Politechnika Opolska

Wydział Elektrotechniki, Automatyki i Informatyki

Katedra Informatyki

PRACA DYPLOMOWA

Studia drugiego stopnia stacjonarne

Kierunek Informatyka

METODYKI WYKRYWANIA FAKE NEWSÓW Z WYKORZYSTANIEM WYBRANYCH MODELI SZTUCZNEJ INTELIGENCJI

Promotor:
dr inż. Anna ZATWARNICKA

Autor:
inż. Oliwia ZAPART
nr albumu: 99517

Opole, lipiec 2024

METODYKI WYKRYWANIA FAKE NEWSÓW Z WYKORZYSTANIEM WYBRANYCH MODELI SZTUCZNEJ INTELIGENCJI

Streszczenie

Głównym celem pracy dyplomowej było opracowanie, implementacja oraz porównanie modeli sztucznej inteligencji w kontekście skutecznego rozpoznawania fake newsów. W ramach pracy zrealizowano zadania takie jak zdobycie wiedzy na temat metod wykrywania fake news, zapoznanie się z technikami przetwarzania języka naturalnego, przygotowanie i oczyszczenie zbiorów danych, wykorzystanie technik tokenizacji i osadzania słów, zapoznanie się z działaniem sieci neuronowych typu LSTM oraz modelem językowym BERT, implementacja tych modeli do klasyfikacji informacji, przeprowadzenie eksperymentów oraz analiza wyników, w tym dokładności, raportów klasyfikacji oraz macierzy pomyłek.

METHODOLOGIES FOR DETECTING FAKE NEWS USING SELECTED ARTIFICIAL INTELLIGENCE MODELS

Summary

The main objective of the thesis was to develop, implement, and compare artificial intelligence models for the effective detection of fake news. The tasks carried out in this work included acquiring knowledge on fake news detection methods, familiarizing with natural language processing techniques, preparing and cleaning datasets, examining word tokenization and embedding techniques, understanding the workings of LSTM neural networks and the BERT language model, implementing these models for information classification, conducting experiments, and analysing results, including accuracy, classification reports, and confusion matrices.

SPIS TREŚCI

SPIS RYSUNKÓW

SPIS LISTINGÓW

SPIS RÓWNAŃ

1. Wstęp

W ciągu ostatnich lat zaobserwowany został dynamiczny rozwój społeczeństwa informacyjnego, współtworzonego przez coraz bardziej złożony krajobraz mediów społecznościowych i internetowych źródeł informacji. Jednakże, równocześnie z tym postępem, pojawiło się zjawisko, które stworzyło poważne wyzwanie dla rzetelności informacji – fake newsy. Dezinformacja, czyli celowe rozpowszechnianie fałszywych lub zmanipulowanych informacji w celu wprowadzenia w błąd odbiorców, stała się powszechnym narzędziem wpływu na opinię publiczną, procesy polityczne i społeczne [Fake newsy, 2024].

W odpowiedzi na rosnące zagrożenie związane z dezinformacją, rozwój technologii sztucznej inteligencji otworzył nowe możliwości w zakresie wykrywania fake newsów. Modele AI, wykorzystujące zaawansowane techniki przetwarzania języka naturalnego, pozwalają na analizę treści medialnych i identyfikację fałszywych informacji. Dzięki zastosowaniu tych technologii możliwe jest automatyczne rozpoznawanie wzorców w danych tekstowych, co znacząco zwiększa skuteczność wykrywania dezinformacji.

Sztuczne sieci neuronowe, takie jak LSTM (z ang. Long Short-Term Memory) [LSTM, 2024] oraz modele językowe typu BERT (z ang. Bidirectional Encoder Representations from Transformers) [Google BERT, 2024], zyskują na popularności ze względu na swoją zdolność do analizy i klasyfikacji tekstu. Te zaawansowane technologie znajdują szerokie zastosowanie w przetwarzaniu języka naturalnego oraz umożliwiają tworzenie systemów, które mogą skutecznie identyfikować fałszywe informacje, a także ograniczać ich negatywny wpływ na społeczeństwo.

1.1 Cel pracy

Celem niniejszej pracy magisterskiej jest opracowanie, implementacja oraz porównanie modeli sztucznej inteligencji w kontekście skutecznego rozpoznawania fake newsów.

W ramach przeprowadzonych analiz i eksperymentów, praca skupi się na ocenie efektywności wybranych modeli – LSTM i BERT. Kluczowym punktem badawczym będzie dostarczenie kompleksowego spojrzenia na problematykę wykrywania dezinformacji, uwzględniając współczesne wyzwania informacyjne.

1.2 Zadania szczegółowe

Przed przystąpieniem do realizacji niniejszej pracy konieczne było zdobycie niezbędnej wiedzy oraz zaplanowanie działań, które pozwoliły na podzielenie całego projektu na mniejsze części, co umożliwiło bardziej efektywne wykonanie zadań:

- odnalezienie informacji na temat metod wykrywania fake news,
- zapoznanie się z technikami przetwarzania języka naturalnego (NLP – z ang. Natural Language Processing) [NLP, 2024],
- przygotowanie oraz oczyszczenie zbiorów danych zawierających prawdziwe i fałszywe wiadomości,
- zbadanie i wybór odpowiednich technik tokenizacji oraz osadzania słów,

- zapoznanie się z działaniem sieci neuronowych typu LSTM,
- zapoznanie się z modelem językowym BERT i jego zastosowaniami,
- implementacja modelu LSTM do klasyfikacji informacji,
- implementacja modelu BERT do klasyfikacji informacji,
- przeprowadzenie eksperymentów dla obu modeli w celu oceny ich efektywności w zależności od liczby epok,
- analiza wyników modeli, w tym dokładności, raportów klasyfikacji oraz macierzy pomyłek.

1.3 Układ pracy

Niniejsza praca dyplomowa składa się z dziewięciu rozdziałów, które zawierają ważne informacje na temat każdego elementu składowego wykonanego projektu.

W pierwszym rozdziale omówiono cel pracy, zadania szczegółowe oraz układ pracy, aby zapewnić czytelnikowi jasny obraz zakresu i struktury dokumentu.

Drugi rozdział wprowadza czytelnika w problematykę fake news, opisując oblicze problemu oraz współczesne rozwiązania stosowane w walce z dezinformacją.

W trzecim rozdziale przeprowadzono analizę wybranych modeli wykrywających fake news, co pozwala na zrozumienie ich zalet i wad.

Czwarty rozdział przedstawia narzędzia oraz technologie używane w projekcie.

Rozdział piąty skupia się na zbiorze danych, który został użyty w projekcie. Omówiono tu źródła danych, a także proces ich przygotowania.

W szóstym rozdziale opisano implementację modeli. Przedstawiono szczegółowy proces budowy modelu LSTM oraz modelu BERT, wraz z kodem i wyjaśnieniami.

Siódmy rozdział prezentuje działanie modeli. Omówiono tu metryki oceny, a także przeprowadzono wyniki eksperymentów klasyfikacji przy użyciu modelu LSTM oraz modelu BERT.

W ósmym rozdziale zawarto wnioski z przeprowadzonych eksperymentów, podsumowując skuteczność obu modeli.

Rozdział dziewiąty zawiera podsumowanie wykonanych prac, w którym opisano osiągnięcia pracy dyplomowej.

Odwołania literaturowe w niniejszej pracy zostały zapisane zgodnie z notacją harwardzką [Harvard, 2024].

2. Wprowadzenie

W niniejszym rozdziale przedstawiono, jak dynamiczny rozwój społeczeństwa informacyjnego oraz zjawisko fake newsów kreują poważne zagrożenie dla rzetelności informacji. Omówiono, jak dezinformacja wpływa na opinię publiczną, procesy polityczne i społeczne, a także wskazano kluczową rolę sztucznej inteligencji w identyfikowaniu oraz przeciwdziałaniu fałszywym informacjom.

2.1 Oblicze problemu

W erze Internetu i mediów społecznościowych problem fake newsów stał się znaczącym wyzwaniem dla rzetelności informacji oraz stabilności procesów politycznych. Szczególnie wyraźnie uwidoczniło się to podczas wyborów prezydenckich w USA w 2016 roku, które charakteryzowały się olbrzymim napływem fałszywych informacji (sfabrykowanych treści, zniekształconych raportów informacyjnych), udostępnianych głównie na platformach społecznościowych.

Badanie przeprowadzone przez Alexandre'a Boveta i Hernána A. Makse [Influence of fake news, 2024] dostarcza dogłębnej analizy wpływu dezinformacji na Twitterze w trakcie kampanii wyborczej. Wykorzystując zbiór danych obejmujący 171 milionów tweetów wysłanych przez 11 milionów użytkowników w ciągu pięciu miesięcy przed dniem wyborów, badacze zidentyfikowali 30,7 milionów tweetów zawierających linki do stron informacyjnych. Spośród tych tweetów, 10% prowadziło do stron zawierających fake newsy lub teorie spiskowe, a 15% do stron o skrajnie stronniczych poglądach. Analiza wykazała, że użytkownicy Twittera udostępniający linki do stron z fake newsami tworzyli bardziej skonsolidowane sieci z mniej zróżnicowaną strukturą połączeń w porównaniu do użytkowników rozpowszechniających wiadomości z rzetelnych źródeł o różnych orientacjach politycznych. Główne jednostki rozpowszechniające prawdziwe informacje to zazwyczaj dziennikarze i postacie publiczne ze zweryfikowanymi kontami na Twitterze, podczas gdy w przypadku fake newsów wiele czołowych kont to użytkownicy nieznani lub konta usunięte.

Wyniki te podkreślają, jak dezinformacja może manipulować opinią publiczną poprzez skoordynowane działania w mediach społecznościowych. Problem ten jest szczególnie istotny w kontekście szybkości rozprzestrzeniania się informacji w Internecie, gdzie brak kontroli nad źródłami powoduje szerzenie się nieprawdziwych lub skrajnie stronniczych treści. Natomiast brak nadzoru nad prawdziwością informacji prowadzi do polaryzacji społecznej, zwiększa dezinformację oraz osłabia zaufanie do instytucji publicznych. Może wpływać na wyniki wyborów, zagrażać zdrowiu publicznemu oraz powodować niepokój społeczny i zamieszki. Dodatkowo, fake newsy mogą destabilizować gospodarki, a także osłabiać relacje międzynarodowe, stanowiąc narzędzie w wojnie informacyjnej.

