

Na poniższym rysunku zostało pokazane działanie algorytmu spadku gradientowego. Celem algorytmu jest odnalezienie takich parametrów, dla których funkcja kosztu ma najmniejszą wartość. Algorytm przeszukuje wartościowanie wykorzystując tzw. *learning rate* - wybrany ręcznie przez programistę rozmiar kroku, który jest wykorzystywany do poszukiwania minimum. Podczas przeszukiwania istnieje ryzyko utknięcia w lokalnym minimum - algorytm *wędrując* po wykresie uzna, że jest to najniższa wartość z jaką się spotkał i zakończy działanie, nie wiedząc, że gdzieś obok czeka globalne minimum.

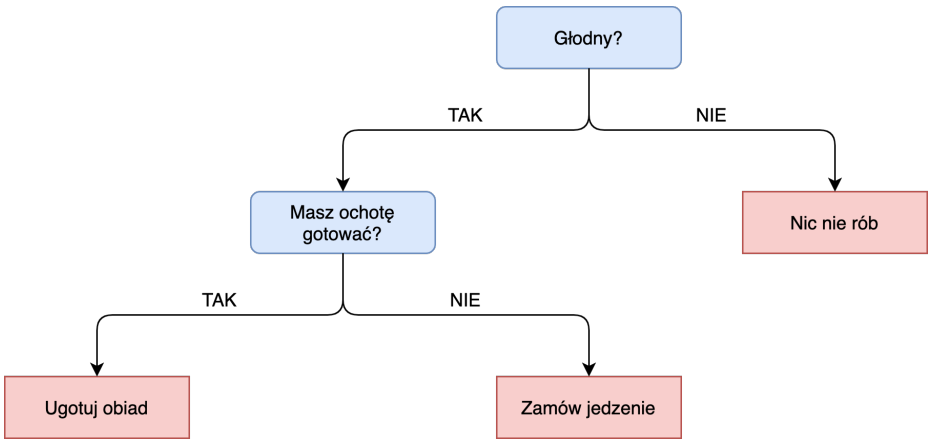


Rysunek 4: Metoda spadku gradientowego.

Stochastyczny Spadek Gradientowy eliminuje ten problem [2], dodatkowa losowość (stochastyczność) w wyborze punktów startowych znacznie ogranicza szanse na utknięcie w lokalnym minimum, dodatkowo znacznie przyśpiesza odnalezienie globalnego minimum.

2.3.4. Random Forest Classifier

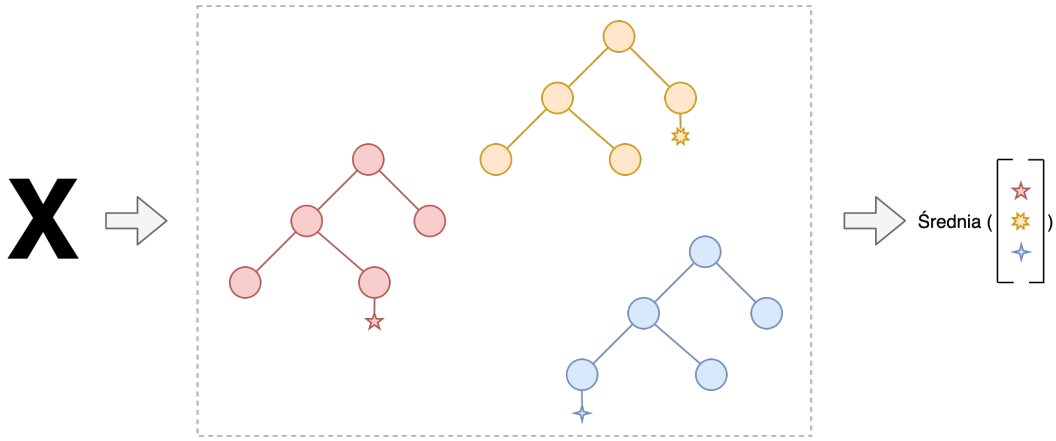
Drzewa decyzyjne to mechanizm definiowania algorytmu wykorzystujący strukturę drzewa. W węzłach drzewa znajdują się warunki, natomiast w liściach znajdują się decyzje. Przykładowe drzewo zostało pokazane na rysunku:



Rysunek 5: Proste drzewo decyzyjne dla problemu gotowania obiadu.

W uczeniu maszynowym, drzewo decyzyjne może zostać wyprodukowane przez komputer na podstawie obserwacji danych treningowych. Okazuje się, że ta stosunkowo prosta metoda jest zdolna osiągać bardzo dobre rezultaty, na przykład w klasyfikacji.

Klasyfikator losowych drzew jest metodą agregującą kilka drzew decyzyjnych w celu otrzymania lepszego klasyfikatora (*ensemble method*). Metoda ta polega na konstruowaniu wielu drzew decyzyjnych w czasie uczenia i generowaniu klasy, która jest dominantą klas lub przewidywaną średnią poszczególnych drzew. Generalnie *lasy losowe* poprawiają jakość klasyfikacji względem zwykłych drzew decyzyjnych.

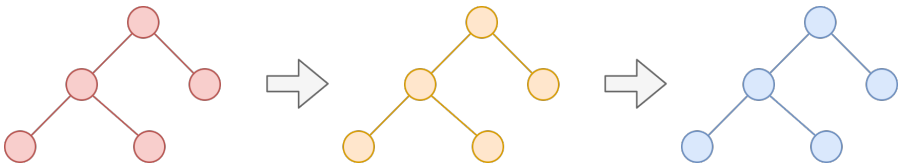


Rysunek 6: Random Forrest Classifier.

Na powyższym rysunku przedstawiony został klasyfikator składający się z wielu drzew, wektor wejściowy X jest przetwarzany przez wiele indywidualnych drzew, które wydają decyzje samodzielnie. Następnie, rezultatem może być średnia wyników wszystkich drzew jako ostateczny wynik klasyfikacji.

2.3.5. Gradient Boosting Classifier

Algorytmy należące do klasy *boosting* odnoszą się do dowolnej metody wykorzystującej kilka algorytmów w celu uzyskania klasyfikatora o lepszej jakości (*ensemble learning*). Metoda *boosting* generalnie polega na treningu klasyfikatorów sekwencyjnie, każdy kolejny stara się poprawić błędy poprzednika.



Rysunek 7: Gradient Boosting Trees.

Na powyższym rysunku zostały przedstawione drzewa decyzyjne połączone w metodę *boosting*. W fazie uczenia procedura *boostingu* trenuje nowy model kilka razy, za każdym razem dostosowując parametry nowego modelu do błędów dotychczas istniejącego modelu *boostowanego*.

Gradient Boosting Classifier działa przez sekwencyjne dodawanie klasyfikatorów, każdy nowo dodany ma za zadanie poprawić błędy swojego poprzednika. Ta metoda zamiast zmieniać parametry modelów w każdej iteracji, stara się dopasować klasyfikatory do błędu rezydualnego (*residual error*).

Błąd rezydualny

Różnica między grupą obserwowanych wartości, a ich średnią arytmetyczną.

2.3.6. Extra Trees Classifier

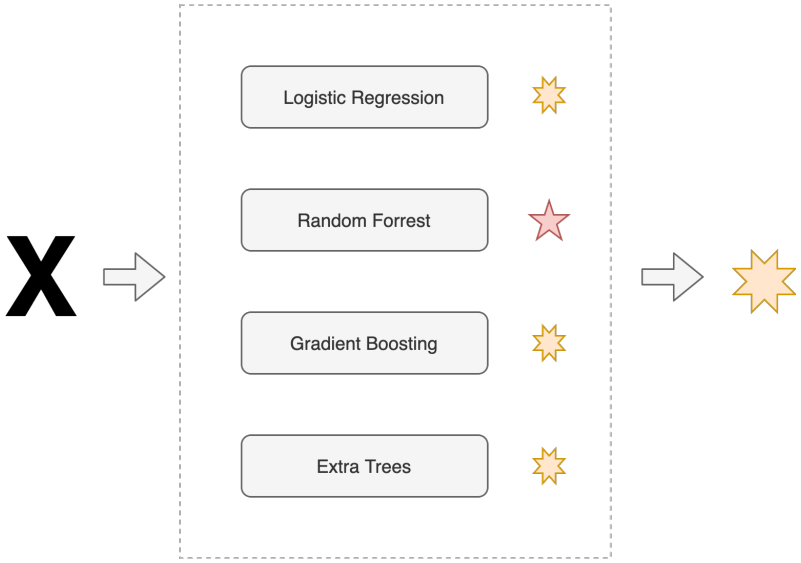
Pełna nazwa to *Extremly Randomized Trees Classifier*. Ekstremalnie losowe drzewa decyzyjne to w zasadzie losowe drzewa decyzyjne, które są tworzone w nieco inny sposób.

Podczas tworzenia drzewa w lesie drzew losowych, wybierany jest losowy podzbiór cech wybierany do podziału węzła. Możliwe jest wprowadzenie dodatkowej losowości, drzewa decyzyjne używają najlepszych możliwych progów (*threshold*) dla każdej z cech, zamiast tego można użyć progów losowych. Ma to także dodatkową zaletę, trening Extra Trees jest bardzo szybki, ponieważ najlepszy próg nie musi być ustalany a jest wybierany losowo. Ta dodatkowa losowość zwiększa *bias*, ale zmniejsza *wariancję*.

2.3.7. Voting Classifier

Jeśli wytrenowanych zostało kilka klasyfikatorów o dobrej dokładności, można połączyć je w jeden tzw. *klasyfikator głosujący*. Finalna decyzja jest wydawana poprzez agregację decyzji wybranych klasyfikatorów. Możliwe są dwa podejścia budowy klasyfikatora głosującego:

- Hard Voting Clasifier - klasa wybierana jest przez głosowanie większościowe
- Soft Voting Clasifier - zebrane zostają prawdopodobieństwa przypisania do danej klasy i zwracana jest średnia prawdopodobieństw



Rysunek 8: Głosujące klasyfikatory.

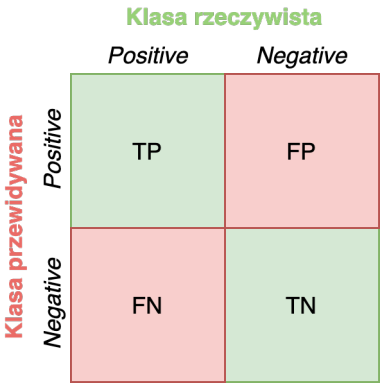
Na powyższym rysunku został przedstawiony przykład głosujących klasyfikatorów gdzie wybór jest większościowy - wygrywa klasa, która otrzyma najwięcej głosów.

2.3.8. Ocena jakości klasyfikacji

Problem jaki jest rozpatrywany w kontekście pisanej pracy jest problemem klasyfikacji binarnej - artykuł jest fałszywy czy nie? Zajmując się takimi problemami można ocenić zachowanie klasyfikatora przez przypisanie jego przewidywań do odpowiednich klas:

- *TP - True Positive* - poprawnie zidentyfikowana wiadomość fałszywa
- *TN - True Negative* - poprawnie zidentyfikowana wiadomość prawdziwa (niefałszywa)
- *FP - False Positive* - błędnie zidentyfikowana wiadomość fałszywa (w rzeczywistości prawdziwy artykuł)
- *FN - False Negative* - błędnie zidentyfikowana prawdziwa wiadomość (w rzeczywistości fałszywy artykuł)

Powyższe wartości można ująć w tabeli o następującej postaci:



Rysunek 9: Macierz pomyłek.

która nosi nazwę macierzy pomyłek i służy do wizualizacji jakości klasyfikacji. Klasyfikator dokonujący prawidłowych werdyktów będzie miał mało elementów fałszywie pozytywnych i fałszywie negatywnych (pomyłki).

Posiadając zdefiniowane klasy (TP, TN, TN i FN), można wykorzystać następujące metryki do oceny jakości klasyfikacji [9] [25]. Do badania jakości klasyfikatorów zostało użytych 5 metryk. Wyniki klasyfikacji przy użyciu metryk zostały przedstawione w rozdziale 5.

- Recall
- Precision
- Accuracy
- F1 score
- ROC AUC score

Recall. Metryka znana również jako *czułość*, a także jako *True Positive Rate*. Odpowiada na pytanie jak kompletne są wyniki. Przykładowo, jeśli komputer spośród 10 fałszywych wiadomości zidentyfikuje 5, wynik wyniesie 50%.

$$Recall = \frac{TP}{TP + FN} \tag{1}$$

Precision. Metryka znana również jako *precyzja* lub *Positive Predictive Value*. Mówi jak przydatne są wyniki przewidywania. Przykładowo, program zidentyfikował 6 fałszywych informacji oraz zidentyfikował 4 wiadomości prawdziwe jako fałszywe, wynik precyzji wyniesie 60%.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

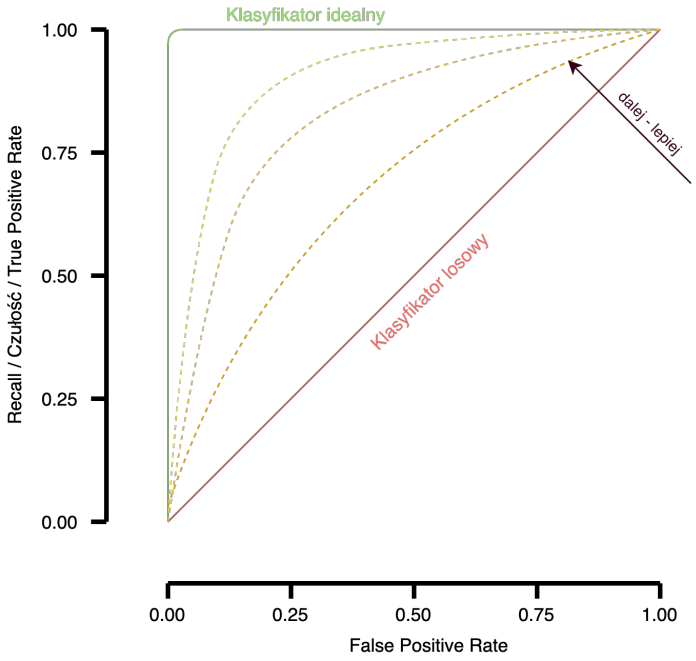
Accuracy. Metryka znana również jako *dokładność*. Mówi o stosunku dobrze dokonanych klasyfikacji do wszystkich dokonanych klasyfikacji.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

F1 score. Jest to średnia harmoniczna *Precision* oraz *Recall*. Dzięki swojej postaci jest w stanie uwzględnić dwie metryki, niewielkie różnice między wartościami nie są wzmacniane, natomiast duże różnice między wartościami są karane.

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

ROC AUC score. Pełna nazwa to *Receiver Operating Characteristic Area Under Curve Score*. Krzywa *ROC* przedstawia fałszywie dodatnią wartość na osi X (prawdziwe wiadomości zidentyfikowane jako fałszywe), a na osi Y pokazuje *Recall*. Następnie obliczane jest pole pod wykresem przez co otrzymywany jest wynik. Klasyfikator idealny posiada *ROC AUC score* na poziomie *1*.



Rysunek 10: Krzywa ROC.

Na powyższym wykresie zostały przedstawione krzywe ROC. Na czerwono zaznaczono krzywą klasyfikatora, który wydaje losowe decyzje, na zielono natomiast zaznaczono krzywą dla klasyfikatora idealnego.

Powyższe metryki są wystarczające w przypadku dużych zbiorów danych. Jednak dla mniej licznych zbiorów przydatna okazuje się technika **k-krotnej walidacji krzy-**

żowej. K-krotna walidacja krzyżowa jest jedną z metod, która próbuje zmaksymalizować wykorzystanie dostępnych danych do trenowania, a następnie testowania modelu. Zasada działania jest następująca:

1. Losowe wymieszanie elementów zbioru
2. Podział zbioru danych na k grup
3. Dla każdej z grup:
   (a) Wybór jednej z grup jako grupy testowej
   (b) Pozostałe grupy służą do treningu
   (c) Trening modelu
   (d) Obliczenie metryk
4. Podsumowanie wyników całego modelu

Co ważne, każda obserwacja w próbie danych jest przypisywana do indywidualnej grupy i pozostaje w tej grupie przez cały czas trwania procedury. Oznacza to, że każda próbka ma szansę być wykorzystana w zbiorze testowym 1 raz i użyta do trenowania modelu *k-1* razy.