2.2 Współczesne rozwiązania

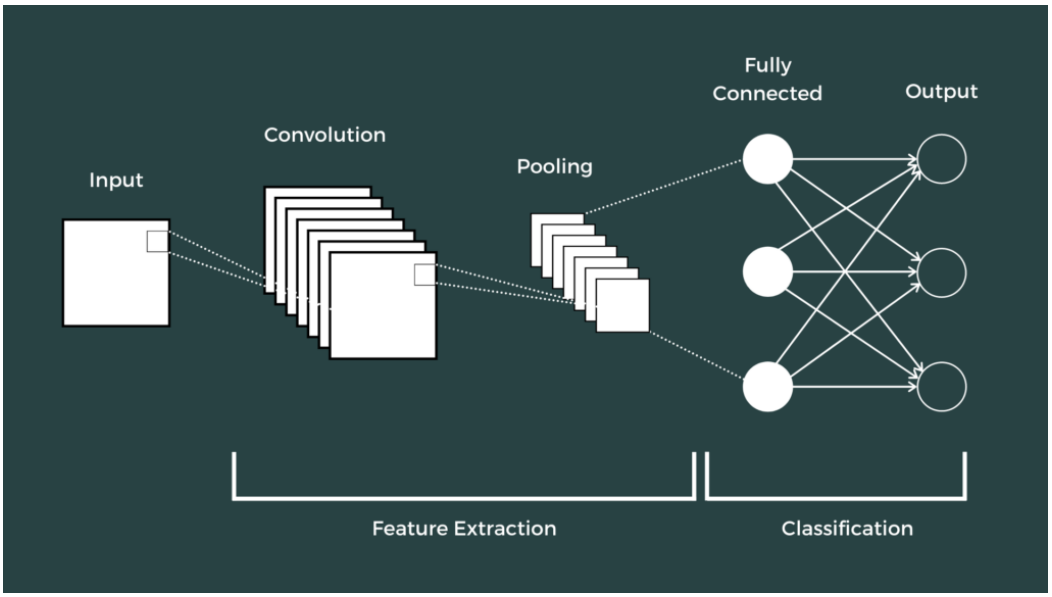
W przeszłości, jednym z podstawowych sposobów wykrywania fałszywych informacji była analiza treści przeprowadzana przez dziennikarzy i redaktorów. Profesjonaliści z branży medialnej polegali na swoim doświadczeniu oraz umiejętnościach krytycznego myślenia, aby identyfikować potencjalnie fałszywe lub wprowadzające w błąd informacje. Analiza ta obejmowała ocenę źródła, sprawdzanie faktów oraz konsultację z ekspertami w danej dziedzinie. Weryfikacja źródeł była kluczowym elementem tradycyjnego tzw. fact-checkingu, czyli identyfikacji i ocenie wiarygodności źródeł informacji [Fact-checking, 2024]. W przypadku podejrzanych wiadomości, dziennikarze często sięgali do pierwotnych źródeł lub szukali potwierdzenia informacji w innych, niezależnych mediach. Proces ten bywał czasochłonny, ale był niezbędny dla utrzymania rzetelności informacji. Natomiast w dobie nowoczesnych technologii, to sztuczna inteligencja oraz uczenie maszynowe stały się kluczowymi narzędziami w walce z dezinformacją.

Sztuczna inteligencja (AI, z ang. Artificial Intelligence) to teoria i rozwój systemów komputerowych zdolnych do wykonywania zadań, które w przeszłości wymagały ludzkiej inteligencji (rozpoznawanie mowy, podejmowanie decyzji czy identyfikowanie wzorców). Sztuczna inteligencja to termin ogólny obejmujący szeroką gamę technologii, w tym uczenie maszynowe oraz uczenie głębokie [Sztuczna inteligencja, 2024]. Algorytmy AI mogą być szkolone do rozpoznawania wzorców charakterystycznych dla fake newsów, takich jak specyficzne użycie języka, struktury tekstu czy źródła informacji. Dzięki analizie ogromnych zbiorów danych, sztuczna inteligencja jest w stanie szybko i efektywnie identyfikować potencjalnie fałszywe treści.

Rysunek 2.1 Zależność między sztuczną inteligencją, uczeniem maszynowym i głębokim
[źródło: opracowanie własne]

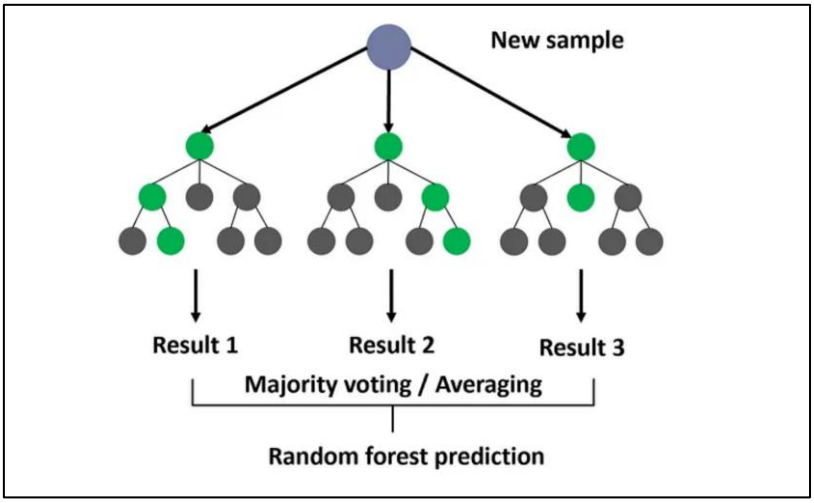
Ewaluacja wydajności najnowszych algorytmów w detekcji fałszywych informacji podkreśla skuteczność głębokiego uczenia w tej dziedzinie. Wykonano badanie, które polegało na systematycznym przeglądzie i meta-analizie metod wykrywania fałszywych wiadomości opartych na głębokim uczeniu, uczeniu maszynowym oraz metodach zespołowych [Antoun W., 2020]. Analiza obejmowała dane z 125 artykułów naukowych, uwzględniając wielkości efektów, heterogeniczność, analizy podgrup, meta-regresję oraz analizę stronniczości publikacji. Główne podejścia stosowane w literaturze to głębokie uczenie (z najczęściej używanym CNN – Convolutional Neural Network), zespołowe głębokie uczenie, zespołowe uczenie maszynowe, hybrydowe metody oraz uczenie maszynowe (z najczęściej stosowanym RF – Random Forest).

Convolutional Neural Network to rodzaj sieci neuronowej zaprojektowanej do przetwarzania danych o strukturze siatki, takich jak np. obrazy [CNN, 2024]. CNN jest szczególnie skuteczna w zadaniach związanych z rozpoznawaniem wzorców oraz wykrywaniem obiektów. Składa się z warstw konwolucyjnych, które stosują sploty do obrazów, wykrywając cechy takie jak krawędzie i tekstury. Warstwy tzw. poolingowe redukują rozmiar danych, co kontroluje przetrenowanie i obliczenia. Dane są przetwarzane przez warstwy całkowicie połączone, kończąc na warstwie wyjściowej, która generuje końcowe wyniki klasyfikacji.



Rysunek 2.2 Convolutional Neural Network (CNN)
[źródło: CNN 2024]

Natomiast Random Forest to metoda uczenia zespołowego, która opiera się na budowie wielu drzew decyzyjnych na losowych podzbiorach danych treningowych oraz cech [Random Forest, 2024]. Poprzez agregację wyników z tych drzew, model zapewnia odporność na przetrenowanie i stabilność predykcji. Dodatkowo, Random Forest ocenia ważność poszczególnych cech, co ułatwia interpretację modeli i selekcję istotnych cech w zadaniach klasyfikacji lub regresji. Jest to wszechstronny oraz skuteczny algorytm, który cieszy się dużą popularnością w analizie danych.



Rysunek 2.3 Random Forest (RF)
[źródło: Random Forest 2024]

Kolejne badanie potwierdziło efektywność głębokiego uczenia przy wykrywaniu fałszywych informacji [Jolly A., Kumar S., 2022]. Oceniono najnowocześniejsze algorytmy do wykrywania fałszywych wiadomości na zbiorze danych zawierającym 7796 wyciągów z różnych kontekstów, w tym komunikatów prasowych, przemówień kampanijnych i programów radiowych oraz telewizyjnych. Dane podzielono na część treningową (80%) i testową (20%). Do walidacji wykorzystano trzy bazy danych: LIAR, zbiory danych fałszywych wiadomości oraz korpusowe. Modele oparte na głębokim uczeniu, takie jak CNN (11%), LSTM (11%) i Bi-LSTM (12%), oraz inne metody jak Naiwny Klasyfikator Bayesa – ang. Naive Bayes (11%), Adaboost (10%) i GAN (z ang. Generative Adversarial Network) (11%), osiągnęły wysoką dokładność w wykrywaniu fałszywych wiadomości w badanych bazach danych korpusowych.

Uwagę badaczy przyciąga model oparty na LTSM, który stanowi rodzaj rekurencyjnej sieci neuronowej. LTSM wykazuje się zdolnością do zachowania kontekstu i zależności długoterminowych, co jest niezwykle istotne w analizie tekstu. W kontekście detekcji fałszywych informacji, gdzie zrozumienie kontekstu oraz sekwencji jest kluczowe, LTSM może stanowić wartościowe narzędzie. Unikanie utraty informacji w dłuższych sekwencjach tekstu, dzięki mechanizmom pamięci długoterminowej, sprawia, że LTSM może efektywnie przeciwdziałać manipulacjom i subtelnościom występującym w fałszywych informacjach. Dlatego też, zastosowanie modelu na bazie LTSM w detekcji fałszywych informacji staje się obiecującą perspektywą w dziedzinie rozwijających się technik analizy tekstu [LSTM, 2024].

Wprowadzenie transformera w 2017 roku zrewolucjonizowało zadania przetwarzania języka naturalnego, umożliwiając lepsze uchwycenie kontekstu oraz informacji semantycznych w tekście [Attention Is All You Need, 2017]. Modele oparte na transformerach, takie jak BERT i jego warianty, znacznie poprawiły wyniki w zadaniach wykrywania fałszywych wiadomości. Modele te są obecnie trendem badawczym w tej dziedzinie, mimo że wciąż mają trudności z osiągnięciem wysokiej wydajności w praktycznych zastosowaniach [Yuan L., 2023]. Warto też wspomnieć, że techniki osadzania słów, takie jak Word2Vec [Word2vec, 2024], FastText [FastText, 2024] i GloVe [GloVe, 2024], również pomagają w zrozumieniu semantyki i

kontekstu, a ich użycie pomaga w zwiększeniu efektywności implementowanych modeli do wykrywaniu fałszywych wiadomości.

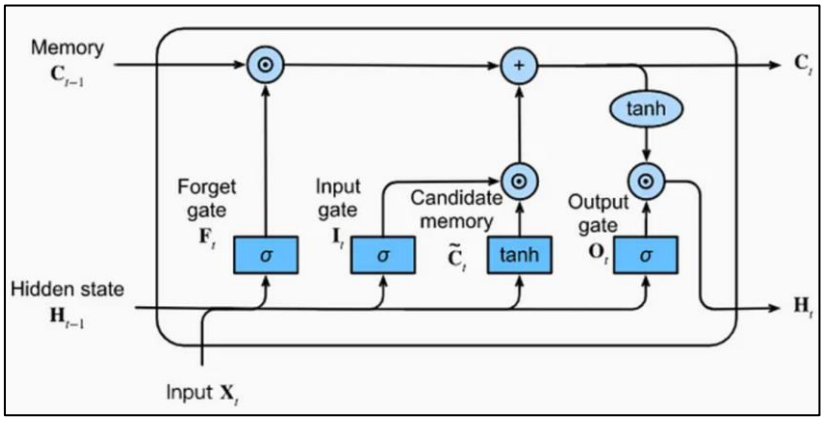
3. Analiza wybranych rozwiązań

W niniejszym rozdziale przedstawiono analizę wybranych rozwiązań do utworzenia modeli, służących do rozpoznawania fake newsów.

3.1 LSTM – Long Short-Term Memory

LSTM (Long Short-Term Memory) to zaawansowana forma rekurencyjnej sieci neuronowej (RNN, z ang. Recurrent Neural Network), zaprojektowana specjalnie do uczenia się długoterminowych zależności w sekwencjach danych tekstowych. W odróżnieniu od standardowych RNN, LSTM wyposażona jest w specjalne komórki pamięci, zwane komórkami LSTM, które umożliwiają przechowywanie informacji na dłuższy czas. Komórki te kontrolują przepływ informacji za pomocą trzech typów bramek: wejściowej, wyjściowej i bramki zapominania [LSTM, 2024].

LSTM jest efektywnym narzędziem do analizy tekstu dzięki swojej zdolności do przechowywania i wykorzystywania informacji z długich sekwencji, co jest kluczowe przy rozpoznawaniu kontekstu w zdaniach czy artykułach. Ponadto, LSTM dobrze radzi sobie z problemem zanikającego gradientu, co pozwala na skuteczniejsze uczenie się na długich sekwencjach tekstu bez utraty informacji. Niemniej jednocześnie wymaga znacznych zasobów obliczeniowych i jest czasochłonna w trenowaniu, co może być istotnym ograniczeniem w przypadku dużych zbiorów danych tekstowych.



Rysunek 3.1 Architektura LSTM
[źródło: Architecture of a LSTM Unit 2024]

3.2 Word2vec

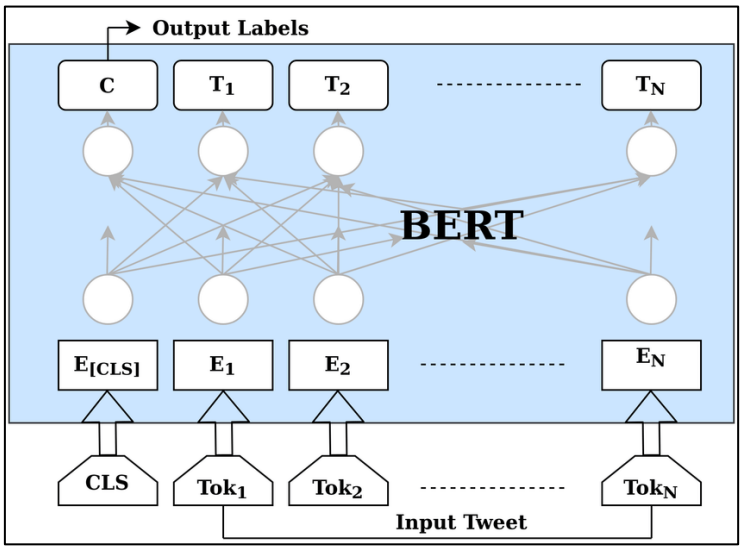
Word2vec to technika konwertująca słowa na wektory liczbowe, co pozwala reprezentować semantyczne znaczenie słów. Model ten trenuje się szybko na dużych korpusach tekstowych i efektywnie uchwytuje podobieństwa semantyczne i syntaktyczne między słowami. Jego zaletą jest również wszechstronność jako wejściowy wektor dla różnych modeli NLP [Word2vec, 2024]. Jednakże, Word2vec jest statyczny (każde słowo ma tylko jeden wektor), co może być niewystarczające w kontekstach z wieloznacznością słów, a także nie uwzględnia kontekstu globalnego w zdaniach.

Ta technika jest wartościowa w połączeniu z LSTM, ponieważ dostarcza sieci neuronowej wektory słów, które są niezbędne do modelowania długoterminowych zależności

w tekstach. Dzięki temu LSTM może efektywnie analizować długie sekwencje tekstu. Jest to kluczowe w detekcji fake newsów, gdzie istotne są subtelne zależności semantyczne i kontekstualne między słowami.

3.3 BERT – Bidirectional Encoder Representations from Transformers

BERT (z ang. Bidirectional Encoder Representations from Transformers) to zaawansowany model językowy, który odniósł ogromny sukces w dziedzinie przetwarzania języka naturalnego dzięki swojej zdolności do efektywnego uczenia się na ogromnych zbiorach tekstowych [Google BERT, 2024]. Model ten składa się z wielu warstw transformerowych, które umożliwiają mu dokładną analizę kontekstu i zależności między słowami w zdaniach. Kluczowym elementem budowy modelu BERT jest warstwa encoder, zawierająca sekwencje warstw transformerowych, które wykonują m.in. self-attention, tłumaczone jako samozwracanie uwagi. Mechanizm self-attention pozwala BERT-owi na równoczesne przetwarzanie wszystkich słów w zdaniu, co umożliwia lepsze zrozumienie kontekstu tekstu. Transformer zyskał popularność dzięki swojej zdolności do równoczesnego przetwarzania długich sekwencji tekstu i generowania wysokiej jakości reprezentacji językowych, co czyni go bardzo skutecznym w zadaniach związanych z przetwarzaniem języka naturalnego. Jego zdolność do uwzględniania szerokiego kontekstu oraz subtelnych różnic w użyciu języka pozwala na wykrywanie manipulacji informacyjnych, a także identyfikację dezinformacji.



Rysunek 3.2 Architektura BERT
[źródło: BERT model architecture 2024]

3.4 Podsumowanie

LSTM i BERT stanowią wybrane przez autorkę pracy rozwiązania do zaimplementowania, aby umożliwić detekcję fake newsów. Oferują one różne podejścia do przetwarzania
i rozumienia tekstu. LSTM w połączeniu z Word2vec jest skutecznym rozwiązaniem, zwłaszcza w kontekście analizy długich sekwencji tekstowych oraz rozpoznawaniu długoterminowych zależności między słowami. Word2vec dostarcza LSTM wektorów słów,

które pomagają w efektywnym uczeniu się reprezentacji semantycznych słów w kontekście długich tekstów, co jest kluczowe przy analizie treści artykułów czy postów.

Z kolei BERT oferuje zaawansowane możliwości przetwarzania języka naturalnego dzięki swojej architekturze transformera. BERT może być używany do detekcji fake newsów, ponieważ posiada zdolność do dwukierunkowego przetwarzania i generowania kontekstualnych reprezentacji słów. Dzięki temu BERT jest w stanie uchwycić subtelne niuanse w kontekście tekstu, co może pomóc w identyfikacji manipulacyjnych technik językowych, sprzeczności semantycznych oraz innych cech charakterystycznych dla fałszywych informacji.

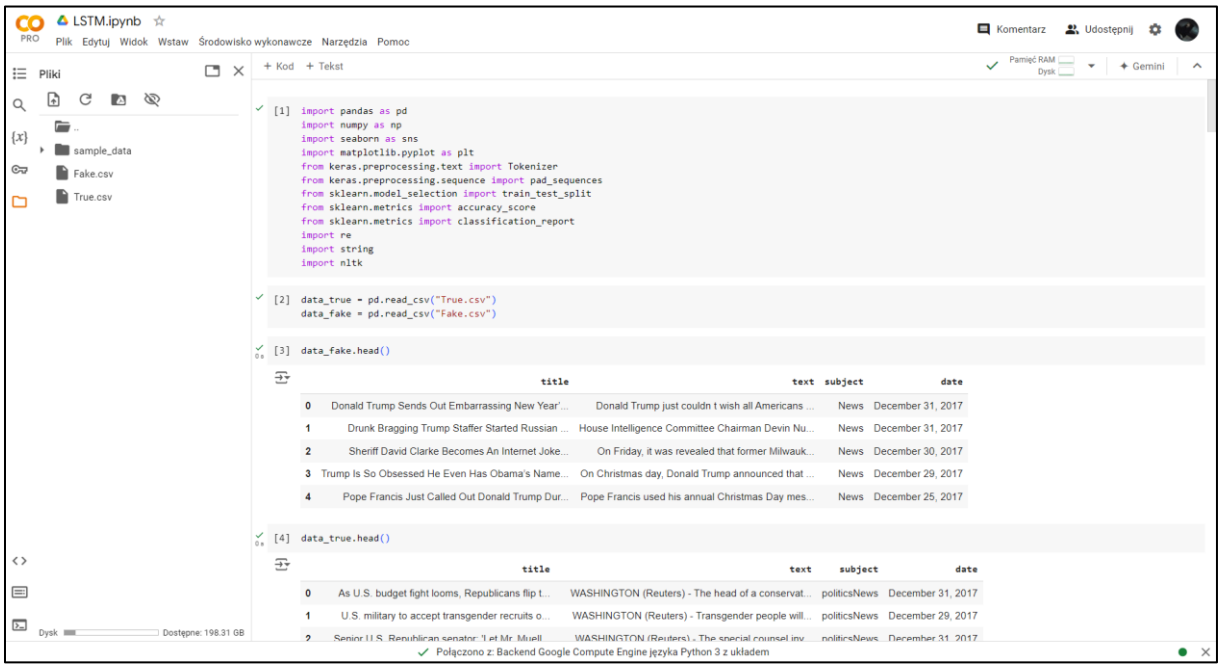
Wybór pomiędzy LSTM z Word2vec a BERT zależy więc od specyfiki problemu: jeśli kluczowe jest modelowanie długich sekwencji tekstowych i długoterminowych zależności, LSTM z Word2vec może być odpowiednim wyborem. Natomiast jeśli istotne jest głębsze zrozumienie kontekstu oraz bardziej zaawansowana analiza semantyczna oraz syntaktyczna tekstu, wówczas BERT może zapewnić lepsze rezultaty. W każdym przypadku, dostępność zasobów obliczeniowych jest istotnym czynnikiem decyzyjnym, ponieważ BERT wymaga znacznie większych zasobów obliczeniowych niż LSTM z Word2vec, szczególnie przy trenowaniu na dużych zbiorach danych.