2.4. Propagacja fałszywych informacji, a propagacja epidemii

Rozprzestrzenianie fałszywych wiadomości może w pewien sposób przypominać rozprzestrzenianie epidemii. Zainfekowane jednostki (źródła fałszywych informacji) przekazują innym jednostkom informacje, te je przyjmują, a następnie podają dalej. Wprost przypomina to transmisje wirusów, zainfekowane jednostki zarażają inne, te natomiast przekazują wirusa dalej.

To jak przekazywane są wirusy opisują rozmaite modele epidemii[8] [7]:

SIR - Susceptible (podatni na zachorowanie), Infectious (zarażeni), Recovered (ozdrowieńcy). Jest to prosty model epidemiologiczny.

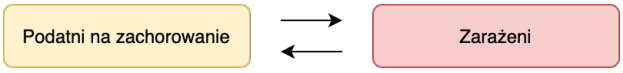
Rysunek 11: Schemat modelu SIR.

Na powyższym rysunku widoczne jest jak dane grupy przechodzą w kolejne. Podatni na zachorowanie to ogół populacji lub populacja pomniejszona o ozdrowieńców / osoby które przebyły chorobę. Jednostki z tej grupy mogą zostać zarażone. Osoby zarażone odpowiadają za dalsze przekazywanie wirusa, po przebyciu choroby stają się ozdrowieńcami. Do grupy ozdrowieńców / osób które przebyły chorobę zalicza się również osoby

[8]Compartmental models in epidemiology, en.wikipedia.org, dostęp: 01.05.2021

zmarłe.

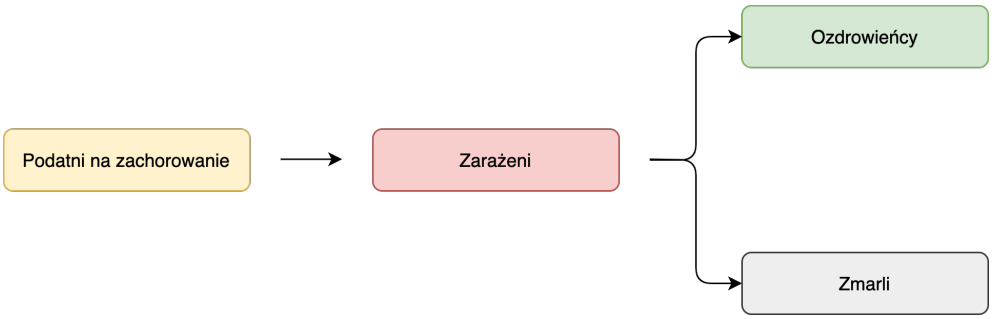
SIS - Susceptible (podatni na zachorowanie), Infectious (zarażeni), Susceptible (podatni na zachorowanie). Jest to kolejny prosty model, który nie zakłada nabycia długotrwałej odporności. Model pasuje do takich chorób jak grypa lub przeziębienie.



Rysunek 12: Schemat modelu SIS.

Jak widać na powyższym wykresie, jednostki populacji krążą miedzy dwoma grupami. Osoba może stać się zarażona i zarażać dalej, ale po przebyciu choroby znów może stać się podatna na zachorowanie.

SIRD - Susceptible (podatni na zachorowanie), Infectious (zarażeni), Recovered (ozdrowieńcy), Deceased (zmarli). Jest to rowzinięcie modelu *SIR*, który wyróżnia rezultat przebycia choroby.



Rysunek 13: Schemat modelu SIRD.

Takie rozróżnienie pozwala na dobranie różnych współczynników dla modelowania rezultatu zakończenia choroby. Przykładowo, można przypisać większy współczynnik przy przejściu do grupy zmarłych dla bardziej śmiercionośnych chorób.

SEIR - Susceptible (podatni na zachorowanie), Exposed (narażeni na zarażenie), Infectious (zarażeni), Recovered (ozdrowieńcy). W przypadku wielu infekcji istnieje znaczny okres inkubacji, podczas którego jednostka jest zakażona, ale sama nie jest jeszcze zakaźna.

Rysunek 14: Schemat modelu SEIR.

Na powyższym wykresie zostały pokazane przejścia jednostek między poszczególnymi grupami. Jak widać, model ten nie wyróżnia jednostek zmarłych.

Obserwacje istniejących modeli epidemii zainspirowały naukowców do stworzenia modelu DK (od nazwisk autorów: Daley i Kendall), który modeluje propagację nieprawdziwych informacji[9] [20]. Model ten zakłada istnienie N osób w populacji, które są podzielone na 3 grupy: Ignorants (ignoranci), Spreaders (rozpowiadacze), Stiflers (spowalniacze). Dokładniej, w kontekście rozprzestrzeniania informacji, znaczenie poszczególnych jednostek jest następujące:

- [I] Ignorants - osoby, które nie wiedzą o plotce
- [S] Spreaders - osoby, które aktywnie rozpowiadają plotkę
- [R] Stiflers - osoby, które słyszały plotkę, ale już nie są zainteresowane jej rozpowiadaniem

Jest to bezpośrednie nawiązanie do modelu SIR. Fałszywa informacja jest propagowana w populacji poprzez bezpośrednie kontakty między osobami aktywnie rozpowiadającymi nieprawdziwe informacje oraz innymi osobami. Osoby rozpowiadające plotki starają się *zarazić* każdą napotkaną osobę. W przypadku, gdy napotkają osobę, która nie słyszała danej informacji, osoba ta, sama może stać się rozpowiadaczem. W przypadku, gdy rozpowiadacz zorientuje się, że rozsiewa nieprawdziwe informacje, a następnie zaprzestaje tej czynności, staje się spowalniaczem.

Na podstawie modelu DK powstał model MK (od nazwiska pierwszego z autorów: Maki i Thompson), który w inny sposób modeluje transmisje zakażenia:

- gdy rozpowiadacz spotyka osobę, która nie wie nic o plotce, osoba nieświadoma zostaje rozpowiadaczem
- gdy spotka się dwóch rozpowiadaczy, jeden z nich zostaje spowalniaczem
- gdy rozpowiadacz spotka spowalniacza, rozpowiadacz traci zainteresowanie w dalszym rozpowszechnianiu informacji i sam zostaje spowalniaczem

Analiza fałszywych wiadomości może być również rozpatrywana pod kątem dynamiki rozprzestrzeniania. Jeżeli mówimy o dynamice rozprzestrzeniania, naturalne będzie wykorzystanie współczynnika R_0 - *Base Reproduction Number*.

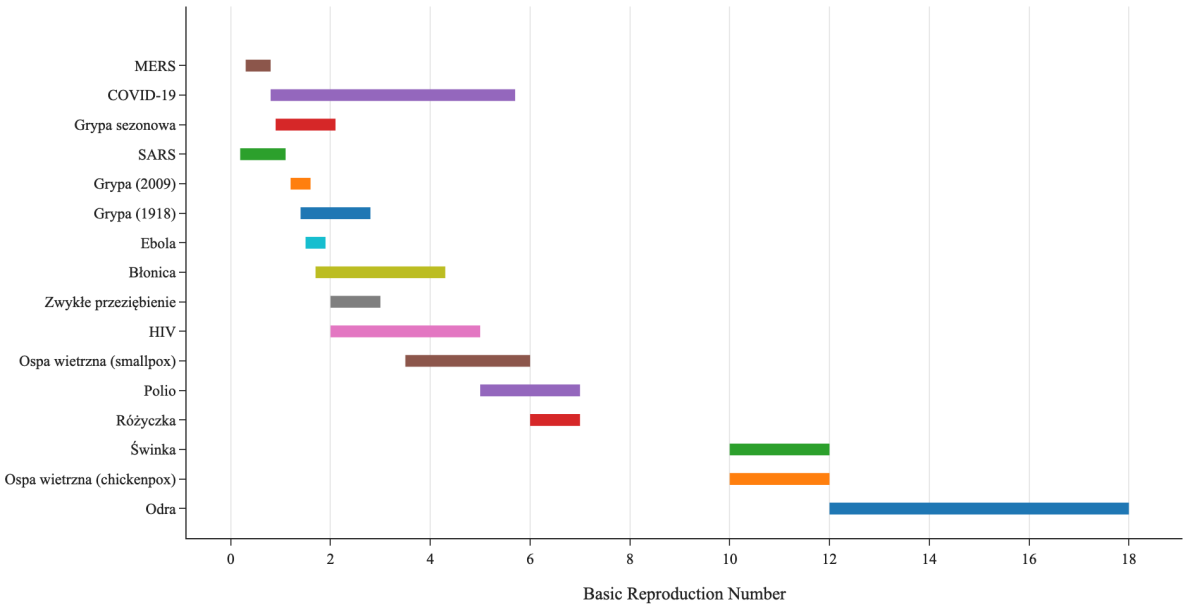
Base Reproduction Number

W epidemiologii: liczba osób, którą chory zaraża zakładając, że wszystkie kontakty zarażonego odbywają się z osobami podatnymi na zakażenie.

Na poniższym wykresie zostały przedstawione wartości współczynnika R_0 dla różnych chorób i wirusów. Wartości zostały ujęte jako przedziały ze względu na to, że wartość może być różna, w zależności w jakim środowisku badana jest choroba (inne

[9] Rumor spread in social network, en.wikipedia.org, dostęp: 01.05.2021

współczynniki na Europy, Afryki, itd. ze względu na różne uwarunkowania środowiskowe).



Rysunek 15: Wykres wartości R_0 dla różnych wirusów/chorób.

Najważniejszym zastosowaniem współczynnika R_0 jest określanie czy epidemia nabiera rozpędu czy jest wygaszana. Jeśli $R_0 > 1$, infekcja będzie rozprzestrzeniała się po populacji, ponieważ jedna zarażona osoba jest w stanie zarazić co najmniej jedną podatną osobę. Gdy $R_0 < 1$, zakażona osoba zaraża mniej niż jedną osobę, co oznacza, że dynamika rozwoju choroby jest wygaszana.

2.5. Potrzeba dalszych badań

Aktualne podejścia, zarówno manualne jak i komputerowe nie są wystarczające. Problem eliminacji fałszywych wiadomości wciąż pozostaje otwarty. Dziennie do internetu trafia ogromna ilość danych, w tym również nieprawdziwych informacji. Największe sieci społecznościowe nie radzą sobie z blokowaniem i usuwaniem szkodliwych informacji. Chociaż eksperci sprawdzający wiadomości ręcznie wykonują swoją prace z wysoką dokładnością, to metody automatyczne są jedynym dostępnym narzędziem, który może działać w dużej skali. Jednak jak się okazuje, aktualnie nie ma podejścia uniwersalnego. Wiadomo jedynie, że analiza cech tekstu jest metodą obiecującą. Wszelkie metody bazujące na automatycznej weryfikacji informacji nie radzą sobie z wiadomościami na nowe tematy. Z tego powodu w dalszej części pracy zaproponowano klasyfikację opartą na analizie tekstu.

Okazuje się również, że analiza propagacji może dawać dobre rezultaty w walce z fałszywymi informacjami. W pracy jednak skupiono się na przyrównaniu rozprzestrzeniania nieprawdziwych wiadomości do rozprzestrzeniania epidemii. Jak pokazano w rozdziale 2.4, istnieją 2 modele specyficzne dla rozprzestrzeniania się fałszywych

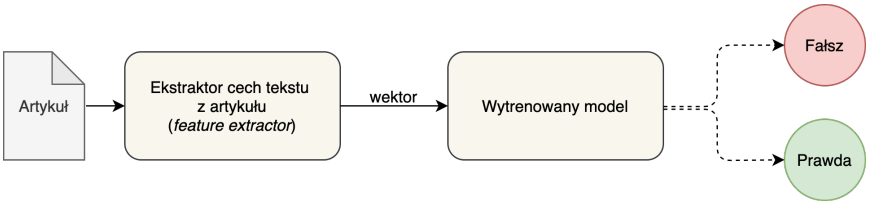
informacji, jednak brakuje opracowań przyrównujących współczynnik R_0 do różnych znanych chorób. W dalszej części pracy zbadano współczynnik oraz przedstawiono graficznie rozprzestrzenianie się fałszywych informacji.

3. Koncepcja klasyfikacji oraz analizy propagacji

W tym rozdziale została omówiona przyjęta strategia klasyfikacji, uargumentowany został wybór podejścia oraz opisany wybór cech tekstu. Opisane zostało także podejście do analizy rozprzestrzeniania się wiadomości.

3.1. Klasyfikacja fałszywych wiadomości

Do zadania wykrywania fałszywych informacji zdecydowano się na podejście oparte na analizie cech tekstu. Ta decyzja została podjęta po przeanalizowaniu aktualnego stanu wiedzy. Okazuje się, że klasyfikatory wykorzystujący tę metodę są aktualnie najczęstszym, a także dającym najlepsze rezultaty podejściem. Dodatkowym czynnikiem wpływającym na wybór metody była dostępność zbiorów treningowych oraz szeroka gama metod przetwarzania języka naturalnego, które zwłaszcza w ostatnich latach zyskały na dużej popularności oraz znacznie się rozwinęły.



Rysunek 16: Wysokopoziomowy schemat klasyfikatora.

Wysokopoziomowa koncepcja klasyfikatora została przedstawiona na powyższym rysunku. Celem narzędzia jest przyjęcie surowego (bez żadnej wstępnej obróbki) tekstu pochodzącego z artykułów, następnie przeanalizowanie go pod kątem wybranych cech omówionych w rozdziale 3.1.1, a następnie przekazać przygotowany wektor cech opisujący artykuł do wytrenowanego klasyfikatora, który podejmuje ostateczną decyzję o przypisaniu do odpowiedniej klasy.

3.1.1. Wybrane cechy tekstu

Na podstawie artykułów przytoczonych w rozdziale 2.2, zaproponowałem następujące cechy, które stanowiły podstawę klasyfikacji tekstu w kontekście fałszywych informacji:

1. Liczba pomyłek literackich (“literówek“, błędów ortograficznych) [21]
2. Liczba unikalnych słów w artykule. [21] [31]
3. Liczba zdań w artykule. [21] [25] [31]
4. Średnia liczba przymiotników w zdaniu. [21]
5. Średnia liczba rzeczowników w zdaniu. [21]
6. Średnia liczba czasowników w zdaniu. [21]

7. Sentyment artykułu. [31]
8. Obiektywność tekstu. [31]
9. *N* najpopularniejszych bigramów. [21] [25] [26]

Poniżej znajduje się uargumentowanie wyboru cech i ich potencjalnego wpływu na dokładność klasyfikacji.

Liczba pomyłek literackich. Fałszywe wiadomości często charakteryzują się licznymi błędami gramatycznymi, ortograficznymi czy błędną interpunkcją. Celem tego parametru jest zliczenie wszystkich błędów ortograficznych ("literówek") występujących w korpusie. Przykład fałszywej informacji z licznymi błędami ortograficznymi [18]:

> Breaking News! **Authority is** looking for this male in his early 20s located in the east bay. He was put in for mandatory quarantine after arrival from Hong Kong. He refused and has escaped the US base quarantine. We **ned** your help to find him before the spread of the deadly virus.

Pogrubione zostały elementy napisane niepoprawnie: zła forma osobowa (*Authority is* zamiast *Authorities are*) (I) oraz *ned* zamiast *need* (II). Błędy te wynikają ze słabej znajomości angielskiego (artykuły produkowane przez tzw. *farmy trolli* z obcych krajów), czy brak nadzoru nad pisanym tekstem, osoby tworzące fałszywe informacje produkują je masowo i szybko. W profesjonalnych redakcjach takie błędy zdarzają się rzadziej z konieczności zachowania profesjonalizmu oraz obecności nadzorców, którzy sprawdzają tekst przed opublikowaniem.

Liczba unikalnych słów w artykule. Kolejny zaproponowany parametr polega na zliczeniu wszystkich unikalnych wyrazów występujących w artykule. Razem z parametrem opisującym długość tekstu może opisywać bogactwo językowe artykułu. Zliczanie wyrazów wymaga dokonania lemmatyzacji, aby różne odmiany nie generowały nadmiarowych wyników.

Liczba zdań w artykule. Opisuje długość artykułu poprzez zliczenie występujących zdań. Razem z poprzednim parametrem może opisywać bogactwo językowe artykułu.

Średnia liczba przymiotników/czasowników/rzeczowników w zdaniu. Zbiór parametrów bazujących na analizie morfologicznej zdania. Te cechy polegają na wykryciu średniej liczby danej części mowy we wszystkich zdaniach artykułu.

Sentyment artykułu. Parametr ten odpowiada za wykrycie sentymentu, tj. emocji jakie płyną z analizowanego tekstu. Prawdziwe, rzetelne artykuły prasowe nie powinny wyrażać emocji autora, a jedynie informować o wydarzeniach bez ukierunkowania czytelnika przez stosowanie wyrazów uznanych za negatywne.

Przykład negatywnego zdania pochodzącego z fałszywego artykułu z jednego ze zbiorów treningowych:

> (ang.) If we don't take action soon, then Obama's **pathetic defeatist monologue** will soon become a reality.
> –
> (pol.) Jeśli wkrótce nie podejmiemy działań, to **żałosny defetystyczny monolog** Obamy wkrótce stanie się rzeczywistością.

Pogrubiony został fragment wyrażający silnie negatywne emocje.

Przykład zdania nie wyrażającego emocji, pochodzącego z prawdziwego artykułu:

> (ang.) Obama spoke to the United Nations General Assembly.
> –
> (pol.) Obama przemawiał na forum Zgromadzenia Ogólnego ONZ.

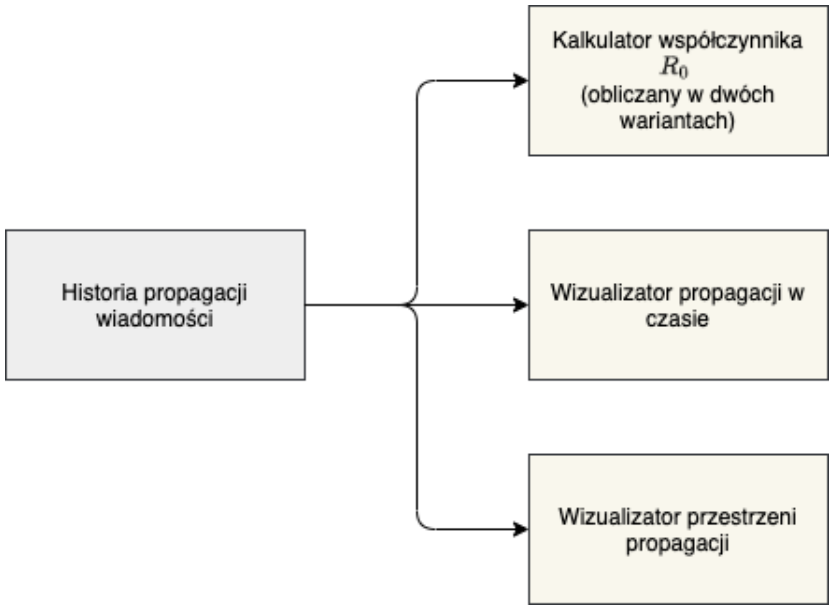
Obiektywność tekstu. Poza brakiem przesadnego nastawienia emocjonalnego względem tekstu (omówione wyżej), ważna jest również obiektywność. Obiektywność wskazuje czy tekst jest opinią czy subiektywnie przekazywaną informacją. Rzetelnie przekazywane informacje, tak jak robią to na przykład agencje prasowe takie jak Reuters słyną z obiektywności i surowego przekazywania faktów.

***N* najpopularniejszych bigramów.** Kolejnym parametrem jest wybór N najpopularniejszych (najczęściej pojawiających się w danym korpousie) bigramów i wykorzystanie ich do klasyfikacji. Niektórej stwierdzenia mogą pojawiać się częściej w fałszywych informacjach, a niektóre w prawdziwych.

Powyższe parametry posłużą do opisania artykułu w postaci wektora i będą wejściem do algorytmu uczenia maszynowego.

3.2. Propagacja fałszywych informacji

W drugiej części pracy przeanalizowałem jak rozpowszechniają się fałszywe informacje przez porównanie ich do rozpowszechniana się epidemii. Taka analiza skupiła się głównie na analizie współczynnika R_0, który został opisany w rozdziale 2.4 oraz graficznej analizy propagacji.



Rysunek 17: Wysokopoziomowy schemat analizy propagacji.

Powyżej została przedstawiona wysokopoziomowa koncepcja analizy rozprzestrzeniania. Na wejście przyjmowana była historia propagacji danej informacji, która następnie była analizowana. Analizie zostały poddane następujące czynniki:

- Obliczony został współczynnik R_0 dla fałszywych i prawdziwych informacji, w dwóch wariantach:
  - Pod uwagę wzięte zostały wszystkie obiekty, bez znaczenia do ilu osób przekazały informację.
  - Pod uwagę wzięte zostały wyłącznie obiekty, które przekazały informację do chociaż jednej jednostki.
- Wizualizacja propagacji informacji w czasie, pozwalająca na zobrazowanie dynamiki przekazywanych informacji.
- Wizualizacja propagacji w przestrzeni, pozwalająca na zidentyfikowanie głównych jednostek przekazujących informacje.

Tak przeprowadzona analiza pozwoliła na porównanie rozprzestrzeniania się fałszywych informacji do rozprzestrzeniania się epidemii.

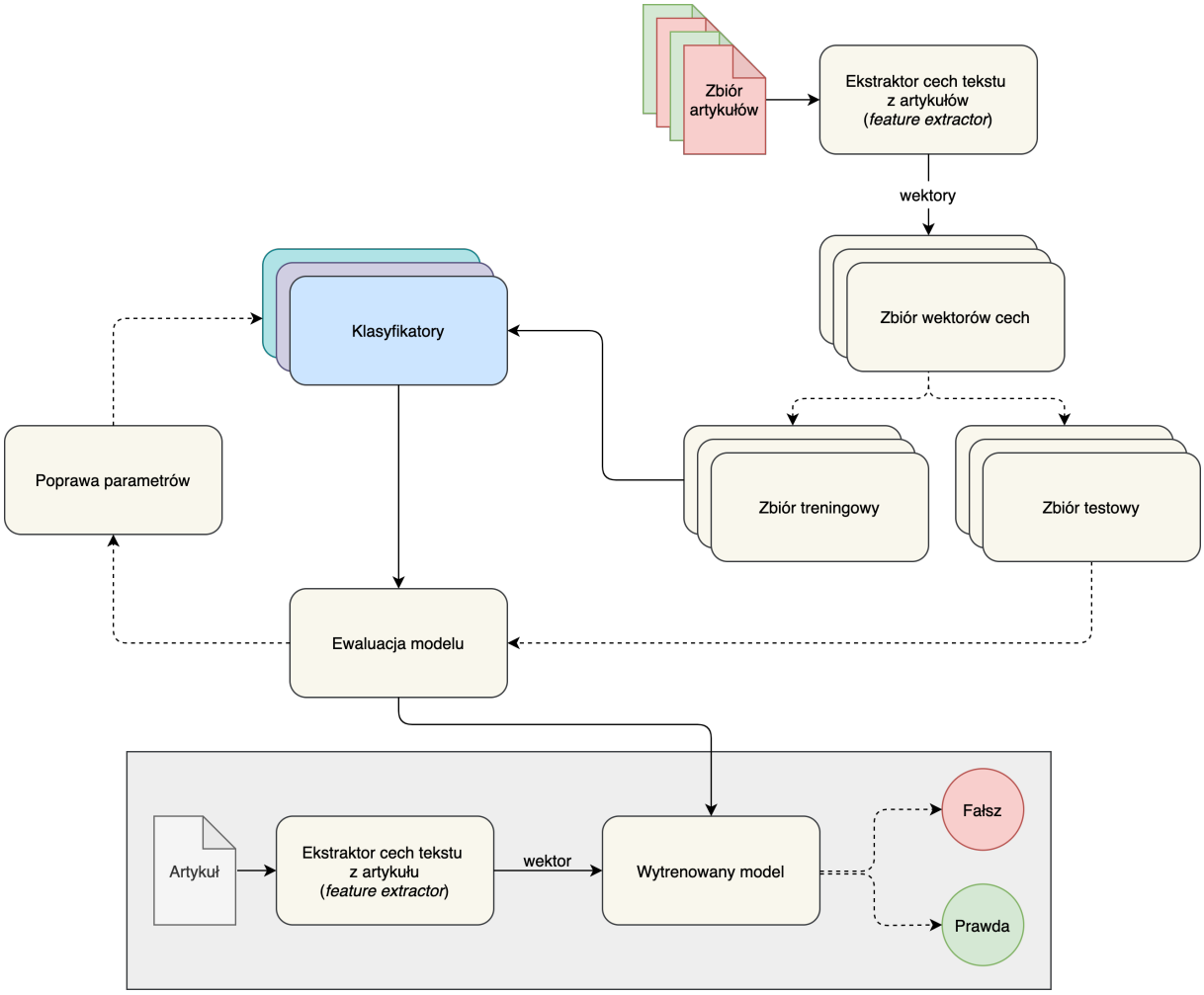
4. Architektura klasyfikatora oraz analizy propagacji

W tym rozdziale zostało opisane działanie programu, wykorzystane algorytmy oraz implementacja klasyfikatora, a także programu analizującego rozprzestrzenianie fałszywych informacji.

Cały program został napisany w języku Python 3.8.0 przy użyciu kilku bibliotek. Konkretne biblioteki zostały wymienione w poniższych podrozdziałach.

4.1. Klasyfikacja wiadomości

W rozdziale 3.1 przedstawiona została wysokopoziomowa koncepcja klasyfikatora, poniżej przedstawiony został szczegółowy schemat architektury programu.



Rysunek 18: Trening algorytmów oraz klasyfikacja

Wykres został omówiony w kolejnych podrozdziałach, począwszy od wejścia programu, kończąc na ostatecznych decyzjach klasyfikatora.

4.1.1. Wejście programu

Jak zostało to wspomniane w rozdziałach 3.1 oraz 4.1, program na wejście jest w stanie przyjąć kolekcje artykułów.

Program może przyjąć na wejście różne kolekcje dokumentów. Jest to cecha szczególnie użyteczna, ponieważ pozwoliła na określenie jakości klasyfikatora, na różnych zbiorach danych, co z kolei pozwoliło na uzyskanie większej pewności co do jakości otrzymanych wyników. Innymi słowy, wykorzystując kilka zbiorów danych, ograniczone zostało ryzyko wysunięcia nieprawidłowych wniosków, po wytrenowaniu modelu na tylko jednym zbiorze danych.

Poniżej znajduje się pseudokod funkcjonalności odpowiedzialnej za wczytywanie artykułów:

```
def get_news(dataset: Dataset, use_local: bool = True) -> List[News]:
    if use_local and _is_dump_available(dataset):
        return _load_dump(dataset)

    news = []

    if dataset is Dataset.DATASET_0:
        news = _get_dataset_0()
    if dataset is Dataset.DATASET_1:
        news = _get_dataset_1()
    if dataset is Dataset.DATASET_2:
        news = _get_dataset_2()

    _save_dump(dataset, news)

    return news
```

Funkcja na wejście przyjmuje identyfikator jednego z trzech przygotowanych zbiorów artykułów. Dodatkową funkcjonalnością jest możliwość użycia lokalnej kopii wczytanych artykułów. Lokalna kopia to pliki binarne, obsługiwane przy pomocy protokołu *pickle*. Pozwalają na przyśpieszenie ładowania danych, ponieważ zamiast wczytywać surowy tekst, używana jest zapisana kolekcja artykułów (plik binarny), która może być szybko załadowana do pamięci i używana w dalszej części programu.

Samo wczytywanie zbioru danych jest ściśle powiązane z formatem w jakim znajduje się ten zbiór. Oznacza to, że każdy zbiór ma swoją wyspecjalizowaną funkcję do wczytywania, najczęściej różnice między zbiorami wynikają z innego kodowania, innego nazewnictwa plików i pól w plikach CSV, bo to właśnie w nich znajdują się artykuły każdego z trzech zbiorów danych.

Do samego wczytywania plików wykorzystane zostały wyłącznie wbudowane funk-

cjonalności języka.

Wczytywanie polega także na przekształceniu pliku CSV do listy obiektów w języku Python:

```
@dataclass
class News:
    news_id: str
    title: Text
    content: Text
    is_fake: bool

    @property
    def is_true(self) -> bool:
        return not self.is_fake

    @property
    def all_text(self) -> Text:
        return self.title + self.content
```

Taki obiekt pozwala na łatwą manipulację danymi w kodzie przez swój ulepszony interfejs i zwiększenie czytelności przez możliwość zastosowania nazwanych pół oraz typowania. Dodatkowo obiekt można rozszerzać o nowe metody i parametry jeszcze bardziej zwiększając komfort korzystania z obiektów.

Sama treść artykułów jest przechowywana jako kolejny obiekt - *Text*:

```
class Text:
    def __init__(self, text_id: str, text: str):
        self.text_id = text_id
        self.text = _clear_text(text)

    @property
    def sentences(self) -> List[str]:
        ...

    @property
    def sentiment(self) -> Sentiment:
        ...

    @property
    def bigrams(self) -> List[str]:
        ...

    ...
```

Obiekt podczas inicjalizacji dokonuje czyszczenia artykułów, czyszczenie polega na:

- usunięciu linków, przy użyciu mechanizmu wyrażeń regularnych - usunięto wszystkie frazy zaczynające się na *http*.

- usunięcie tagów HTML, np. hiperłącza lub obrazki w tekście. Filtrowanie wykonane przy użyciu biblioteki *BeautifulSoup* służącej do parsowania HTML.

- usunięcie nawiasów prostokątnych, które niezwykle często występowały w tytułach oraz treści artykułów.

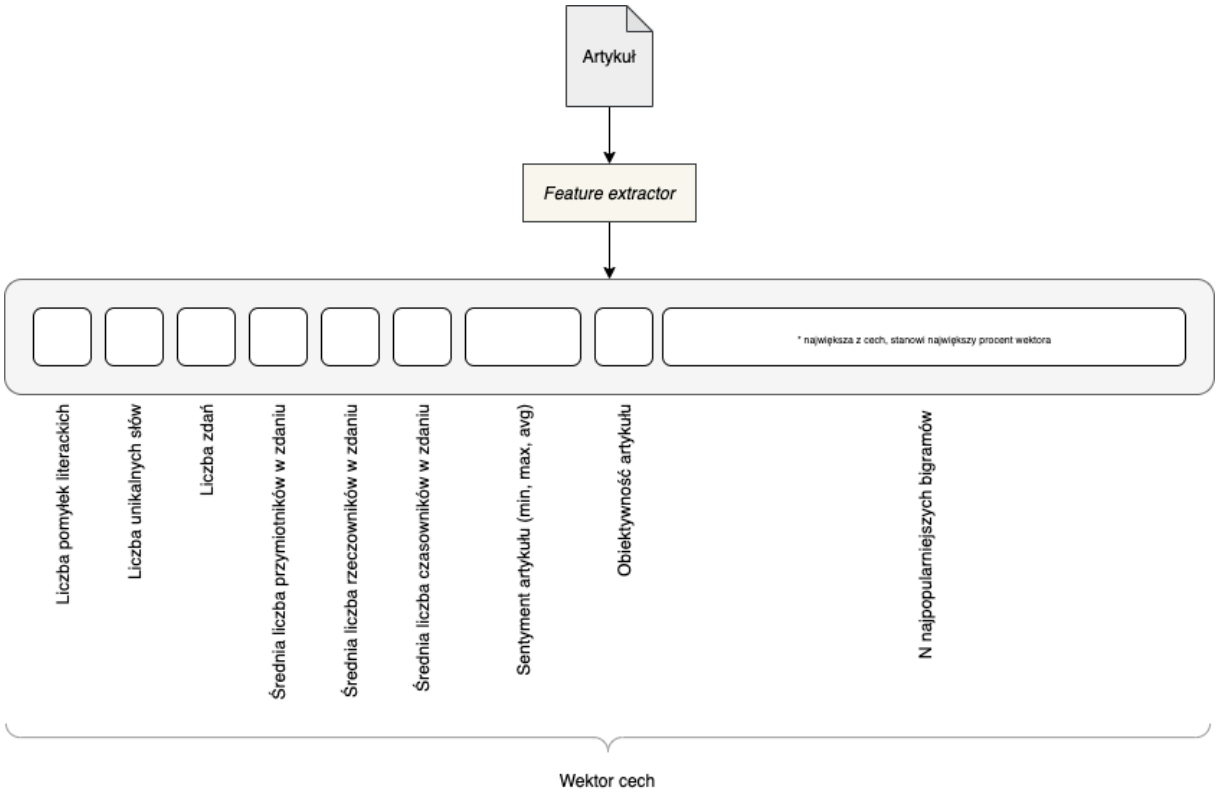
- rozwinięcie skrótu *U.S.* do *United States*, który również występował niezwykle często, a sprawiał problemy tokenizatorowi na dalszym etapie przetwarzania.