4. Wykorzystane narzędzia i technologie

W niniejszym rozdziale omówione zostały wszystkie narzędzia i technologie, które autorka użyła w ramach pracy dyplomowej.

4.1 Google Colab

Google Colab to usługa chmurowa oferowana przez Google, umożliwiająca pisanie i uruchamianie kodu w języku Python bezpośrednio w przeglądarce. Użytkownicy mają dostęp do darmowego GPU (z ang. Graphics processing unit) i TPU (z ang. Tensor Processing Unit), co sprawia, że Google Colab jest optymalnym narzędziem do trenowania modeli uczenia maszynowego. Autorka pracy zdecydowała się również na użycie Colab Pro, czyli płatnej wersji, która oferuje lepsze zasoby obliczeniowe (szybsze oraz bardziej stabilne GPU oraz TPU), dłuższe sesje pracy, wyższe limity pamięci i inne zaawansowane funkcje, co znacznie przyśpiesza proces trenowania i eksperymentowania z modelami [Google Colab, 2024].



Rysunek 4.1 Interfejs Google Colab
[źródło: Google Colab 2024]

4.2 Python

Autorka zdecydowała się wykorzystać język Python do implementacji modeli. Python to wszechstronny język programowania wysokiego poziomu, szeroko stosowany w nauce o danych, uczeniu maszynowym i przetwarzaniu języka naturalnego. Jego prostota oraz szeroki wybór bibliotek sprawia, iż jest on popularnym wyborem w dziedzinach analizy danych i uczenia maszynowego, umożliwiając łatwe manipulowanie danymi, tworzenie modeli czy też przeprowadzanie analiz [Python, 2024].

4.3 Pandas

Pandas to biblioteka Pythona, która oferuje struktury danych i narzędzia do manipulacji oraz analizy danych tabelarycznych, takich jak DataFrame. Jest szczególnie przydatna do wstępnego przetwarzania i analizy zbiorów danych, umożliwiając filtrowanie, agregowanie i manipulację danymi tekstowymi, co jest kluczowe w projekcie detekcji fake newsów [Pandas, 2024].

4.4 Numpy

Numpy to biblioteka Pythona dostarczająca wsparcie dla wielowymiarowych tablic oraz różnorodnych funkcji matematycznych. Numpy jest podstawą dla wielu operacji numerycznych i obliczeń, niezbędnych w procesie przetwarzania dużych zbiorów danych oraz trenowania modeli uczenia maszynowego [Numpy, 2024].

4.5 TensorFlow/Keras

TensorFlow to rozwijana przez Google biblioteka do uczenia maszynowego [TensorFlow, 2024], a Keras to jej wysokopoziomowe API [Keras, 2024]. Keras ułatwia budowanie i trenowanie złożonych modeli sieci neuronowych. W projekcie detekcji fake newsów, TensorFlow oraz Keras mogą być używane do tworzenia zaawansowanych modeli NLP (ang. Natural Language Processing), które analizują i klasyfikują teksty jako prawdziwe lub fałszywe.

4.6 TensorFlow Hub

TensorFlow Hub to repozytorium pretrenowanych modeli TensorFlow. Umożliwia łatwe korzystanie z zaawansowanych modeli do różnych zadań, w tym przetwarzania języka naturalnego. W kontekście detekcji fake newsów, TensorFlow Hub pozwala na szybkie zaimportowanie i zastosowanie gotowych modeli NLP, co znacząco przyspiesza rozwój projektu i poprawia jego efektywność [TensorFlow Hub, 2024].

4.7 Scikit-learn

Scikit-learn to biblioteka Pythona oferująca szeroki wachlarz algorytmów do uczenia maszynowego, takich jak klasyfikacja, regresja, klasteryzacja i redukcja wymiarów. W detekcji fake newsów, Scikit-learn jest używana do budowania i ewaluacji modeli klasyfikacyjnych oraz do wstępnego przetwarzania danych [Scikit-learn, 2024].

4.8 Preprocess_kgptalkie

Preprocess_kgptalkie to biblioteka Pythona dedykowana wstępnemu przetwarzaniu danych tekstowych, obejmująca różnorodne techniki, takie jak tokenizacja, stemming, lematyzacja i usuwanie stop-słów. Użycie tej biblioteki pozwala na skuteczne przygotowanie tekstów do dalszej analizy i modelowania [Preprocess_kgptalkie, 2024].

4.9 NLTK (Natural Language Toolkit)

NLTK to kompleksowa biblioteka do przetwarzania języka naturalnego, oferująca narzędzia i zasoby do tokenizacji, stemmingu, lematyzacji, analizy składniowej oraz semantycznej. NLTK może być używana do wstępnego przetwarzania tekstów oraz do bardziej zaawansowanych analiz lingwistycznych, które mogą wspomóc klasyfikację tekstu [NLTK, 2024].

4.10 Gensim

Gensim to zaawansowana biblioteka Python służąca do modelowania tematów i przetwarzania dużych korpusów tekstowych. Gensim jest szczególnie znana ze swojej efektywności w budowaniu modeli tematycznych, takich jak Latent Dirichlet Allocation (LDA), oraz w tworzeniu wektorowych reprezentacji słów przy użyciu algorytmu Word2Vec. Biblioteka jest niezwykle przydatna w zaawansowanych technikach przetwarzania języka naturalnego [Gensim, 2024].

4.11 Matplotlib

Matplotlib to biblioteka do tworzenia wizualizacji danych w Pythonie, umożliwiająca generowanie różnorodnych wykresów i diagramów. Autorka użyła Matplotlib do prezentacji danych w sposób przejrzysty i zrozumiały [Matplotlib, 2024].

4.12 Seaborn

Seaborn to biblioteka oparta na Matplotlib, która upraszcza tworzenie estetycznych i informacyjnych wizualizacji danych. Oferuje wyższy poziom abstrakcji oraz łatwiejsze generowanie skomplikowanych wykresów. W kontekście pracy dyplomowej, Seaborn został użyty do wizualizacji wyników, co pomaga w ich analizie i interpretacji [Seaborn, 2024].

5. Zbiór danych

Niniejszy rozdział skupia się na analizie cech rzetelnego zbioru danych. Omówione zostały również korzyści, które wynikają z wykorzystania platformy Kaggle oraz formatu CSV.

Dobry zbiór danych jest podstawą każdego skutecznego modelu AI, w tym oczywiście modeli do wykrywania fake newsów. Wybrany zestaw danych powinien być przede wszystkim kompletny i reprezentatywny dla problemu, jaki ma rozwiązywać. **Kompletny** zbiór danych to taki, który zawiera wystarczająco dużo przykładów różnych klas. Natomiast **reprezentatywność** oznacza, że dane muszą odzwierciedlać rzeczywisty rozkład przypadków w świecie rzeczywistym, co zapewnia, że model będzie działał poprawnie na nowych, niewidzianych wcześniej danych.

Jedną z najbardziej popularnych platform do selekcjonowania oraz udostępniania zestawów danych jest Kaggle [Kaggle, 2024]. Zbiory dostępne na Kaggle są dobrze udokumentowane i przetestowane przez społeczność. Ich dokumentacja jest zazwyczaj bardzo szczegółowa – zawiera informacje na temat źródeł danych, metod ich zbierania, a także potencjalnych ograniczeń. Dzięki temu użytkownicy mają możliwość pełnego zrozumienia kontekstu danych. Ponadto, zestawy są oceniane i komentowane przez innych członków społeczności, co umożliwia weryfikację danych oraz wprowadza pewny poziom ich kontroli – popularne zbiory mają setki, a nawet tysiące komentarzy, które pomagają w identyfikacji błędów lub problemów, co z kolei zwiększa ich wiarygodność.

W związku ze wszystkimi zaletami Kaggle, które zostały opisane powyżej, autorka pracy zdecydowała się na wybór zbioru właśnie z tej platformy. Wybrano rozwiązanie członka społeczności i specjalisty ds. danych Sameera Patela. Zaproponował on dwa zestawy danych – jeden zawierający tylko prawdziwe informacje, a drugi jedynie fałszywe. Ta propozycja cieszy się dużym uznaniem użytkowników, co potwierdzała wysoka liczba ocen zbioru danych. Dodatkowo, zbiór ten był najbardziej aktualny w porównaniu do innych wysoko ocenianych zestawów, co zapewniło jego większą adekwatność oraz aktualność [Patel S., 2024].