Obiekt posiada także szereg metod (dodatkowych atrybutów), pozwalających na wygodne uzyskanie zdań, tokenów, sentymentu, lemmatów, bigramów, itd. obliczanych na etapie działania programu. Dokładne działanie mechanizmu zostało opisane w rozdziale 4.1.2.

Wczytana lista artykułów zostaje następnie zrzucona do postaci pliku binarnego, w celu umożliwienia wczytania jej przy następnym wywołaniu funkcji z pominięciem parsowania pliku CSV i czyszczenia artykułów, co zostało opisane wyżej.

4.1.2. Ekstrakcja cech

Przygotowana lista artykułów, musi być następnie przekształcona do listy wektorów.



Rysunek 19: Cechy w postaci wektorowej

Proces obrazuje powyższy rysunek, na którym pojedynczy artykuł trafia do ekstraktora cech, który przekształca go do postaci wektora cech, który składa się z liczb, które rozumie algorytm uczenia maszynowego. Taka zamiana tekstu na wektor liczb wykonana została przy użyciu biblioteki *sklearn*.

```
class AttributesAdder(BaseEstimator, TransformerMixin):
    def fit(self, x, y=None):
        return self

    def transform(self, x: List[News]) -> np.array:
        return np.c_[
            num_of_misspellings(x),
            num_of_unique_words(x),
            num_of_sentences(x),
            avg_num_of_adjectives(x),
            avg_num_of_nouns(x),
            avg_num_of_verbs(x),
            news_length(x),
            news_sentiment(x),
            news_subjectivity(x),
            top_frequent_bigrams(x)
        ]

pipeline = Pipeline([
    ("attrs_adder", AttributesAdder()),
])

transformed_news = pipeline.fit_transform(all_news)
```

Do tego procesu przydał się mechanizm *Pipeline*, który przyjmuje listę kroków jakie musi wykonać biblioteka na zadanej liście, przed przekazaniem do algorytmu uczenia maszynowego. W przypadku tej pracy wykorzystany został jeden krok o nazwie *AttributesAdder*, który przyjmuje listę artykułów i zamienia ją na listę wektorów z biblioteki *numpy*. Mechanizm *Pipeline* nie był elementem koniecznym, da się uzyskać podobny efekt w inny sposób, ale jest to podejście eleganckie oraz zgodne z zamysłem twórców biblioteki *sklearn* [9].

Przekształcony artykuł ma postać wektora o długości 650 komórek, każda komórka wektora to liczba zmiennoprzecinkowa. Poniżej znajduje się fragment wektora jednego z artykułów:

```
2.93000000e+02, 3.50000000e+01, 2.57142857e+00, 5.48571429e+00, 3.50000000e+01, ...
```

Taki wektor powstaje przez sklejenie wektorów kolumnowych kilku cech przy użyciu funkcji z biblioteki *numpy* - *np.c_*. Poniżej znajduje się omówienie implementacji

poszczególnych cech.

num_of_misspellings - jest to cecha najbardziej wymagająca obliczeniowo, implementacja polega na obliczeniu dla każdego artykułu liczby pomyłek w całym tekście. Pomyłki są określane na podstawie:

- sprawdzeniu czy dane słowo istnieje w słowniku języka angielskiego biblioteki *Enchant*. Sprawdzona zostaje różna kapitalizacja słów - zaczynające się od wielkiej litery, wszystkie litery małe oraz wszystkie litery duże. Takie rozróżnienie wielkości liter pozwala na wychwycenie istnienia skrótów, nazw własnych oraz pospolitych słów.
- sprawdzeniu czy dane słowo istnieje w słowniku biblioteki *NLTK*. Tutaj podobnie zostają sprawdzone różne wersje słowa (różne wielkości liter).
- sprawdzeniu czy słowo istnieje w słowo-sieci (WorldNet) biblioteki *NLTK*. WordNet został użyty poniewaz jest to jedyna metoda poza słownikiem na sprawdzenie poprawności słowa bez znajomości kontekstu jego występowania.

WordNet - słowo-sieć

WordNet jest dużą leksykalną bazą danych języka angielskiego. Rzeczowniki, czasowniki, przymiotniki i przysłówki są pogrupowane w zbiory synonimów kognitywnych (synsety), z których każdy wyraża odrębne pojęcie. Synsety są powiązane ze sobą za pomocą relacji konceptualno-semantycznych i leksykalnych.

Jeśli żaden zasób nie odnalazł słowa, uznawane zostawało za pomyłkę literacką. Tak duże zasoby musiały zostać użyte, aby poprawnie zidentyfikować jak najwięcej słów. Wykorzystane zostały 2 słowniki oraz słowo-sieć, jak się okazuje żaden z tych zasobów, sam nie jest w stanie zidentyfikować poprawności wszystkich słów. Najbardziej problematyczne okazywały się nazwy własne - skróty organizacji, a przede wszystkim nazwiska. W rozdziale 3.1.1 zostały opisane przykładowe pomyłki, omawiana cecha jest w stanie uchwycić tylko pomyłki takie jak II (literówki), nie została przygotowana do wyłapywania błędów gramatycznych takich jak I.

num_of_unique_words - implementacja polega na zliczeniu unikalnych lemmatów z każdego z artykułów. Do wyznaczenia lemmatów wykorzystana została biblioteka *NLTK*, a dokładniej obiekt *WordNetLemmatizer* z niej pochodzący, który używa słowo-sieci do wyznaczenia poprawnego lemmatu. Po otrzymaniu wszystkich lemmatów dla danego tekstu, umieszczane są w Pythonowej kolekcji *set*, która ma właściwość eliminacji zduplikowanych elementów. Następnie obliczany jest rozmiar kolekcji i zapisywany w wynikowym wektorze.

num_of_sentences - polega na zliczeniu liczby zdań występujących w artykule. Działanie zostało oparte na funkcji *sent_tokenize* z biblioteki *NLTK*. Funkcja ta wykorzystuje wytrenowany model do podziału tekstu na zdania.

avg_num_of_adjectives - wynikiem tej funkcji liczba przymiotników. Zliczanie przymiotników odbywa się przy użyciu biblioteki *NLTK*, a dokładniej funkcji *pos_tag*, która na swoje wejście przyjmuje ztokenizowane zdanie, a na wyjściu zwraca części mowy przypisane do danego słowa. Części mowy zapisane są w formacie tagów części mowy Penna (Penn Part of Speech Tags), które w odróżnieniu od uniwersalnych części mowy (Universal POS tags) rozróżniają czas, stopniowanie czy mnogość wyrazów.

Dlatego, aby wykryć czy dane słowo jest przymiotnikiem należy sprawdzić czy należy do zbioru następujących tagów:

- JJ - przymiotnik
- JJR - przymiotnik w stopniu wyższym
- JJS - przymiotnik w stopniu najwyższym

avg_num_of_nouns - zliczanie liczby rzeczowników następuje w bardzo podobny sposób do zliczania przymiotników. Wykorzystana została ta samą funkcja z pakietu *NLTK*. Tagi opisujące rzeczowniki w notacji Penna, to:

- NN - rzeczownik, liczba pojedyncza
- NNS - rzeczownik, liczba mnoga
- NNP - rzeczownik własny, liczba pojedyncza
- NNPS - rzeczownik własny, liczba mnoga

avg_num_of_verbs - określanie liczby czasowników następuje w sposób bliźniaczy do określania liczby przymiotników i rzeczowników. Tagi opisujące rzeczowniki w notacji Penna, to:

- VB - czasownik, forma podstawowa
- VBD - czasownik, czas przeszły
- VBG - czasownik, gerund lub imiesłów teraźniejszy
- VBN - czasownik, past participle
- VBP - czasownik, pierwsza i druga osoba liczby pojedynczej teraźniejszej
- VBZ - czasownik, 3 osoba liczby pojedynczej teraźniejszej

news_length - długość artykułu została wyrażona poprzez zliczenie liczby zlemmatyzowanych słów.

news_sentiment - obliczenie sentymentu płynącego z artykułu to tak naprawdę, najpierw obliczenie sentymentu każdego zdania w tekście, a następnie obliczenie wartości

średniej, minimalnej oraz maksymalnej.

Obliczanie sentymentu zdań odbywa się przy użyciu biblioteki *NLTK*, a dokładniej obiektu *SentimentIntensityAnalyzer*, który używa analizatora sentymantu - VADER (*Valence Aware Dictionary for Sentiment Reasoning*). VADER jest modelem regułowym przygotowanym do analizowania danych pochodzących z internetu.

news_subjectivity - z kolei obiektywność artykułu jest obliczana na podstawie całego tekstu, bez podziału na pojedyncze zdania, w przeciwieństwie do wartości sentymentu. Obiektywność artykułu jest pojedynczą wartością liczbową, obliczaną przez bibliotekę *TextBlob*. Biblioteka ocenia obiektywność w skali od 0 (obiektywny) do 1 (subiektywny).

TextBlob oblicza obiektywnośc / subiektywność poprzez spojrzenie na "intensywność". Intensywność określa czy dane słowo modyfikuje następne słowo. W języku angielskim, przysłówki są używane jako modyfikatory (np. "bardzo dobry- bardzo dobry obiad - subiektywne odczucie).

top_frequent_bigrams - do wyznaczenia najpopularniejszych (najczęściej pojawiających się) bigramów ponownie wykorzystany został pakiet *NLTK* oraz obiekt *BigramCollocationFinder*, który na podstawie ztokenizowanego artykułu jest w stanie zwrócić posortowane bigramy. Bigramy mogą być posortowane według popularności, którą określa miara PMI.

O ile wyznaczenie par słów (bigramów) jest stosunkowo prostym zadaniem, to przekształcenie ich do postaci liczb wymaga dodatkowej pracy.

Do zamiany słów na liczby wykorzystany został model *Word2Vec*.

Word2Vec

Word2Vec jest techniką tworzenia osadzeń słów w przestrzeni wielowymiarowej, wynalezioną w 2013 roku.

Polega na wytrenowaniu sieci neuronowej, której zadaniem jest odkrycie powiązań między słowami, dzięki czemu słowa mogą być reprezentowane jako wektory liczb osadzane w przestrzeni.

Do treningu przygotowanej sieci neuronowej wykorzystano bibliotekę *Word2Vec*. Trening sieci wymagał przekazania wszystkich artykułów ze zbioru treningowego przekształconych do postaci listy tokenów. Tak przygotowana lista służyła do wykrycia powiązań w danym zbiorze i później umożliwia zamianę bigramów na wektory.

Każda z powyższych cech została wyposażona w mechanizm zrzucania wyniku do pliku binarnego za pomocą protokołu binarnego w celu uniknięcia ponownego obliczania

danych już obliczonych, podobnego do tego opisanego w rozdziale 4.1.1, przy ponownym uruchomieniu programu.

Jak zostało to przedstawione na rysunku 18, po stworzeniu zbioru wektorów, które odpowiadają zbiorowi artykułów, następuje podział wektorów na 2 rozłączne zbiory - zbiór treningowy i zbiór testowy. Zbiór treningowy służy do treningu klasyfikatora, a zbiór testowy służy do oceny jakości wytrenowanego klasyfikatora. Taki rozdział jest konieczny, ponieważ nie można używać tych samych przykładów do treningu i oceny, ponieważ klasyfikator mógłby oszukiwać, poprzez wykorzystywanie faktu, że miał już do czynienia z testowanym przykładem. Taka sytuacja doprowadziłaby do testowania czy klasyfikator zapamiętał przykłady, a nie czy potrafi ocenić próbki, których jeszcze nie widział.

Do podziału zbioru wykorzystana została funkcjonalność pakietu *sklearn*, który umożliwił przeznaczenie 75% artykułów do celów treningowych oraz pozostałe 25% do testowania.

Dodatkowo zapewnione zostało zbalansowanie zbiorów, to znaczy, w zbiorach treningowych oraz testowych znajdowało się tyle samo artykułów prawdziwych jak i fałszywych,

4.1.3. Klasyfikacja

Mając przygotowane zbiory do treningu oraz do walidacji, można przystąpić do treningu klasyfikatorów. Program został przygotowany, tak aby trenowane klasyfikatory były wymienne. Wykorzystane algorytmy klasyfikacji pochodzą z pakietu *sklearn*:

- LogisticRegression
- PassiveAggressiveClassifier
- SGDClassifier
- RandomForestClassifier
- GradientBoostingClassifier
- ExtraTreesClassifier
- HardVotingClassifier (LogisticRegression, RandomForestClassifier, GradientBoostingClassifier, ExtraTreesClassifier)
- SoftVotingClassifier (LogisticRegression, RandomForestClassifier, GradientBoostingClassifier, ExtraTreesClassifier)

i są tymi samymi algorytmami, które zostały opisane w rozdziale 2.3. Każdy z klasyfikatorów udostępnia interfejs, który pozwala na przekazanie zbioru treningowego do algorytmu uczenia maszynowego, następnie biblioteka sama uruchamia trening. Po przeprowadzeniu

uczenia, do dyspozycji pozostaje wytrenowany model, do którego można przekazać zbiór walidacyjny w celu obliczenia metryk. Wyniki poszczególnych algorytmów zostały opisane w rozdziale 5.1.2.

4.1.4. Wyjście programu

Najlepszy wytrenowany model można wykorzystać w aplikacji analizującej artykuły, w tym celu należy wykorzystać moduł odpowiedzialny za konwertowanie artykułu do postaci wektorowej, a następnie przekazać dane klasyfikatora. Jak zostało to pokazane na rysunku 18, model może określić przynależność do jednej z dwóch klas (artykuł prawdziwy lub fałszywy).

Do zdecydowania, który algorytm najlepiej nadaje się do klasyfikowania wiadomości, konieczne jest użycie metryk. Mechanizm obliczania metryk również pochodzi z pakietu *sklearn*, a samo działanie metryk zostało opisane w rozdziale 2.3.8. Funkcje z biblioteki *sklearn* do obliczania wartości różnych metryk to:

- recall_score
- accuracy_score
- precision_score
- f1_score
- roc_auc_score

Funkcje wykorzystują wzory 1, 2, 3, 4 opisane w rozdziale 2.3.8. Wyłonienie najlepszego klasyfikatora odbyło się w rozdziale 5.