Pierwszy zestaw danych skupia się na fałszywych informacjach. Dane te reprezentują próbki informacyjne, które są nieprawdziwe i mają potencjał wprowadzenia w błąd czytelnika. Zestaw ten zawiera 23 481 rekordów w formacie CSV, a jego polami są: title (tytuł), text (treść), subject (temat) oraz date (data). Format CSV (z ang. Comma-Separated Values) jest jednym z najpopularniejszych formatów do przechowywania danych w formie tabelarycznej. Jego główną zaletą jest prostota i uniwersalność – pliki CSV mogą być otwierane i edytowane. CSV jest także efektywny pod względem przechowywania danych, ponieważ nie zawiera dodatkowych informacji o formatowaniu, które mogą zajmować miejsce, dzięki czemu jest idealny do przechowywania dużych zbiorów danych [CSV, 2024].

	title	text	subject	date
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017

Rysunek 5.1 Widok zbioru danych z fake newsami z Kaggle
[źródło: Patel S. 2024]

Drugi zbiór danych składa się z informacji prawdziwych, zweryfikowanych poprzez renomowane źródła, takie jak Reuters – międzynarodowej agencji informacyjnej, która specjalizuje się w zbieraniu, redagowaniu i dystrybucji wiadomości oraz informacji z całego świata. Dane z tego zestawu stanowią punkt odniesienia dla analizy wiarygodności informacji. Zestaw ten zawiera 21 417 rekordów również w formacie CSV, z identyczną strukturą pól: title, text, subject oraz date.

	title	text	subject	date
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017

Rysunek 5.2 Widok zbioru danych z prawdziwymi informacjami z Kaggle
[źródło: Patel S. 2024]

Oba zestawy danych są kluczowymi elementami badawczymi i aby umożliwić efektywne nauczanie modeli sztucznej inteligencji, konieczne było dostosowanie ich w jeden spójny zbiór. Oba zestawy zostały scalone oraz dostosowane pod trening modelu. W tym celu skorzystano z procesu tzw. czyszczenia danych (ang. data cleaning), który obejmuje identyfikację, korektę lub usuwanie błędów oraz nieścisłości w danych. Proces ten jest kluczowy dla utrzymania jakości i spójności informacji, co ma istotne znaczenie dla skutecznego szkolenia modeli rozpoznawania fake newsów [Data Cleaning, 2024].

Czyszczenie danych obejmowało szereg operacji mających na celu usunięcie potencjalnych zakłóceń oraz ustandaryzowanie struktury i formatu informacji. Oto główne kroki podjęte w trakcie tego procesu:

1. Sprawdzenie i usunięcie duplikatów – przeprowadzono analizę zbioru w celu identyfikacji oraz eliminacji wszelkich duplikatów, aby uniknąć nadmiernego wpływu powtarzających się informacji na proces uczenia modelu.
2. Usunięcie informacji o agencji prasowej i o nazwie użytkownika.
3. Sprawdzenie i usunięcie rekordów bez treści – skontrolowano kolumnę "text", aby zachować tylko kompleksowe informacje.
4. Standaryzacja wielkości liter – wszystkie teksty zostały przekształcone do jednolitej formy poprzez standaryzację wielkości liter, co pozwoliło na zwiększenie spójności i analizy.
5. Dodanie kolumny "label" – na podstawie rodzaju danych (fałszywe czy prawdziwe), utworzono nową kolumnę "label", w której wartości odpowiadające fake newsom zostały oznaczone jako 1, a wartości prawdziwych informacji jako 0.
6. Usunięcie znaków specjalnych – w celu redukcji szumu i ułatwienia analizy semantycznej, usunięto znaki specjalne, takie jak: @, ., #, które mogły wpływać na interpretację tekstu.

Po zakończeniu procesu czyszczenia danych i scaleniu zbiorów, powstał jednolity oraz gotowy do wykorzystania zestaw danych. Kluczową kwestią dostosowywania zbioru było zapewnienie, że dane fałszywe i prawdziwe są reprezentatywne oraz podlegają równym warunkom analizy.

	text	label
22953	watch the local news report as they explain wh...	0
37970	chinese foreign minister wang yi will visit my...	1
682	if there s anyone in this country who s an exp...	0
26988	call them china s bubble generation they were ...	1
22705	us president donald trump arrived in the phili...	1

Rysunek 5.3 Widok przygotowanego zbioru danych do trenowania modeli detekcji fake newsów
[źródło: opracowanie własne]

6. Implementacja modeli

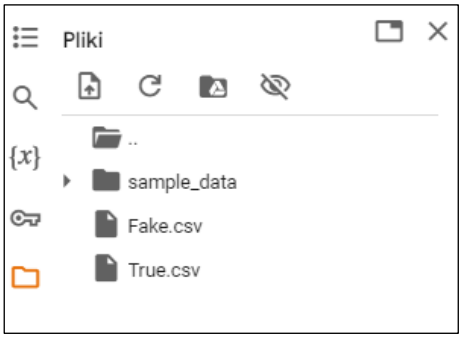
W niniejszym rozdziale ukazano implementację modeli, czyli przedstawiono kod oraz omówiono jego działanie. Rozdział podzielono na trzy podrozdziały. W pierwszym omawiane jest import i przygotowanie danych, na których oba modele pracują. W drugim znajduje się implementacja modelu LSTM, a w trzecim implementacja modelu BERT.

6.1 Zbiór danych

Implementacja modelu rozpoczęła się od zaimportowania dwóch zestawów danych w formacie CSV. Oba pliki przesłano do pamięci sesji w środowisku wykonawczym Google Colab (rysunek 6.1), aby następnie móc je wczytać do odpowiednich zmiennych (`data_true` i `data_false`).

```
import pandas as pd
import numpy as np
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
import string
import nltk
import seaborn as sns
import matplotlib.pyplot as plt
```

Listing 6.1 Zaimportowanie potrzebnych bibliotek



Rysunek 6.1 Widok na przesłane zbiory danych
[źródło: Google Colab 2024]

```
data_true = pd.read_csv("True.csv")
data_fake = pd.read_csv("Fake.csv")
```

Listing 6.2 Wczytanie zbiorów do zmiennych

Następnie dodawana jest kolumna `label`, która jest etykietą symbolizującą prawdziwość poszczególnych pozycji. Dane z zestawu `data_true` zostają oznaczone jako 1, a dane z `data_fake` przypisane zostało 0.

```
data_fake["label"] = 0
data_true["label"] = 1
```

Listing 6.3 Dodanie etykiet

W kolejnej części kodu rozpoczyna się proces czyszczenia danych, który został dokładnie omówiony w rozdziale 5. Ukazany fragment skupia się na usunięciu informacji o agencji prasowej z zestawu danych z prawdziwymi informacjami za pomocą funkcji `remove_reuters_city`, aby kolumna `text` zawierała jedynie treść informacji (już bez źródła ich pobrania, ponieważ nie jest ono na tym poziomie istotne, a może jedynie przeszkadzać w poprawnym wyszkoleniu modelu).

```
def remove_reuters_city(text):
    parts = text.split('-', 1)
    return parts[1].strip() if len(parts) > 1 else text.strip()

data_true['text'] = data_true['text'].apply(remove_reuters_city)

data_true.head()
```

Listing 6.4 Funkcja usuwająca informacje o agencji prasowej

text	text
WASHINGTON (Reuters) - The head of a conservat...	The head of a conservative Republican faction ...
WASHINGTON (Reuters) - Transgender people will...	Transgender people will be allowed for the fir...
WASHINGTON (Reuters) - The special counsel inv...	The special counsel investigation of links bet...
WASHINGTON (Reuters) - Trump campaign adviser ...	Trump campaign adviser George Papadopoulos tol...
SEATTLE/WASHINGTON (Reuters) - President Donal...	President Donald Trump called on the U.S. Post...

Rysunek 6.2 Porównanie wyczyszczonych danych
[źródło: opracowanie własne]

Analogicznie wykonujemy podobną operację na kolumnie `text` w zbiorze z fałszywymi informacjami `data_fake`, aby zawierała jedynie treść fake newsów (bez nazwy użytkownika).

```
def remove_twitter_handles(text):
    handle_pattern = r'@\w+'
    cleaned_text = re.sub(handle_pattern, '', text)
    cleaned_text = re.sub(r'\(\s*\)', '', cleaned_text)
    cleaned_text = re.sub(r'\s+', ' ', cleaned_text).strip()
    return cleaned_text

data_fake['text'] = data_fake['text'].apply(remove_twitter_handles)

data_fake.head()
```

Listing 6.5 Funkcja usuwająca informacje o nazwie użytkownika

W kolejnym kroku następuje łączenie danych prawdziwych oraz fałszywych w jeden zestaw za pomocą funkcji `concat`. Następnie usunięte zostają kolumny: `title`, `subject`, `date`, ponieważ nie są istotne w kontekście trenowania modelu. Dodatkowo dane zostają losowo przemieszane za pomocą `data.sample(frac = 1)`, aby uzyskać równomierne rozproszenie danych, co zapobiega kolejnościowym efektom ubocznym i skutkuje zwiększeniem szansy na naukę bardziej uniwersalnych wzorców przez model.

```
data_concat = pd.concat([data_fake, data_true], axis = 0 )
data = data_concat.drop(["title", "subject", "date"], axis = 1)
data.isnull().sum()
data = data.sample(frac = 1)
```

Listing 6.6 Łączenie, usuwanie i tasowanie zbioru danych

Na listingu 6.7 ukazane zostało resetowanie indeksu danych, a następnie usunięcie starego indeksu, aby ostatecznie uporządkować dane po wszystkich wcześniejszych operacjach. W ten sposób powstał zbiór o strukturze widocznej na rysunku 6.3.

```
data.reset_index(inplace = True)
data.drop(["index"], axis = 1, inplace = True)
data.head()
```

Listing 6.7 Resetowanie indeksu i uporządkowanie danych

	text	label
0	Tune in to the Alternate Current Radio Network...	0
1	A bill to make California a sanctuary state by...	1
2	NonEU Norway called on Brussels and London on ...	1
3	Tim Poole is citizen journalist who s done som...	0
4	What a great photo of President Trump joined b...	0

Rysunek 6.3 Zbiór po uporządkowaniu danych
[źródło: opracowanie własne]

Następny etap to przetwarzanie tekstu. Za pomocą zaimportowanego pakietu `preprocess_kgptalkie` jako `ps` na kolumnie `text` została zastosowana funkcja `remove_special_chars`, która usuwa znaki specjalne.

```
import preprocess_kgptalkie as ps
data['text'] = data['text'].apply(lambda x: ps.remove_special_chars(x))
```

Listing 6.8 Usuwanie znaków specjalnych

6.1 Model LSTM

W tym podrozdziale zostanie przedstawiona implementacja modelu LSTM. Fragment kodu z listingu 6.9 jest kontynuacją kodu z listingu 6.8. Następuje tu parę ważnych operacji. Po pierwsze – usunięcie tzw. stop-słów (z ang. stopwords). Stopwords to często używane słowa (np. the, is, and), które nie niosą dużo informacji w analizie tekstu. Usunięcie ich pozwala na skupienie się na istotnych słowach. Po drugie – tokenizacja, która polega na podziale tekstu na mniejsze jednostki, takie jak słowa lub zdania. Po trzecie – normalizacja, czyli ustandaryzowana została wielkość liter (każde zdanie przetworzono na małe litery).

```
X = []
y= data['label'].values

nltk.download('stopwords')
nltk.download('punkt')

stop_words = set(nltk.corpus.stopwords.words("english"))
tokenizer = nltk.tokenize.RegexpTokenizer(r'\w+')
for par in data["text"].values:
    tmp = []
    sentences = nltk.sent_tokenize(par)
    for sent in sentences:
        sent = sent.lower()
        tokens = tokenizer.tokenize(sent)
        filtered_words = [w.strip() for w in tokens if w not in stop_words and
len(w) > 1]
        tmp.extend(filtered_words)
    X.append(tmp)
```

Listing 6.9 Usuwanie Stopwords, tokenizacja i normalizacja

Listing 6.10 pokazuje użycie techniki Word2Vec do wektorowej reprezentacji słów, która mapuje słowa na wielowymiarowe wektory numeryczne. Każde słowo zostaje zaprezentowane jako punkt w przestrzeni wektorowej, gdzie podobne słowa są umiejscowione bliżej siebie. To umożliwia modelom lepsze rozumienie kontekstu zdań. Parametr `sentences=X` przekazuje przetworzone teksty jako zbiór zdań do modelu Word2Vec, `vector_size=100` określa wymiarowość wektorów wyjściowych (każde słowo będzie miało reprezentację jako wektor o 100 wymiarach), `window=10` określa maksymalną odległość między słowami, jaka jest brana pod uwagę podczas uczenia modelu, `min_count=1` określa minimalną liczbę wystąpień słowa w korpusie, aby zostało ono uwzględnione w modelu.

```
import gensim
w2v_model = gensim.models.Word2Vec(sentences=X, vector_size=100, window=10,
min_count=1)
```

Listing 6.10 Użycie techniki Word2vec

Tokenizacja tekstów do sekwencji liczbowych tokenów jest niezbędna dla późniejszego wykorzystania ich jako danych wejściowych do modelu. Najpierw uczymy tokenizator na naszych tekstach, co pozwala mu stworzyć indeks słów. Następnie konwertujemy teksty na sekwencje tokenów.

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X)
X = tokenizer.texts_to_sequences(X)
```

Listing 6.11 Tokenizacja

Następnie stworzona zostaje macierz wag dla warstwy embedding. Tworzymy macierz wag dla warstwy embedding, aby móc zamienić numeryczne indeksy słów na ich wektoryczne reprezentacje. Macierz ta wykorzystuje wytrenowane wektory Word2Vec, które zawierają semantyczne informacje o słowach, co powoduje, że model może lepiej rozumieć kontekst słów od samego początku trenowania. Dodatkowo przypisujemy wektor zerowy, jeśli słowo nie ma wektora w modelu Word2Vec, aby zapewnić jednolitą długość wektorów dla wszystkich słów.

To zapobiega błędom w modelu, który wymaga, aby każde słowo miało reprezentację o jednakowej długości.

```
X = pad_sequences(X, maxlen=1000)

vocab_size = len(tokenizer.word_index) + 1
vocab = tokenizer.word_index

def get_weight_matrix(model, tokenizer):
    vocab_size = len(tokenizer.word_index) + 1
    DIM = model.vector_size

    weight_matrix = np.zeros((vocab_size, DIM))
    zero_vector_count = 0
    total_word_count = 0

    for word, i in tokenizer.word_index.items():
        total_word_count += 1
        try:
            weight_matrix[i] = model.wv[word]
        except KeyError:
            weight_matrix[i] = np.zeros(DIM)
            zero_vector_count += 1

    return weight_matrix, total_word_count, zero_vector_count

embedding_vectors, total_words, zero_vector_words = get_weight_matrix(w2v_model,
tokenizer)
```

Listing 6.12 Macierz wag dla warstwy embedding

Listing 6.13 ukazuje budowę modelu sekwencyjnego LSTM, który jest wykorzystywany do klasyfikacji binarnej. Model składa się z warstwy embedding, która wykorzystuje wcześniej wygenerowane wektory. Następnie jest warstwa LSTM, jaka posiada liczbę jednostek i której złożoność przetwarzania sekwencji wynosi 128 jednostek, co zapewnia wystarczającą zdolność do uczenia się złożonych wzorców w danych sekwencyjnych, bez nadmiernego zwiększania poziomu zaawansowania modelu. Model kończy się warstwą Dense z jednym neuronem, który używa funkcji aktywacji `sigmoid` do predykcji prawdopodobieństwa klasyfikacji binarnej. Model jest kompilowany z optymalizatorem `adam`, który dostosowuje tempo uczenia się w trakcie trenowania, co przyspiesza proces i poprawia dokładność. Posiada również funkcję straty odpowiednią do problemów binarnych – `binary_crossentropy` jest efektywnym wyborem, ponieważ mierzy odległość między rozkładem prawdopodobieństwa prawdziwych etykiet a przewidywaniami modelu. Dodatkowo `metrics=['acc']` ocenia dokładność klasyfikacji modelu, czyli stosunek poprawnie przewidzianych etykiet do wszystkich próbek.

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Embedding, LSTM, Conv1D, MaxPool1D
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from tensorflow.keras.layers import Reshape

model = Sequential()
model.add(Embedding(vocab_size, output_dim=100, weights = [embedding_vectors],
input_length=maxlen, trainable=False))
model.add(Reshape((maxlen, 100), input_shape=(maxlen,)))
model.add(LSTM(units=128))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])

model.summary()
```

Listing 6.13 Budowa modelu sekwencyjnego LSTM

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 1000, 100)         22801500

 reshape (Reshape)           (None, 1000, 100)         0

 lstm (LSTM)                 (None, 128)               117248

 dense (Dense)               (None, 1)                 129

=================================================================
Total params: 22918877 (87.43 MB)
Trainable params: 117377 (458.50 KB)
Non-trainable params: 22801500 (86.98 MB)
```

Rysunek 6.4 Podsumowanie architektury modelu

Ostatnim krokiem było podzielenie danych na zestawy treningowe i testowe: 20% danych zostało użytych jako zbiór testowy, a pozostałe 80% jako zbiór treningowy. Potem nastąpiło wytrenowanie modelu na danych treningowych.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model.fit(X_train, y_train, validation_split=0.2, epochs=6)
```

Listing 6.14 Podział zbioru na zestaw treningowy i testowy oraz trenowanie modelu na bazie LSTM

6.2 Model BERT

Listing 6.15 przedstawia kontynuację kodu z listingu 6.8 w scenariuszu budowy modelu BERT. W przeciwieństwie do operacji w podrozdziale 6.1, nie jest tu potrzebna metoda z manualnym czyszczeniem tekstu. Warstwa `bert_preprocess` z TensorFlow Hub automatycznie zajmuje się przetwarzaniem tekstu, które jest wymagane przez model BERT. Obejmuje to tokenizację, czyli między innymi podzielenie tekstu na tokeny zgodne

z wymaganiami modelu czy też dodanie specjalnych tokenów (oznaczanych w literaturze jako [CLS] i [SEP]). Token [CLS] (z ang. classification token) jest dodawany na początku każdej sekwencji tekstowej wprowadzanej do modelu BERT. Pełni on rolę specjalnego znacznika używanego do reprezentacji całej sekwencji. Token [SEP] (z ang. separator token) jest używany jako znacznik końca sekwencji oraz jako separator między dwoma zdaniami w zadaniach, które wymagają porównania dwóch fragmentów tekstu, takich jak pytanie-odpowiedź. Dodatkowo warstwa automatycznie standaryzuje wielkość liter do małych. Warto wspomnieć, że BERT modeluje kontekst wszystkich słów, więc usuwanie Stopwords (tak jak zostało to opracowane w rozdziale 6.1) mogłoby zniszczyć struktury zdania i kontekst, który jest kluczowy dla tego modelu. Natomiast `encoder` to warstwa, która przyjmuje przetworzony tekst, a potem generuje jego reprezentację wektorową.

```
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_text as text

bert_preprocess =
hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")
encoder = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-
768_A-12/4")
```

Listing 6.15 Ładowanie warstwy przetwarzania wstępnego i warstwy encodera BERT

Następnie zbiór został podzielony na zestaw treningowy i testowy: 20% danych zostało użytych jako zbiór testowy, a pozostałe 80% jako zbiór treningowy.

```
max_len=100
data_text= data["text"]
data_label= data["label"]

X_train, X_test, y_train, y_test = train_test_split(data_text, data_label,
stratify = data_label, test_size = 0.2, random_state =42)
```

Listing 6.16 Podział zbioru na zestaw treningowy i testowy dla modelu BERT

Na listingu 6.17 definiowane jest wejście `text_input` – czyli warstwa wejściowa Keras, w której `shape=()` oznacza, że każde wejście jest pojedynczym ciągiem tekstowym. Następuje również przetwarzanie tekstu za pomocą warstwy `bert_preprocess` i kodowanie tekstu za pomocą `encoder`. Wynikiem enkodowania tekstu jest słownik zawierający różne wyjścia, w tym `pooled_output` (reprezentacja całego wejściowego tekstu oparta na tokenie [CLS]) oraz `sequence_output` (reprezentacje poszczególnych tokenów w tekście). Potem dodana została warstwa Dropout jako technika regularyzacji, która pomaga zapobiegać przetrenowaniu modelu. Warstwa zdefiniowana została w taki sposób, że z prawdopodobieństwem 10% (0.1) jednostki w tej warstwie będą losowo zerowane podczas treningu. Ostatnia linijka na listingu 6.17 ukazuje dołączenie warstwy Dense z funkcją aktywacji sigmoid – przekształca ona wyjście w wartość z przedziału [0, 1].

```
text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')
preprocessed_text = bert_preprocess(text_input)
outputs = encoder(preprocessed_text)

layer = tf.keras.layers.Dropout(0.1, name="dropout")(outputs['pooled_output'])
layer = tf.keras.layers.Dense(1, activation='sigmoid', name="output")(layer)
```