4.1.5. Podsumowanie wykorzystanych bibliotek

W części klasyfikacyjnej wykorzystałem następujące biblioteki:

1. BeautifulSoup[10] - do oczyszczenia artykułów z pozostałych tagów HTML
2. numpy[11] - do operacji na macierzach
3. sklearn[12] - do transformacji wejścia oraz algorytmów uczenia maszynowego
4. Enchant[13] - do dostępu do słownika języka angielskiego
5. NLTK[14] - do dostępu do operacji przetwarzania języka naturalnego (lemmatyzacja, tokenizacja, analiza sentymentu), słowo-sieci, słownika języka angielskiego

[10] pypi.org/project/beautifulsoup4/, dostęp: 22.05.2021
[11] pypi.org/project/numpy/, dostęp: 22.05.2021
[12] pypi.org/project/scikit-learn/, dostęp: 22.05.2021
[13] pypi.org/project/pyenchant, dostęp: 22.05.2021
[14] pypi.org/project/nltk/, dostęp: 22.05.2021

6. TextBlob[15] - do analizy obiektywności tekstu

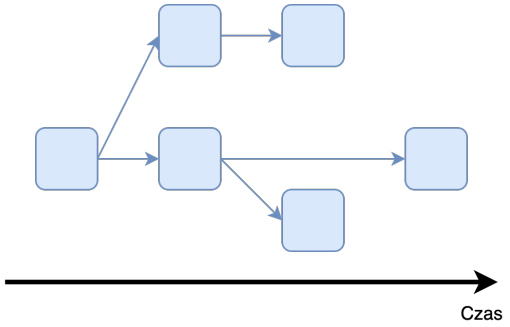
7. Word2Vec[16] - do tworzenia osadzeń bigramów w przestrzeni wielowymiarowej

4.2. Analiza propagacji

W rozdziale 2.4 opisane zostały modele epidemii oraz parametry jakie mogą zostać wykorzystane do analizy propagacji fałszywych informacji. Rysunek 17 przedstawia, schemat programu, który zajmuje się analizą rozprzestrzeniania. W poniższych rozdziałach zostało opisane dokładne działanie każdej z części analizującej.

4.2.1. Wczytywanie danych

Program został przystosowany do analizy historii propagacji wpisów pochodzących z serwisu Twitter.



Rysunek 20: Przykładowa propagacja wpisów w czasie.

Jak zostało to pokazane na rysunku powyżej, wpis może być wielokrotnie podawany dalej (*retweetowany*), co składa się na jego historię. Historia może mieć strukturę drzewa, w którym węzły (niebieskie kafelki) mają dowolną liczbę potomków.

Dane wejściowe dostarczają informacje potrzebne do budowy drzewa oraz czas jaki upływał pomiędzy poszczególnymi podaniami dalej. Dodatkowo zbiór danych posiada kolekcję wpisów fałszywych, prawdziwych oraz niezweryfikowanych. Oznacza to, że można skonstruować obiekt w następującej postaci:

```
@dataclass
class Tweet:
    content: str
    tweet_type: TweetType
    delay: float
    children: List[Tweet]
```

[15] pypi.org/project/textblob/, dostęp: 22.05.2021
[16] pypi.org/project/gensim/, dostęp: 22.05.2021

Przyjęta struktura ma charakter rekurencyjny i w dobry sposób oddaje charakterystykę danych wejściowych. Plik tekstowy przetworzony do postaci obiektów, również posiada mechanizm zrzutu do plików binarnych podobny do tego opisanego w rozdziale 4.1.1.

4.2.2. Analiza współczynnika R_0

Jak zostało to opisane w rozdziale 3.2, obliczanie współczynnika R_0 odbywa się w dwóch trybach. Dzięki zastosowanej strukturze danych, rozróżnienie węzłów, które uzyskały chociaż jedno przekazanie dalej od tych, które nie mają potomków jest łatwe, ponieważ polega na sprawdzeniu czy dany wpis ma dzieci.

Obliczenie współczynnika R_0 dla propagacji *Tweetów* nie wymagało użycia dodatkowych bibliotek.

4.2.3. Analiza propagacji w czasie

Do tego jak przebiegała propagacja wpisu na przestrzeni czasu wykorzystano analizę graficzną. Jak zostało to opisane w rozdziale 3.2, wizualizacja dynamiki rozprzestrzeniania polega na wizualizacji przyrostu liczby podań dalej w czasie.

Do zadania wizualizacji wykorzystano bibliotekę *plotly*.

4.2.4. Analiza propagacji w przestrzeni

Jak zostało to opisane w rozdziale 3.2, analiza propagacji w przestrzeni polega na wizualizacji sieci połączeń miedzy *Tweetami*. Ponieważ jeden wpis może zostać podany wielokrotnie, taka analiza może pozwolić na identyfikację głównych źródeł rozprzestrzeniania fałszywych informacji.

Ponieważ, analiza powiązań między wpisami przypomina analizę grafu, do tego zadania wykorzystano bibliotekę *NetworkX*.

4.2.5. Podsumowanie wykorzystanych bibliotek

Do analizy rozprzestrzeniania fałszywych informacji wykorzystano następujące biblioteki:

1. Plotly[17] - do rysowania wykresów
2. matplotlib[18] - do rysowania wykresów
3. NetworkX[19] - do budowania i rysowania grafów

[17] pypi.org/project/plotly/, dostęp: 22.05.2021
[18] pypi.org/project/matplotlib/, dostęp: 22.05.2021
[19] pypi.org/project/networkx/, dostęp: 22.05.2021

5. Eksperymenty

W tym rozdziale zostały przedstawione badania napisanego programu. Najpierw została zweryfikowana jakość klasyfikacji, a później zbadana została propagacja fałszywych informacji.

5.1. Badanie poprawności klasyfikatora fałszywych wiadomości

Badanie poprawności klasyfikatora polegało na treningu klasyfikatorów na kilku różnych zbiorach, następnie obliczeniu metryk opisanych w rozdziale 2.3.8 oraz porównaniu wyników z innymi pracami, które wykorzystywały te same zbiory.

5.1.1. Wykorzystane zbiory danych

Klasyfikacja została zbadana przy użyciu 3 różnych zbiorów danych. Kilka zbiorów zostało wykorzystanych, aby zwiększyć pewność co do słuszności wybranych cech, które są przekazywane do klasyfikatora, tak jako to zostało opisane w rozdziale 4.1.1.

Wykorzystane zostały następujące zbiory danych:

1. Zbiór 45 tysięcy fałszywych i prawdziwych artykułów. Zbiór zawiera głównie artykuły dotyczące wyborów w USA w 2016 roku[20].

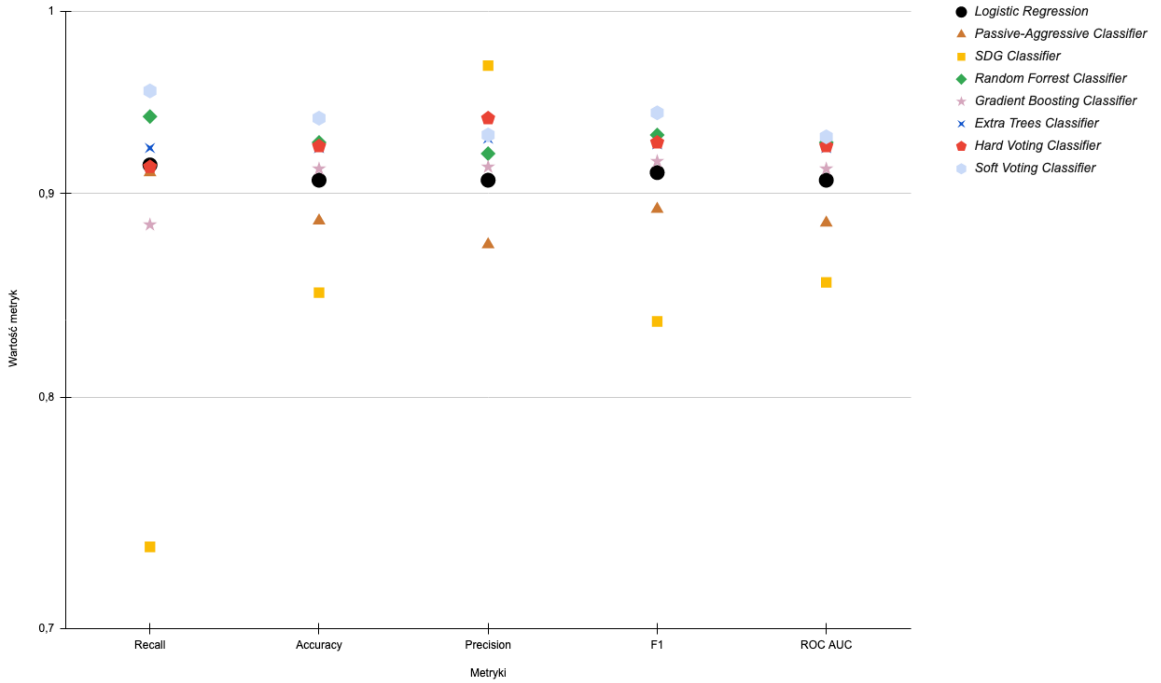
2. Zbiór 180 fałszywych i prawdziwych artykułów dotyczących wyborów w USA w 2016 roku[21].

3. Zbiór 25 tysięcy fałszywych i prawdziwych artykułów dotyczących wielu różnych tematów[22].

5.1.2. Wyniki klasyfikacji

Do określenia jakości klasyfikacji wykorzystano metryki opisane w rozdziale 2.3.8. Dla pierwszego zbioru danych wyniki były następujące:

[20]Fake and real news dataset, kaggle.com, dostęp: 15.05.2021
[21]FakeNewsNet, kaggle.com, dostęp: 15.05.2021
[22]Fake News Detection, github.com/rockash, dostęp: 15.05.2021



Rysunek 21: Wyniki poszczególnych klasyfikatorów na Zbiorze Danych 1.

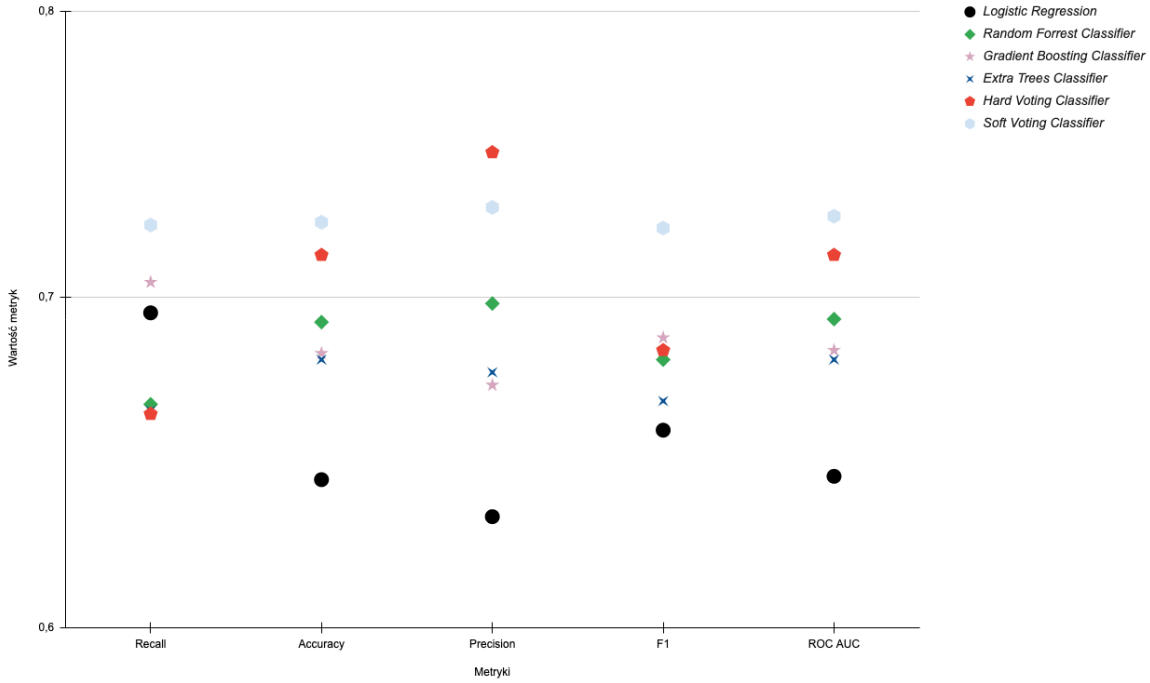
Większość badanych klasyfikatorów otrzymała wyniki powyżej 0.90, wyjątkiem są *SDG Classifier* oraz *Passive-Aggresive Classifier*.

Mimo tego, że precyzja klasyfikatora *SDG* wyniosła rekordowe 0.97, to uznałem, że algorytm poradził sobie najgorzej spośród badanych algorytmów, ze względu na przeważające niskie wyniki klasyfikatora w innych metrykach. Precyzja na tak wysokim poziomie oznacza, że program na ogół nie mylił się na zadanym zbiorze, ale nie potrafił wykryć zadowalającej liczby przypadków. Daleko mu także do klasyfikatora idealnego, który według miary ROC AUC powinien wynosić 1. Z tego powodu algorytm został wyeliminowany z dalszych badań.

Drugi najgorszy wynik należy do klasyfikatora *pasywno-agresywnego*, którego wyniki we wszystkich metrykach mieszczą się w przedziale od 0.87 do 0.91. Ten algorytm również został wyeliminowany z dalszych rozważań.

Pozostała grupa algorytmów uzyskała wyniki bardzo zbliżone do siebie, ale znacznie lepsze od wyeliminowanych klasyfikatorów. Na pierwszym zbiorze danych, najlepsze wyniki uzyskał *Soft Voting Classifier*. Takie wyniki pokazują, że łączenie kilku klasyfikatorów (nawet takich o niższej jakości) ma sens, ponieważ razem są w stanie przewyższać inne klasyfikatory.

Drugi zbiór był wykorzystany do treningu klasyfikatorów z wyłączeniem klasyfikatora *SDG* i klasyfikatora *pasywno-agresywnego*.



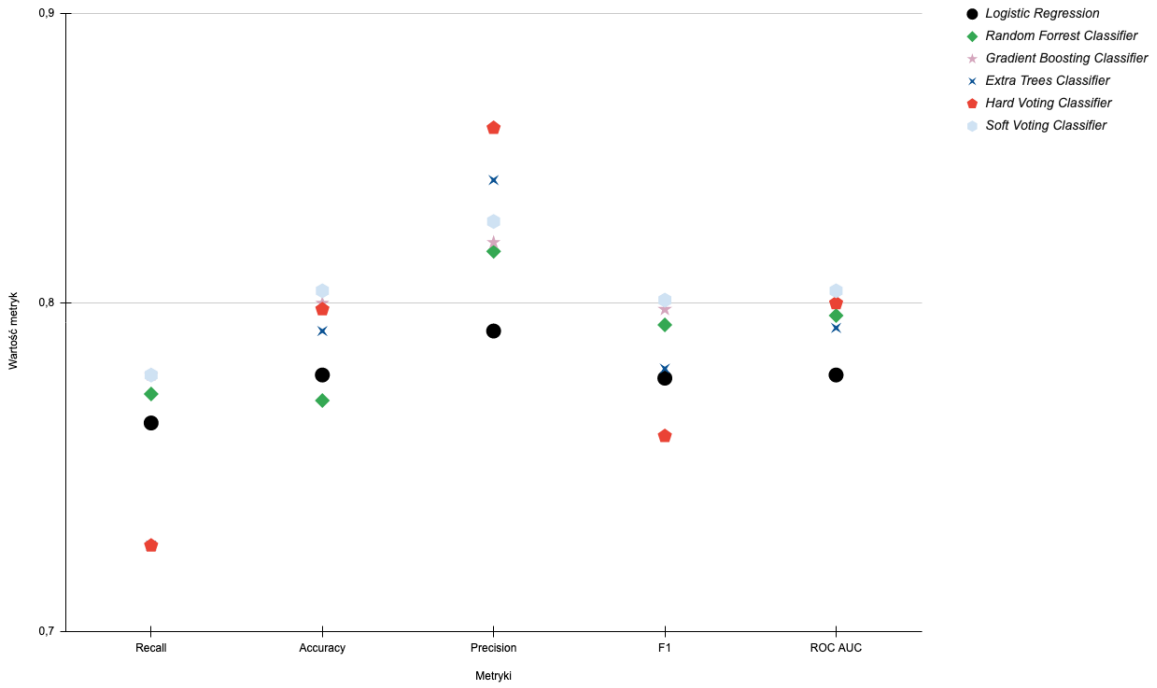
Rysunek 22: Wyniki poszczególnych klasyfikatorów na Zbiorze Danych 2.