Listing 6.17 Budowa modelu do binarnej klasyfikacji tekstu z użyciem BERT

Listing 6.18 przedstawia tworzenie modelu Keras, który łączy wszystkie zdefiniowane wcześniej warstwy w jedną sieć neuronową. Wejście modelu to surowy tekst, który jest przetwarzany i kodowany przez BERT, a następnie przekazywany przez warstwę Dropout i warstwę Dense w celu uzyskania ostatecznej predykcji. Potem jest kompilacja modelu do treningu z odpowiednimi parametrami, czyli optymalizatorem `adam`, funkcją straty binarnej entropii krzyżowej `binary_crossentropy` oraz metryką dokładność klasyfikacji modelu.

```
model = tf.keras.Model(inputs=[text_input], outputs = [layer])

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Listing 6.18 Kompilacja modelu do binarnej klasyfikacji tekstu

Ostatnim krokiem jest wykorzystanie funkcji `fit`, która trenuje model na danych treningowych. Ważnymi parametrami tej funkcji, oprócz danych wejściowych i etykiet treningowych, są również epoki oraz rozmiar partii.

```
model.fit(X_train, y_train, epochs=1, batch_size = 32)
```

Listing 6.19 Trenowanie modelu BERT na danych treningowych

Liczba epok to liczba przejść modelu przez cały zbiór treningowy, natomiast rozmiar partii to liczba próbek przetwarzanych przez model przed aktualizacją wag. Rozmiar partii jest równy 32, ponieważ stanowi kompromis między małymi a dużymi rozmiarami partii, oferując równowagę między dokładnością aktualizacji wag a czasem treningu.

7. Działanie modeli

W niniejszym rozdziale przedstawiono metryki oceny oraz wyniki dwóch modeli stosowanych do detekcji fake newsów. W celu oceny obu modeli, przeprowadzono szereg eksperymentów, których wyniki zostały przedstawione za pomocą różnych metryk, takich jak dokładność (z ang. accuracy), raport klasyfikacji (z ang. classification report) oraz macierz pomyłek (z ang. confusion matrix) dla różnych wartości epok, aby ocenić efektywność i stabilność modeli w czasie treningu.

7.1 Metryki oceny

W ocenie modeli do klasyfikacji fake newsów w skryptach wykorzystano następujące metryki:

- Dokładność (Accuracy) – jest to stosunek liczby poprawnych przewidywań do całkowitej liczby przewidywań.
  Obliczane na podstawie wzoru:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

  gdzie:
  TP (True Positive): prawdziwie pozytywne przykłady;
  TN (True Negative): prawdziwie negatywne przykłady;
  FP (False Positive): fałszywie pozytywne przykłady;
  FN (False Negative): fałszywie negatywne przykłady.

- Raport klasyfikacji (Classification Report) – raport klasyfikacji podaje szczegółowe informacje na temat działania modelu, w tym precyzję, przypomnienie (z ang. recall), f1-score oraz wskaźniki dla każdej klasy.
  Komponenty:
  - Precision: Miara tego, jak wiele z przewidzianych pozytywnych przykładów jest rzeczywiście pozytywnych.
    Obliczane na podstawie wzoru::

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

  - Recall: Miara tego, jak wiele z rzeczywiście pozytywnych przykładów zostało poprawnie przewidzianych.
    Obliczane na podstawie wzoru::

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

- F1-Score: Średnia harmoniczna precyzji i przypomnienia, używana jako pojedyncza miara skuteczności modelu.
  Obliczane na podstawie wzoru::

$$F1 - Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (4)$$

- Macierz pomyłek (Confusion Matrix) – jest to narzędzie wizualizacyjne, które pokazuje, ile razy klasy rzeczywiste zostały poprawnie lub niepoprawnie sklasyfikowane przez model.
  Komponenty:
  - TP (True Positive): Liczba przypadków, które są rzeczywiście pozytywne i zostały poprawnie sklasyfikowane jako pozytywne.
  - TN (True Negative): Liczba przypadków, które są rzeczywiście negatywne i zostały poprawnie sklasyfikowane jako negatywne.
  - FP (False Positive): Liczba przypadków, które są rzeczywiście negatywne, ale zostały niepoprawnie sklasyfikowane jako pozytywne.
  - FN (False Negative): Liczba przypadków, które są rzeczywiście pozytywne, ale zostały niepoprawnie sklasyfikowane jako negatywne.

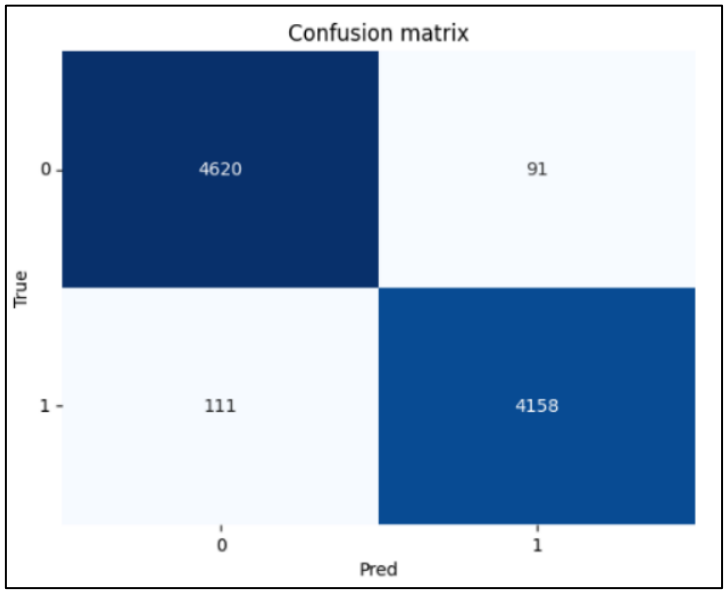
7.2 Klasyfikacja przy użyciu LSTM

W tym podrozdziale zostaną przedstawione wyniki działania modelu z LSTM. Przeprowadzono eksperymenty z różną liczbą epok, aby ocenić, jak wpływają one na dokładność modelu oraz znaleźć optymalną liczbę epok dla najlepszych wyników. Wybór liczby epok do eksperymentów był podyktowany chęcią monitorowania, jak model uczy się z upływem czasu. W efekcie zakończenie szkolenie zakończyło się na 7 epokach, ponieważ dalsze epoki mogłyby nie przynieść znaczącej poprawy wyników, a jedynie zwiększyć ryzyko przeuczenia.

Wynik dokładności i raport klasyfikacji dla pierwszego eksperymentu dla modelu z LSTM, w którym do szkolenia użyto 1 epoki przedstawiono na rysunku 7.1. Natomiast macierz pomyłek dla tego samego eksperymentu obrazuje rysunek 7.2.

```
accuracy_score(y_test, y_pred)

0.9775055679287306

print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       0.98      0.98      0.98      4711
           1       0.98      0.97      0.98      4269

    accuracy                           0.98      8980
   macro avg       0.98      0.98      0.98      8980
weighted avg       0.98      0.98      0.98      8980
```

Rysunek 7.1 Dokładność i raport klasyfikacji dla modelu z LSTM – 1 epoka



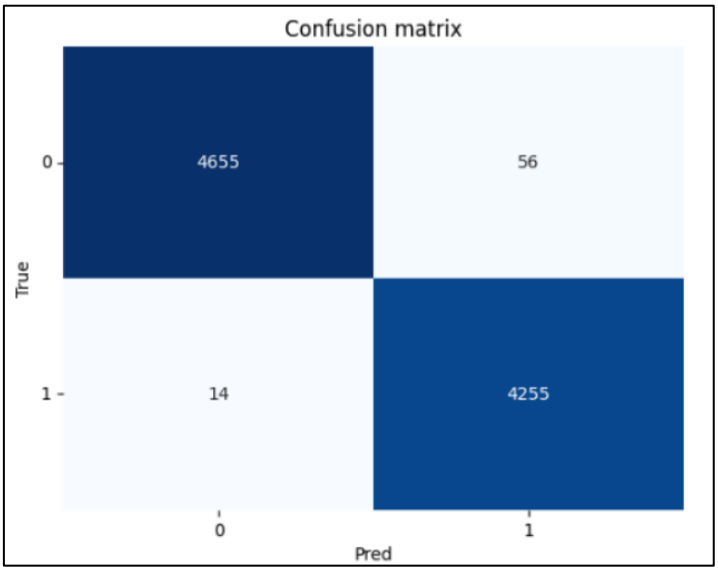
Rysunek 7.2 Macierz pomyłek dla modelu z LSTM – 1 epoka

Podsumowując, model z LSTM osiągnął bardzo wysoką dokładność po jednej epoce treningu. Wysokie wartości precyzji, czułości i F1-score dla obu klas wskazują na równomiernie dobrą wydajność modelu w klasyfikacji zarówno klasy 0, jak i klasy 1. Macierz pomyłek pokazuje, że liczba błędnych klasyfikacji jest niewielka w porównaniu do liczby przypadków poprawnie sklasyfikowanych.

Wynik dokładności i raport klasyfikacji dla drugiego eksperymentu dla modelu z LSTM, w którym do szkolenia użyto 2 epok przedstawiono na rysunku 7.3. Natomiast macierz pomyłek dla tego samego eksperymentu obrazuje rysunek 7.4.