Ponieważ drugi zbiór danych był mało liczny, konieczne było wykorzystane mechanizmu *k-krotnej walidacji*. Wyniki przedstawione na rysunku 22 są średnim wynikiem pięciokrotnej walidacji.

Wykorzystanie tak małego zbioru znacznie wpłynęło na wyniki wszystkich klasyfikatorów, ponieważ ich metryki były niższe o około 0.2 w stosunku do poprzedniego zbioru danych. Mimo tego, zauważalne są pewne różnice między wydajnością poszczególnych algorytmów. W tym przypadku, regresja logistyczna poradziła sobie najgorzej z zadaniem detekcji fałszywych informacji, jednak nie została wyeliminowana ze zbioru klasyfikatorów ze względu na ogólne trudności wszystkich algorytmów w uchwyceniu natury fałszywych informacji.

W drugim badaniu, ponownie najlepsze wyniki uzyskał klasyfikator zbudowany z regresji logistycznej, lasu losowego, gradient boostingu oraz extra drzew losowych wykorzystujący średnią prawdopodobieństw klas na przestrzeni wszystkich klasyfikatorów (*Soft Voting Classifier*). Jedyną metryką, w której *Soft Voting Classifier* przegrywa z innym klasyfikatorem jest precyzja, tyle, że klasyfikator przegrywa z *Hard Voting Classifier* czyli inną metodą polegającą na składaniu głosów kilku słabszych klasyfikatorów i wydawaniu ostatecznej decyzji w głosowaniu większościowym. Po raz kolejny udowodnione zostało, że metody wykorzystujące kilka klasyfikatorów potrafią dobrze radzić sobie z zadaniem wykrywania fałszywych informacji, nawet mimo tego, że w skład głosowania wchodzi klasyfikator o niskich wynikach - regresja logistyczna.

Ostatni przetestowany zbiór był znacznie większy od drugiego, ale niemal o połowę mniejszy od pierwszego, dodatkowo zawierał artykuły o różnej tematyce.



Rysunek 23: Wyniki poszczególnych klasyfikatorów na Zbiorze Danych 3.

W tym przypadku, klasyfikator wykorzystujący głosowanie większościowe zaliczył duże spadki, z powodu niedostatecznej jakości działania extra losowych drzew, spowodowało to znaczny spadek czułości, co następnie przełożyło się na miarę F1, mimo wysokiej precyzji. *Soft Voting Classifier* pozostał liderem mimo gorszych wyników precyzji. Tutaj został wyprzedzony przez *Hard Voting Classifier* oraz ekstra drzewa losowe. Regresja logistyczna okazała się mieć lepszą dokładność od lasu losowego oraz lepsza czułość od *Hard Voting Classifier*, ale ma najgorszą precyzję i miarę ROC AUC.

Zbadanie klasyfikatorów na trzech różnych zbiorach pokazało następujące rzeczy:

1. metody łączące kilka klasyfikatorów o dostatecznie wysokiej jakości skutkuja uzyskaniem klasyfikatora o jeszcze wyższej jakości

2. głosowanie oparte o wyliczaniu średniej prawdopodobieństwa jest lepsze od głosowania większościowego jeśli chodzi o metryki *recall*, *accuracy*, *F1* oraz *ROC AUC*.

3. niskie wyniki na drugim zbiorze danych są wynikiem niedostatecznego rozmiaru zbioru danych względem długości wektora cech

5.1.3. Wynik klasyfikacji na tle innych istniejących rozwiązań

Wykorzystane zbiory były używane przez dużą grupę naukowców, ale także amatorów uczenia maszynowego. Na samym serwisie Kaggle, Zbiór Danych 1, ma 266 przykładowych rozwiązań[23]. Okazuje się, że bardzo wiele z nich osiągnęło *accuracy* na poziomie bliskim 100%. Przeważają metody oparte na sieciach neuronowych, które trenowane są na wektorach stworzonych przy użyciu TF-IDF (opisanych w rozdziale 2.2.4). Ale jak się okazuje nie potrzeba używać sieci neuronowych, aby uzyskać takie wyniki. Istnieje praca[24], która również uzyskała wysokie wskazania metryk przy użyciu Random Forrest Classifier wraz z metodą TF-IDF.

Rozwiązanie	**Recall**	**Accuracy**	**Precision**	**F1**
Istniejące	0.99	0.99	1.0	0.99
Proponowane	0.94	0.92	0.92	0.93

Tabela 1: Porównanie wartości metryk dla Random Forrest Classifier

W powyższej tabeli zebrane zostały wyniki istniejącej pracy oraz rozwiązania proponowanego w tej pracy, dla tego samego algorytmu. Okazuje się ze zbiór zaproponowanych cech przekazany do Random Forrest Classifier otrzymuje gorsze wyniki od TF-IDF. Różnice wynoszą od 5 do 8% między poszczególnymi metrykami. Nie jest to różnica duża, ale jest to różnica zauważalna. Przy tak wysokich wynikach liczy się każdy punkt procentowy. Gdyby rozwiązanie miało być wdrażane w środowisku produkcyjnym, proponowane rozwiązanie musiałoby zostać odrzucone, ze względu na dostępność lepszego podejścia.

Drugi zbiór danych, był zbiorem znacznie mniej licznym od poprzedniego. Ponieważ gromadzenie danych do treningu jest zadaniem czasochłonnym i kosztownym, idealne klasyfikatory powinny mieć zdolność generalizacji dla mniejszej liczby przykładów. Spośród 6 zgłoszonych rozwiązań na serwisie Kaggle[25], najlepsze wyniki uzyskało podejście zaproponowane przez Kumud'a Chauhan'a[26]. Poniżej zostały przedstawione wyniki autora na algorytmach wytrenowanych przy użyciu wektorów stworzonych przy użyciu techniki TF-IDF. Niestety autor nie wykorzystał techniki k-krotnej walidacji do określenia jakości swoich wyników, dlatego wyniki zostały obliczone wprost.

Rozwiązanie	**Accuracy**
Istniejące	0.67
Proponowane	0.67

Tabela 2: Porównanie dokładności dla Logistic Regression

[23]Stan na 22.05.2021
[24]Fake News Classification using Random Forest, kaggle.com, dostęp 22.05.2021
[25]Stan na 22.05.2021
[26]Exploratory Analysis and fake news classification on Buzzfeed News, kaggle.com, dostęp: 22.05.2021

Autor w swoich rozważaniach kierował się tylko metryką dokładności. W powyższej tabeli zebrane zostały wartości metryki dla istniejącego rozwiązania i rozwiązania proponowanego. Okazuje się, że dla tego zbioru danych i tego samego algorytmu wartość metryki wynosi tyle samo.

Rozwiązanie	**Accuracy**
Istniejące	0.80
Proponowane	0.78

Tabela 3: Porównanie dokładności dla Random Forrest Classifier

Autor zbadał również dokładność swojego klasyfikatora na algorytmie Random Forrest Classifier. W tym przypadku dokładność proponowanego rozwiązania jest niższa o 2.56%. W przypadku drugiego algorytmu, wyniki są nieco lepsze, ale nie jest to znaczna różnica jak na poprzednim zbiorze danych.

Warte zaznaczenia jest również to, że autor przeprowadzał eksperymenty dla innych cech, a wyniki dla TF-IDF były wynikami najlepszymi. Autor wypróbował m.in. najpopularniejsze wyrażenia jako wektor cech, ale nie uzyskał satysfakcjonujących wyników (wartość metryki *accuracy* od 0.55 do 0.75 dla różnych algorytmów), co ostatecznie doprowadziło go do wybrania TF-IDF jako najlepszego podejścia transformacji tekstu do reprezentacji liczbowej, dającego najwyższą dokładność.

Trzeci zbiór danych pochodził z serwisu GitHub i został dostarczony wraz z gotowymi algorytmami[27]. Niestety, autor wykorzystał całkiem odmienne algorytmy do problemu klasyfikacji od tych zaproponowanych w tej pracy.

Model	**Accuracy**
Naive Bayes	0.72
SVM	0.88
LSTM	0.94

Tabela 4: Wyniki istniejącego klasyfikatora

Algorytmy jakich użył autor wraz z wynikami dokładności zostały ujęte w powyższej tabeli. Najniższa wartość metryki została uzyskana dla algorytmu Naive Bayess, który uzyskał dokładność na poziomie 0.72. W przypadku proponowanego rozwiązania, najniższa wartość metryki *accuracy* wyniosła 0.79 dla algorytmu Random Forrest Classifier.

Najwyższa wartość metryki w proponowanym rozwiązaniu wynosi 0.80 i jest niższa od kolejnego algorytmu, który wykorzystał autor w swoim podejściu, gdzie metryka wyniosła 0.88 dla algorytmu SVM. Jednak mimo wszystko najlepsze wyniki uzyskała

[27]Fake News Detection, github.com/rockash, dostęp: 22.05.2021

sieć neuronowa LSTM. Daje to pewien pogląd na to, że dla tego samego wektora cech, możliwe jest uzyskanie lepszych wyników przez podmianę algorytmów co również zostało pokazane w tej pracy. Dodatkową informacją płynącą z tej pracy jest to, że dalszym kierunkiem rozwoju mojej pracy może być wykorzystanie sieci neuronowej, w celu sprawdzenia czy uda uzyskać się jeszcze lepsze wyniki.

Porównanie proponowanego podejścia do istniejących rozwiązań pokazało następujące rzeczy:

1. Zaproponowany wektor cech oraz użycie algorytmów dało wyniki zbliżone do istniejących rozwiązań, ale nie udało się zbudować podejścia znacznie lepszego.

2. Porównywanie różnych rozwiązań jest kłopotliwe, różni autorzy wykorzystują różne algorytmy, wykorzystują tylko niektóre metryki. Z tego powodu w tej pracy wykorzystano szereg metryk oraz kilka algorytmów, aby ułatwić czytelnikom łatwiejsze porównanie wyników.

3. Różne zbiory danych mają różną trudność co znacznie wpływa na wartości metryk. Przykładowo zbiór pierwszy jest bardzo monotematyczny co pozwala na uzyskiwanie wartości metryk bliskich 1, ale zbiory drugi i trzeci na takie wyniki nie pozwalają, ponieważ są zbiorami znacznie trudniejszymi (mniejszy rozmiar, większa różnorodność tematów).

5.1.4. Liczba bigramów, a dokładność klasyfikacji

Przeprowadzone badania wykazały, że cecha *10 najpopularniejszych bigramów* jest najważniejszą cechą w wektorze cech.

Rozwiązanie	**Recall**	**Accuracy**	**Precision**	**F1**	**ROC**
Wszystkie cechy	0.94	0.92	0.92	0.93	0.92
Tylko TOP10 bigramów	0.93	0.92	0.92	0.92	0.92
Cechy bez TOP10 bigramów	0.87	0.86	0.86	0.87	0.86

Tabela 5: Porównanie wartości metryk dla Random Forrest Classifier na pierwszym zbiorze danych dla różnych cech

Nie jest to wielkim zaskoczeniem, ponieważ cecha ta zajmuje aż 98% długości całego wektora cech. Ale jeśli cecha zajmuje tyle miejsca można pomyśleć nad ewentualnym ograniczeniem rozmiaru wektora lub wręcz przeciwnie, jego zwiększeniem w nadziei, że poprawi to wyniki. W poniższej tabeli zostały ujęte wyniki różnej liczby bigramów w wektorze cech, poza bigramami w wektorze znajdowały się pozostałe cechy.

Rozwiązanie	Recall	Accuracy	Precision	F1	ROC
TOP05 bigramów	0.92	0.91	0.91	0.92	0.91
TOP10 bigramów	0.94	0.92	0.92	0.93	0.92
TOP15 bigramów	0.94	0.92	0.92	0.93	0.92

Tabela 6: Porównanie wartości metryk dla Random Forrest Classifier na pierwszym zbiorze danych dla różnej liczby bigramów

Jak się okazuje zmniejszenie liczby bigramów pociąga za sobą nieznaczny spadek wartości wszystkich metryk (różnica o 0.01). Z kolei zwiększenie liczby bigramów o 5, nie niesie za sobą żadnych dodatkowych korzyści pod względem metryk. Zwiększenie liczby bigramów dodatkowo niepotrzebnie zwiększa wymagania pamięciowe programu.

Analiza doboru liczby bigramów pokazała następujące rzeczy:

1. Najpopularniejsze bigramy to najważniejsza cecha w wektorze.
2. Same bigramy pozwalają na osiągnięcie wysokich wskazań metryk.
3. Dodatkowe cechy pozwalają na podniesienie i tak stosunkowo wysokich wartości metryk.
4. 10 najpopularniejszych bigramów to rozsądne optimum, stosowanie większej liczby nie ma uzasadnionego sensu. Możliwe jest rozważenie mniejszej liczby bigramów jeżeli zajętość pamięci stanie się problemem.
5. Dalsze rozważania mogą zostać skupione na analizie rozmiaru n-gramów, na przykład zastosowanie trigramów [12].

5.2. Badanie propagacji fałszywych informacji

Badanie propagacji odbywało się poprzez analizę współczynnika R_0 tak jak zostało to opisane w rozdziale 3.2 oraz analizę propagacji i rozprzestrzeniania przy użyciu metod graficznych.

5.2.1. Wykorzystane zbiory danych

W internecie brakuje dobrych zbiorów obrazujących propagację fałszywych i prawdziwych informacji. Zasadniczo istnieje jeden zbiór danych, na którym oparte jest wiele prac. Zbiór ten jest odpowiednio wzbogacany w zależności od potrzeb badań. Do analizy propagacji został wykorzystany wspomniany zbiór kilkunastu wpisów pochodzących z Twittera [16] bez dodatkowych modyfikacji. Zbiór również został opisany w rozdziale 4.2.1. To na nim został zbadany został współczynnik R_0 oraz rozpatrzona została propagacja w czasie i przestrzeni za pomocą metod wizualizacji.

5.2.2. Analiza współczynnika R_0

W rozdziale 3.2 zostały opisane 2 warianty badania współczynnika R_0. Pierwszy z nich zakładał branie pod uwagę wszystkie obiekty, bez rozróżnienia czy przekazały dalej wpis.

Rodzaj informacji	$R_0 Mean$
Fałszywa	0.99
Prawdziwa	0.99

Tabela 7: Porównanie wartości R_0 dla pierwszego wariantu

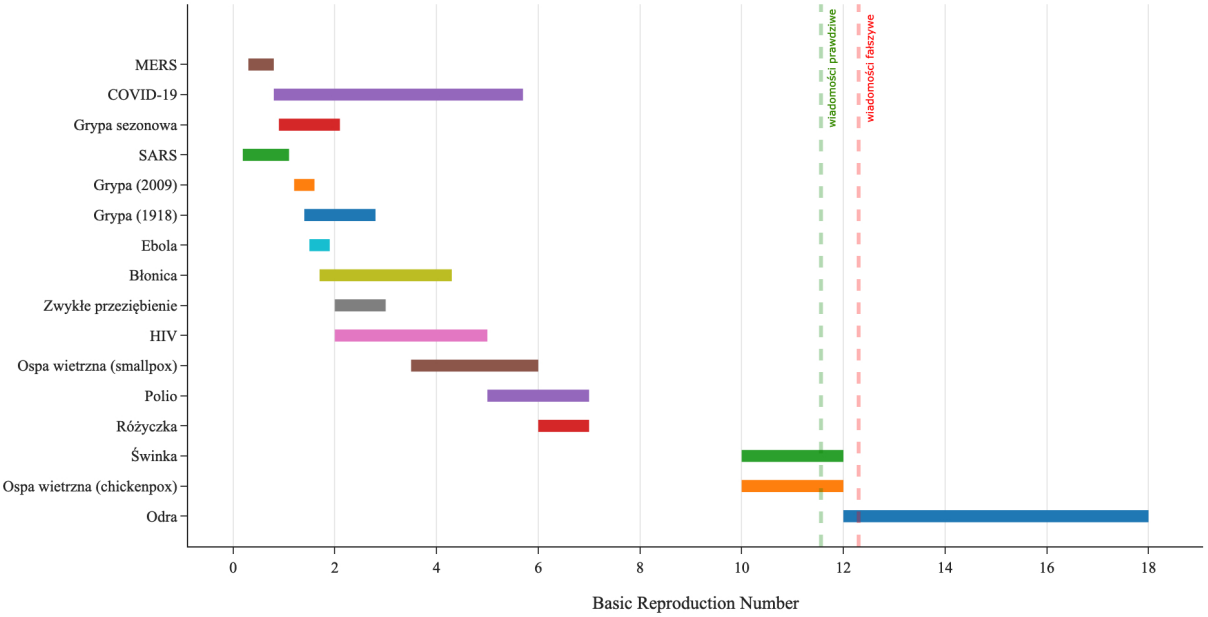
Jak zostało to pokazane na powyższym obrazku, średnie wartości R_0 dla fałszywej i prawdziwej informacji wyniosły dokładnie tyle samo. Powoduje to, że nie można wysnuć żadnych wniosków co do różnic w propagacji fałszywych i prawdziwych informacji względem propagacji epidemii. Taka wartość współczynnika jest wynikiem tego, że w zbiorze danych przeważająca większość jednostek nie była źródłem nowych podań dalej.

Z tego powodu powstał drugi wariant analizy, który pomija jednostki nie będących źródłem dalszych podań dalej.

Rodzaj informacji	$R_0 Mean$
Fałszywa	12.30
Prawdziwa	11.56

Tabela 8: Porównanie wartości R_0 dla drugiego wariantu

W powyższej tabeli widoczne jest to, że średnie wartości współczynnika R_0 różnią się od siebie, a naniesione na wykres 24 kształtują się następująco:



Rysunek 24: Wykres wartości R_0 dla różnych wirusów/chorób wraz ze średnimi wartościami R_0 dla fałszywych i prawdziwych informacji

Wartości choć są zbliżone to pokrywają się z odmiennymi chorobami. Pozwala to na wnioskowanie, że dla wykorzystanego zbioru, biorąc pod uwagę jednostki, które są źródłem rozprzestrzeniania, fałszywe wiadomości swoim rozprzestrzenianiem przypominają ordę, a prawdziwe wiadomości swoim rozprzestrzenianiem przypominają świnkę lub ospę wietrzną.

5.2.3. Analiza propagacji wiadomości w czasie

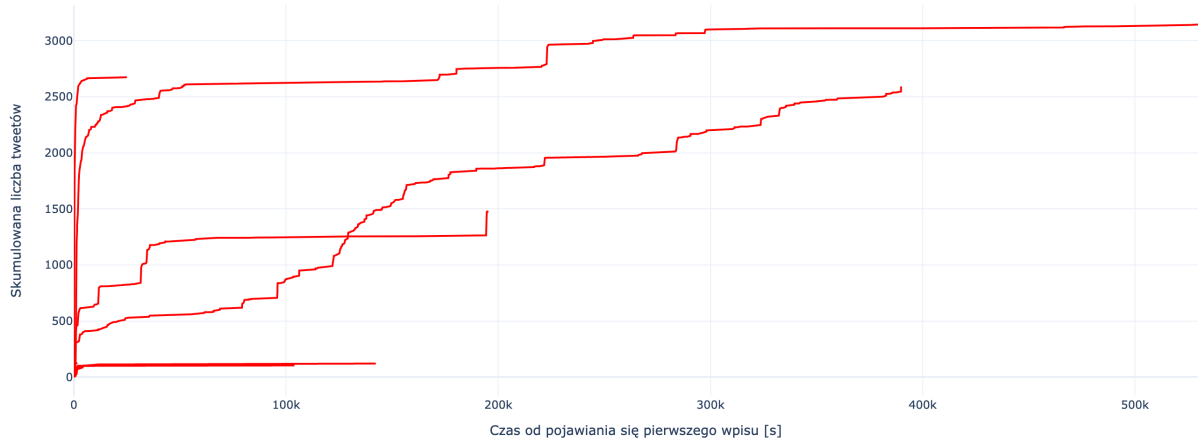
Analiza rozprzestrzeniania może być postrzegana również jako analiza propagacji w czasie.

Rysunek 25: Wykres propagacji prawdziwych informacji czasie.

Na powyższym wykresie przedstawiona została propagacja w czasie informacji prawdziwych. Prawdziwe tweety dotyczyły przede wszystkim legalizacji małżeństw homoseksual-

nych w USA oraz zamachu terrorystycznego na Charlie Hebdo w Paryżu. Przedstawiona reprezentatywna próbka charakteryzuje się niemal pionowymi wzrostami w początkowej fazie życia po którym następuje wypłaszczenie. Oznacza to początkowe zainteresowanie tematem. Gdy pojawia się nowa informacja następuje duże zaangażowanie użytkowników, które po czasie słabnie, gdy informacja zostaje przyswojona i staje się powszechnie znana. Krzywe wykresu propagacji prawdziwej informacji w badanym zbiorze, niemal zawsze są asymptotyczne.

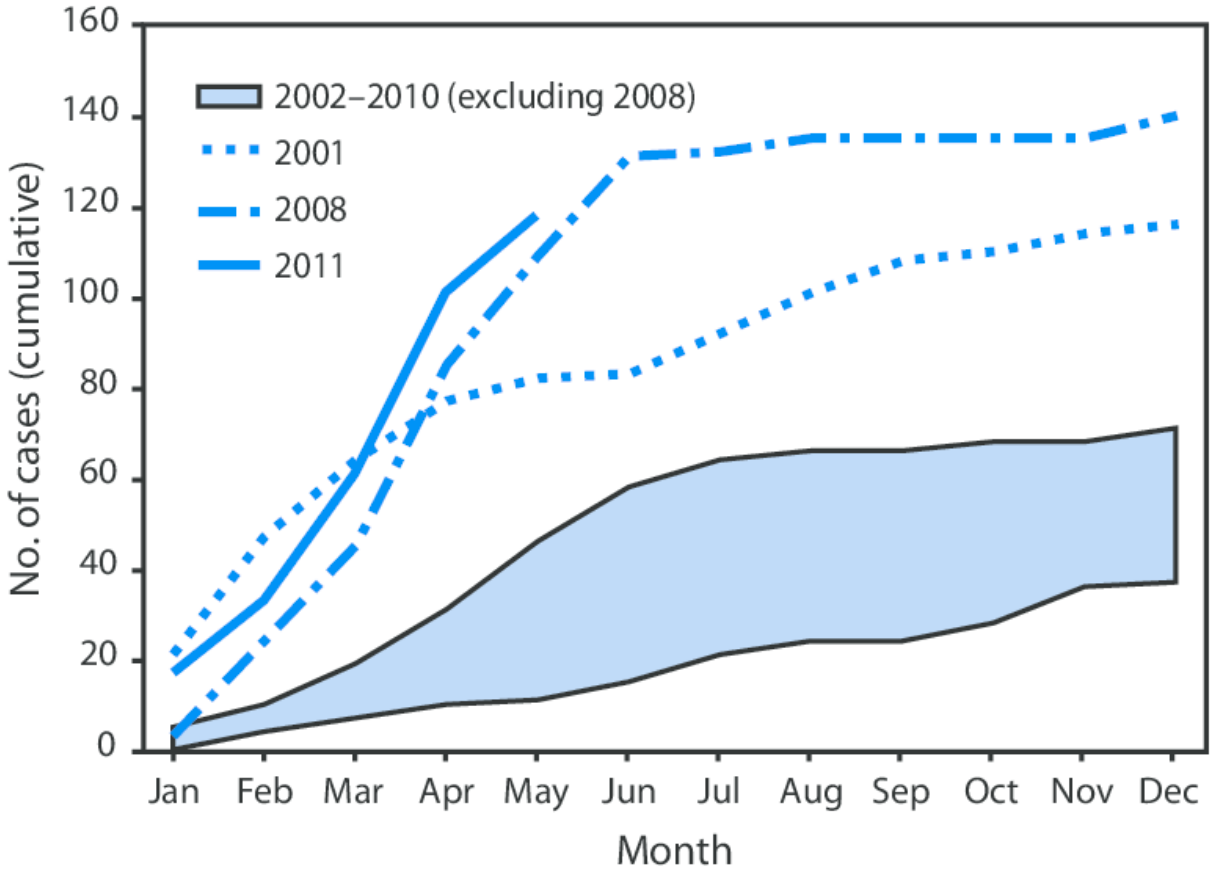
Inaczej wygląda propagacja fałszywych informacji:



Rysunek 26: Wykres propagacji fałszywych informacji czasie.

Porównując powyższy wykres z wykresem wcześniejszym zauważalne są następujące różnice:

- Fałszywe informacje są w stanie propagować się wśród większej liczby użytkowników.
- Charakterystyka wzrostów jest inna, wzrosty propagacji fałszywych informacji są w stanie przyjmować postać funkcji asymptotycznych, ale też niemal liniowych.
- W propagacji fałszywych informacji widoczne są pionowe ściany, które w przypadku propagacji informacji prawdziwych są znacznie mniejsze. Takie zjawisko może wynikać z uruchomienia botów, które generują sztuczne podania dalej, lub do propagatorów dołączają popularne konta, które skupiają zwolenników danej informacji, przez co informacja w pewien sposób odżywa i odnotowuje kolejne wzrosty, podobne do wzrostów początkowych.



Rysunek 27: Wykres propagacji odry w USA

Jeżeli w rozdziale 5.2.2 przyjęto, że propagacja fałszywych informacji przypomina odrę pod względem współczynnika R_0, to można porównać propagację fałszywych informacji w czasie do propagacji odry w czasie[28].

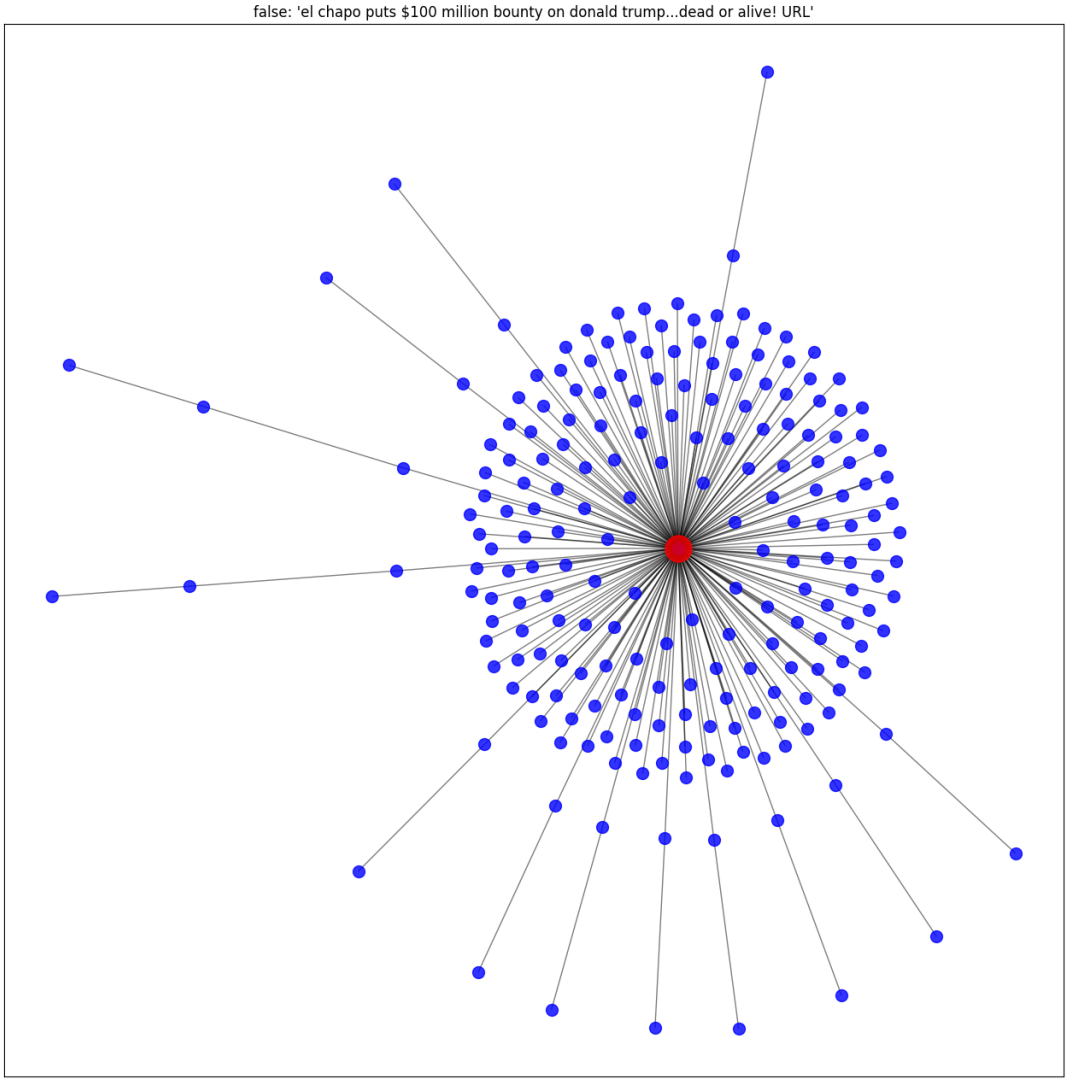
Niestety ani osie X ani osie Y nie współgrają ze sobą, ale uważnie obserwując kształt krzywych, można stwierdzić pewne podobieństwa.

- Zachorowania w 2008 mają przebieg asymptotyczny podobny do niektórych fałszywych, ale też prawdziwych informacji (różnica R_0 dla informacji prawdziwych i fałszywych nie jest znaczna).
- Zachorowania w 2001 mają łagodny stopniowy wzrost, podobny do wzrostu propagacji niektórych fałszywych informacji. W przypadku informacji prawdziwych nie zaobserwowano takich wzrostów.

[28]Measles Cases and Outbreaks, cdc.gov, dostęp: 22.05.2021

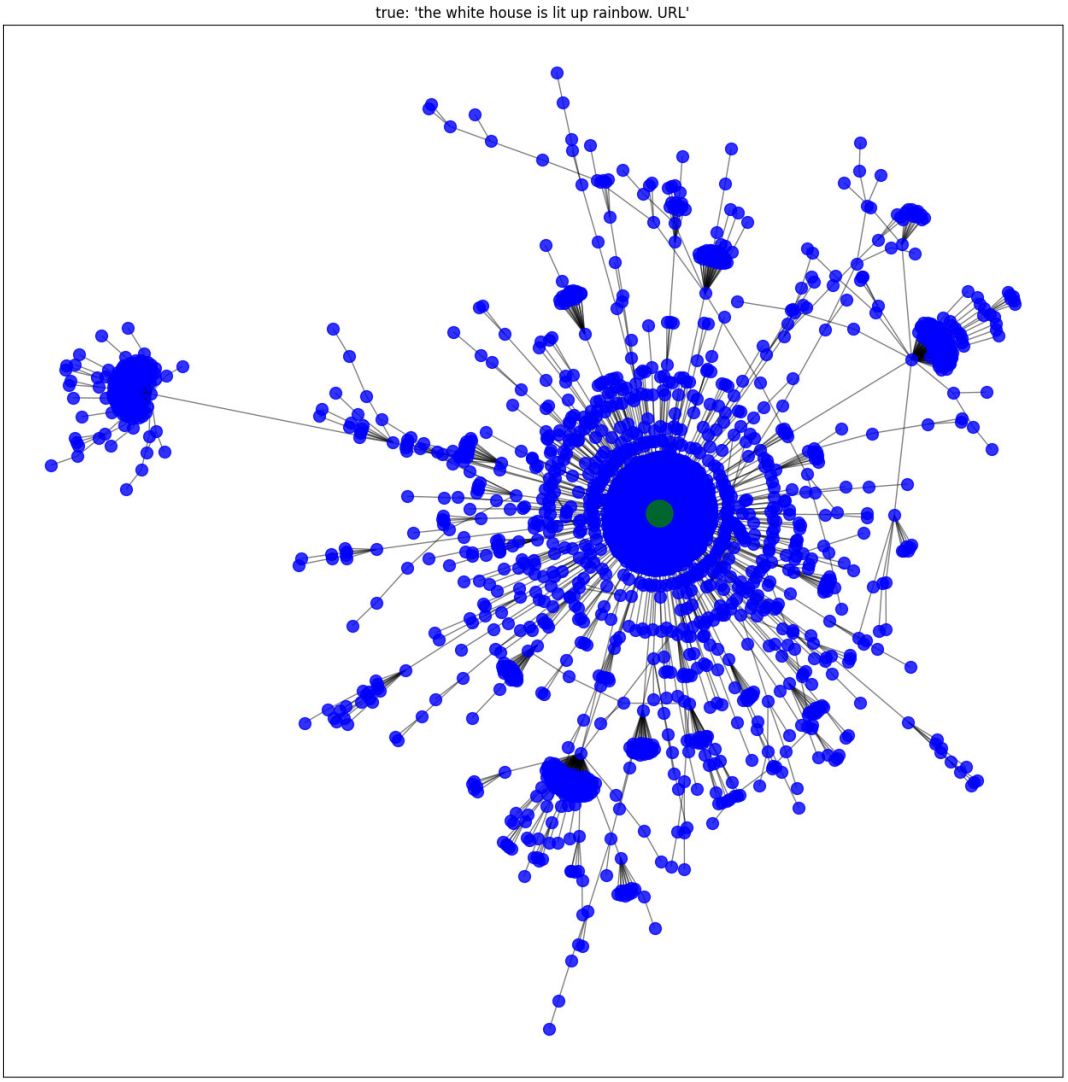
5.2.4. Analiza propagacji wiadomości w przestrzeni

Analiza propagacji wiadomości w przestrzeni opisana w rozdziale 3.2, może zostać użyta do zidentyfikowania źródeł, które są odpowiedzialne za dalsze przekazywanie informacji. W epidemiologii odpowiadałoby to odnajdywaniu głównych ognisk nowych zakażeń.



Rysunek 28: Wykres propagacji przykładowej fałszywej informacji w przestrzeni

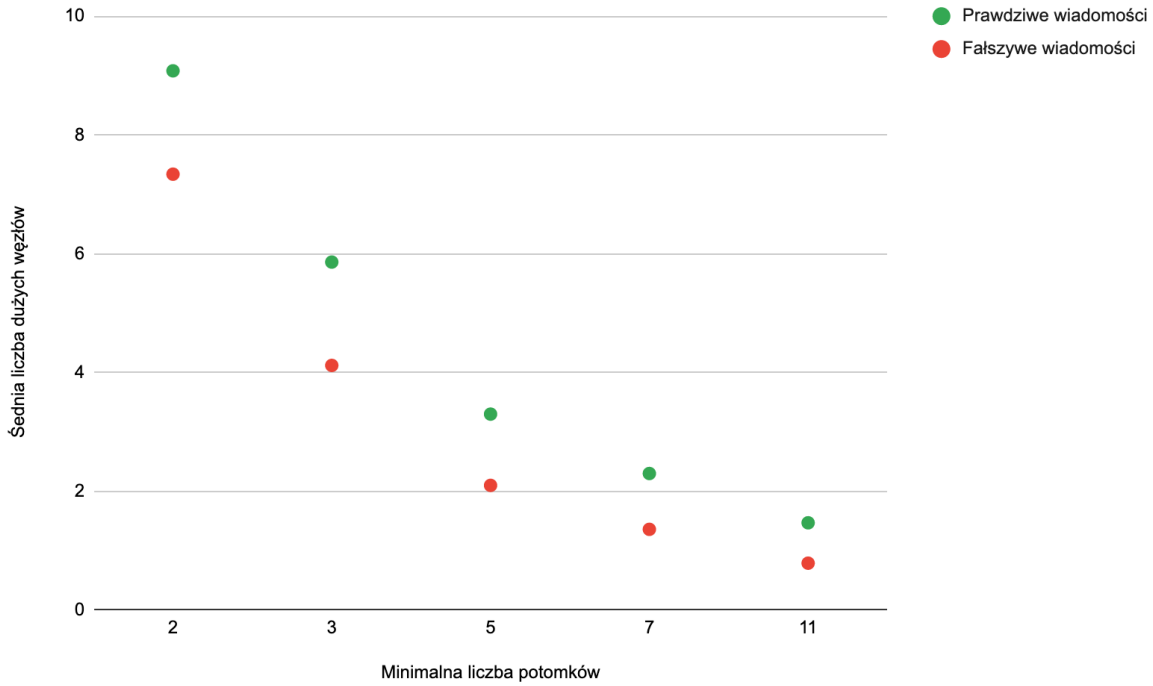
Na powyższym wykresie została zobrazowana propagacja przykładowego fałszywego wpisu w przestrzeni. Na czerwono zostało zaznaczone główne źródło, miejsce, od którego zaczęła się propagacja informacji. Jak widać ten konkretny przykład posiada jedno główne źródło, z którego wychodzą inne węzły (podania dalej).



Rysunek 29: Wykres propagacji przykładowej prawdziwej informacji w przestrzeni

Z kolei na powyższym rysunku przedstawiona została propagacja w przestrzeni przykładowej prawdziwej informacji. Jak widać ta informacja była znacznie częściej podawana dalej, ponadto istnieje kilka węzłów o dużym zagęszczeniu potomków. To oznacza, że główna prawdziwa informacja (zaznaczona na zielono) została podana dalej przez popularne konta, następnie podania dalej głównego wpisu zostały podane dalej przez kolejnych użytkowników.

Aby odpowiedzieć na pytanie ile jest średnio dużych węzłów dla wpisów prawdziwych i fałszywych, konieczne jest narzucenie minimalnej liczby potomków danego węzła. Przetestowane zostało kilka wartości minimalnych - 2, 3, 5, 7 i 11 (pierwsze pięć liczb pierwszych).



Rysunek 30: Wykres średniej liczby głównych węzłów w zależności od minimalnej liczby potomków.

Powyższy wykres wskazuje, że fałszywe wiadomości cechują się mniejszą liczbą dużych węzłów, a prawdziwe wiadomości cechują się większą liczbą dużych węzłów.

Oznacza to, że fałszywe informacje są propagowane głównie przez pojedyncze konta, które są obserwowane przez mało wpływowych użytkowników. Przykładowo może to oznaczać wykorzystanie botów (farmy trolli, które z reguły mają mało ludzkich obserwujących) do propagowania wiadomości.

W przypadku prawdziwych informacji, większa ilość dużych węzłów może wskazywać na zwiększoną rzetelność wykorzystywanych kont do dalszej propagacji, np. konta dziennikarzy czy agencji prasowych.

Niestety porównanie liczby dużych węzłów do ognisk epidemii nie było możliwe przez brak informacji ile osób zostaje zakażonych w ogniskach epidemii.

6. Podsumowanie

W tym rozdziale zostały podsumowane wyniki obu części pracy oraz wyznaczone dalsze kierunki rozwoju.

Aby umożliwić dalszy rozwój prac, cały kod pracy oraz zbiory danych zostały umieszczone w repozytorium w serwisie GitHub[29].

6.1. Klasyfikacja fałszywych informacji

Po dogłębnym przeanalizowaniu istniejących rozwiązań, zdecydowano się na budowę klasyfikatora opartego na ekstrakcji cech z tekstu przy użyciu technik przetwarzania języka naturalnego. Analizowane były różne parametry tekstu, takie jak zliczanie części mowy, wyznaczanie najpopularniejszych zwrotów, określanie sentymentu czy obiektywności. Wybrane podejście pozwoliło na zgłębienie wiedzy w zakresie przetwarzania języka naturalnego oraz uczenia maszynowego przez zastosowanie szeregu algorytmów klasyfikacji, a także zaznajomienie się z różnymi metrykami opisującymi jakość algorytmu.

Przygotowanie programu do działania na różnych zbiorach danych wymagało zademonstrowania znajomości dobrych technik projektowania oprogramowania, a także umożliwiło porównanie rozwiązania z innymi istniejącymi podejściami.

W kwestii wyboru algorytmu klasyfikacji, najważniejszym wnioskiem płynącym z pracy jest to, że metody agregujące kilka klasyfikatorów w jeden dają bardzo dobre rezultaty. Jeżeli mamy do dyspozycji kilka klasyfikatorów o wysokich wskazaniach metryk, możliwe jest jest ich połączenie w jeden klasyfikator agregujący, który jako wynik głosowania zależy od średniej prawdopodobieństw przypisania do danej klasy (Soft Voting Classifier). Wtedy możliwe jest uzyskanie klasyfikatora o jeszcze większej skuteczności.

Porównując rozwiązanie zaproponowane w tej pracy do rozwiązań istniejących, okazało się, że udało się uzyskać klasyfikator o zbliżonej jakości (wskazania metryk bliskie lub niższe o kilka procent względem istniejących rozwiązań). Nie udało się natomiast zaproponować rozwiązania znacznie lepszego. Dodatkowym wnioskiem płynącym z porównywania prac jest trudność w wykonaniu tego zadania, różni autorzy wykorzystują różne podzbiory metryk lub inaczej podchodzą do kwestii walidacji (np. brak k-krotnej walidacji). Aby umożliwić łatwe porównywanie innych prac z tą pracą, zostały przedstawione wskazania aż pięciu różnych metryk, tak aby przyszli czytelnicy mogli porównać wyniki w zależności od swoich potrzeb.

Dalszy rozwój prac powinien zostać ukierunkowany na wykorzystanie sieci neuronowych do zadania klasyfikacji, zastosowania trigramów zamiast bigramów w wektorze cech, a także propozycji nowych cech dla wektora cech.

[29] github.com/pkardas/agh-fake-news, dostęp: 23.05.2021

6.2. Propagacja fałszywych informacji

Druga część pracy polegała na analizie propagacji fałszywych informacji i porównaniu tego rozprzestrzeniania do rozprzestrzeniania się epidemii. Był to pewnego rodzaju eksperyment myślowy, który jak się okazało ma poparcie w rzeczywistych modelach matematycznych, które opisują propagację fałszywych wiadomości.

Do przeprowadzenia eksperymentu został wykorzystany jeden zbiór danych opisujący propagację wpisów w serwisie Twitter. Niestety brakuje większej liczby podobnych zbiorów, dlatego większość badań w tym zakresie oparta jest o ten zbiór danych.

Analiza współczynnika R_0 wykazała, że fałszywe wiadomości rozprzestrzeniają się niczym odra, a prawdziwe rozpowszechniają się podobnie do ospy i świnki. Jednak różnice we współczynnikach nie są znaczne.

Z kolei analiza propagacji w czasie faktycznie wykazała, że fałszywe wiadomości rozpowszechniają się inaczej od wiadomości prawdziwych. Porównanie propagacji fałszywych informacji do propagacji odry w czasie jest problematyczne ze względu na całkowicie odmienną charakterystykę danych.

Analiza propagacji w przestrzeni również wykazała, że prawdziwe i nieprawdziwe informacje rozprzestrzeniają się inaczej, biorąc pod uwagę liczbę ognisk (miejsc, gdzie nastąpiło wiele przekazań wpisu). Niestety brakuje danych na temat tego ile, przykładowo nowych zakażeń odry powstaje w ogniskach zakażeń, dlatego takie porównanie nie było możliwe.

W tej części, dalsze prace powinny być ukierunkowane na wykorzystanie bardziej zaawansowanych mechanizmów matematycznych w celu porównania rozprzestrzeniania fałszywych informacji z rozprzestrzenianiem na przykład odry. Dalsze prace mogłyby być także skupione na odnalezieniu lub zgromadzeniu informacji na temat liczby nowych zakażeń w ogniskach epidemii, aby umożliwić porównanie propagacji w przestrzeni.

Spis rysunków

Spis tabel

Literatura

[1] H. Allcott and M. Gentzkow. Social media and fake news in the 2016 election, 2017.

[2] L. BottouAT. Stochastic gradient learning in neural networks.

[3] J. Cao, P. Qi, Q. Sheng, T. Yang, J. Guo, and J. Li. Exploring the role of visual content in fake news detection. *CoRR*, abs/2003.05096, 2020.

[4] U. Chitra and C. Musco. Analyzing the impact of filter bubbles on social network polarization. *Conference: WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining*, 2020.

[5] L. Ehrlinger and W. Wöß. Ttowards a definition of knowledge graphs. In *Joint Proceedings of the Posters and Demos Track of 12th International Conference on Semantic Systems - SEMANTiCS2016 and 1st International Workshop on Semantic Change and Evolving Semantics*, 2016.

[6] X. Fang and J. Zhan. Sentiment analysis using product review data. 2015.

[7] M. Fudaliński and J. Sroka. Modele rozprzestrzeniania się epidemii, 2020.

[8] V. Gangadharan, D. Gupta, A. L., and A. T.A. Paraphrase detection using deep neural network based word embedding techniques. In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, pages 517–521, 2020.

[9] A. Geron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, 2017.

[10] A. Hanselowski, A. PVS, B. Schiller, F. Caspelherr, D. Chaudhuri, C. M. Meyer, and I. Gurevych. A retrospective analysis of the fake news challenge stance-detection task. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1859–1874, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics.

[11] L. Hasher, D. Goldstein, and T. Toppino. Frequency and the conference of referential validity. *Journal of Verbal Learning and Verbal Behavior*, 16(1):107–112, 1977.

[12] D. Jurafsky and J. H. Martin. *Speech and Language Processing (2nd Edition).* Prentice-Hall, Inc., USA, 2009.

[13] M. Karamibekr and A. A. Ghorbani. Sentence subjectivity analysis in social domains. 2013.

[14] M. Kumar and R. Vig. Term-frequency inverse-document frequency definition semantic (tids) based focused web crawler. Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[15] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis, 1998.

[16] J. Ma, W. Gao, P. Mitra, S. Kwon, B. J. Jansen, K.-F. Wong, and C. Meeyoung. Detecting rumors from microblogs with recurrent neural networks. In *The 25th International Joint Conference on Artificial Intelligence*. AAAI, 2016.

[17] B. Marwick. Discovery of emergent issues and controversies in anthropology using text mining, topic modeling, and social network analysis of microblog content. 2014.

[18] E. L. Meltzer. The grammar of fake news, 2020.

[19] M. Pawela. Ujawniamy: skala rosyjskiej dezinformacji o covid-19 w polsce. fakenews.pl. Dostęp: 03.04.2021.

[20] J. R. Piqueira, M. Zilbovicius, and C. M. Batistela. Daley–kendal models in fake-news scenario, 2019.

[21] P. Przybyła. Capturing the style of fake news. Institute of Computer Science, Polish Academy of Sciences, 2020.

[22] S. Robertson. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation - J DOC*, 60:503–520, 10 2004.

[23] B. Rosario. Latent semantic indexing: An overview, 2000.

[24] A. Sa, K. Hinkelmann, and F. Corradini. Combining machine learning with knowledge engineering to detect fake news in social networks-a survey. 03 2019.

[25] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.*, 19(1):22–36, Sept. 2017.

[26] C.-M. Tan, Y. Wang, and C.-D. Lee. The use of bigrams to enhance text categorization. 2002.

[27] S. Tschiatschek, A. Singla, M. Gomez Rodriguez, A. Merchant, and A. Krause. Fake news detection in social networks via crowd signals. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 517–524, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[28] M. Waszak. Postprawda i fake news czy weryfikacja treści i źródeł informacji, 2017.

[29] X. Zhou, A. Jain, V. V. Phoha, and R. Zafarani. Fake news early detection: A theory-driven model. *Digital Threats: Research and Practice*, 1(2), June 2020.

[30] X. Zhou and R. Zafarani. A survey of fake news. *ACM Computing Surveys*, 53(5):1–40, Oct 2020.

[31] X. Zhou and R. Zafarani. A survey of fake news: Fundamental theories, detection methods, and opportunities. Syracuse University, 2020.