```
accuracy_score(y_test, y_pred)

0.9922048997772829

print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       1.00      0.99      0.99      4711
           1       0.99      1.00      0.99      4269

    accuracy                           0.99      8980
   macro avg       0.99      0.99      0.99      8980
weighted avg       0.99      0.99      0.99      8980
```

Rysunek 7.3 Dokładność i raport klasyfikacji dla modelu z LSTM – 2 epoki



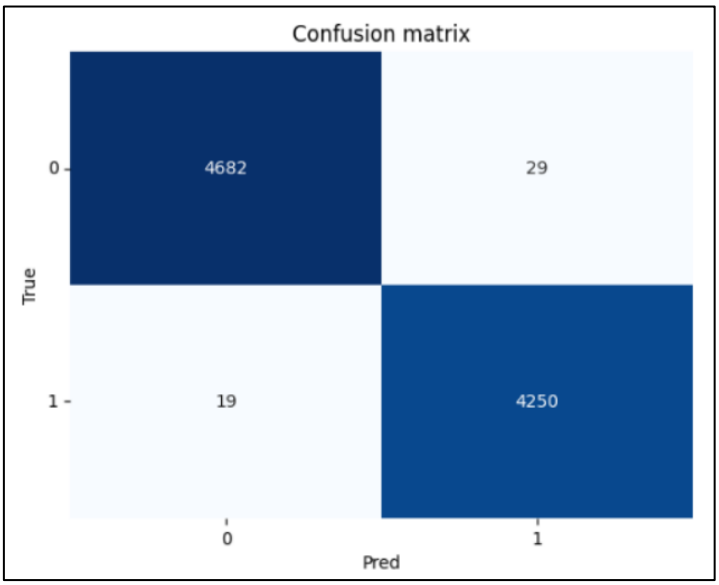
Rysunek 7.4 Macierz pomyłek dla modelu z LSTM – 2 epoki

Dodatkowa epoka znacząco poprawiła wydajność modelu, zmniejszając liczbę błędnych klasyfikacji oraz zwiększając precyzję, czułość i F1-score dla obu klas. Wskazuje to na korzystny wpływ dłuższego treningu na zdolność modelu do dokładnego rozróżniania między klasami.

Wynik dokładności i raport klasyfikacji dla trzeciego eksperymentu dla modelu z LSTM, w którym do szkolenia użyto 3 epok przedstawiono na rysunku 7.5. Natomiast macierz pomyłek dla tego samego eksperymentu obrazuje rysunek 7.6.

```
accuracy_score(y_test, y_pred)

0.9946547884187082

print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       1.00      0.99      0.99      4711
           1       0.99      1.00      0.99      4269

    accuracy                           0.99      8980
   macro avg       0.99      0.99      0.99      8980
weighted avg       0.99      0.99      0.99      8980
```

Rysunek 7.5 Dokładność i raport klasyfikacji dla modelu z LSTM – 3 epoki



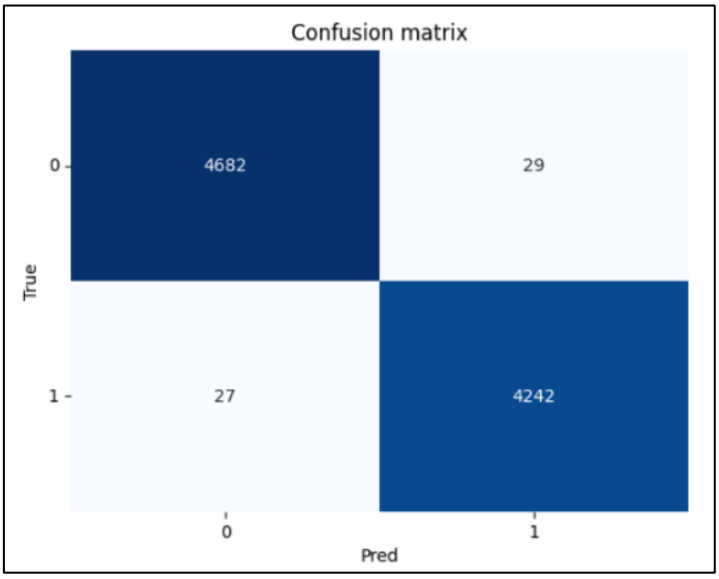
Rysunek 7.6 Macierz pomyłek dla modelu z LSTM – 3 epoki

Szkolenie modelu z dodatkową, trzecią epoką, przyniósł dalszą poprawę wyników, szczególnie w dokładności klasyfikacji. Choć wartości precision, recall i F1-score dla obu klas pozostały niemal niezmienione i bardzo wysokie, poprawa widoczna jest w macierzy pomyłek, gdzie liczba błędnych klasyfikacji klasy 0 znacząco się zmniejszyła. To wskazuje, że model stał się bardziej precyzyjny i dokładny dzięki dłuższemu treningowi, mimo że ogólne metryki pozostały stabilnie wysokie.

Wynik dokładności i raport klasyfikacji dla czwartego eksperymentu dla modelu z LSTM, w którym do szkolenia użyto 4 epok przedstawiono na rysunku 7.7. Natomiast macierz pomyłek dla tego samego eksperymentu obrazuje rysunek 7.8.

```
accuracy_score(y_test, y_pred)

0.9937639198218263

print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       0.99      0.99      0.99      4711
           1       0.99      0.99      0.99      4269

    accuracy                           0.99      8980
   macro avg       0.99      0.99      0.99      8980
weighted avg       0.99      0.99      0.99      8980
```

Rysunek 7.7 Dokładność i raport klasyfikacji dla modelu z LSTM – 4 epoki



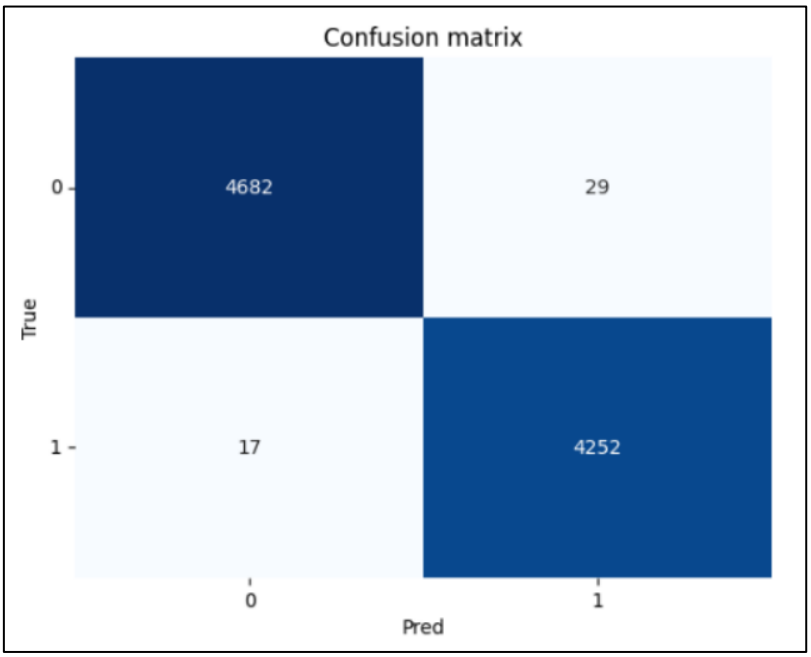
Rysunek 7.8 Macierz pomyłek dla modelu z LSTM – 4 epoki

Dodatkowa, czwarta epoka treningu dla modelu z LSTM nie przyniosła zauważalnej poprawy wyników w porównaniu do treningu przez 3 epoki. Dokładność modelu nieznacznie się zmniejszyła, a liczba błędnych klasyfikacji dla klasy 1 wzrosła. Wartości precision, recall i F1-score pozostały stabilnie wysokie, co może oznaczać, że model mógł osiągnąć swoje optymalne parametry po 3 epokach treningu.

Wynik dokładności i raport klasyfikacji dla piątego eksperymentu dla modelu z LSTM, w którym do szkolenia użyto 5 epok przedstawiono na rysunku 7.9. Natomiast macierz pomyłek dla tego samego eksperymentu obrazuje rysunek 7.10.

```
accuracy_score(y_test, y_pred)

0.9948775055679288

print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       1.00      0.99      1.00      4711
           1       0.99      1.00      0.99      4269

    accuracy                           0.99      8980
   macro avg       0.99      0.99      0.99      8980
weighted avg       0.99      0.99      0.99      8980
```

Rysunek 7.9 Dokładność i raport klasyfikacji dla modelu z LSTM – 5 epok



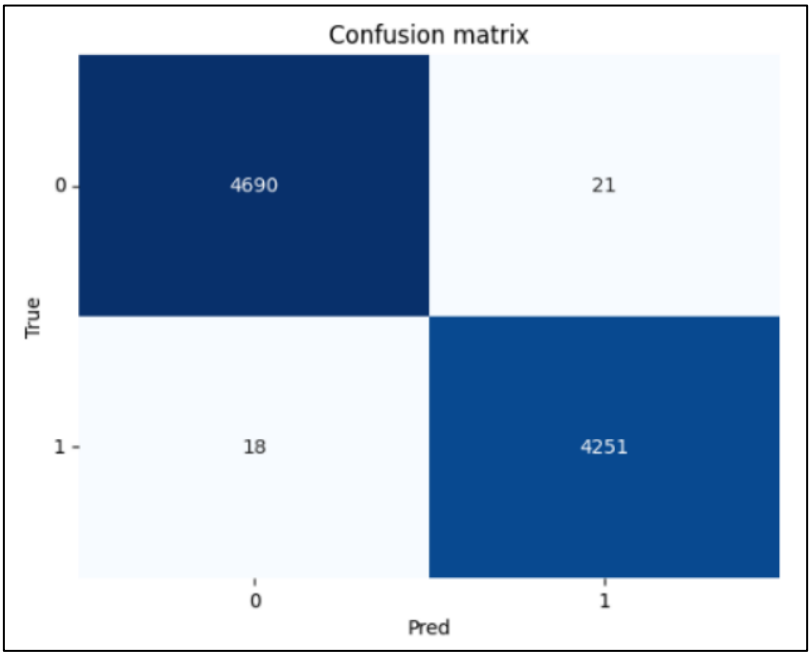
Rysunek 7.10 Macierz pomyłek dla modelu z LSTM – 5 epok

Dodatkowa, piąta epoka treningu dla modelu z LSTM przyniosła widoczną poprawę wyników w porównaniu do treningu przez 4 epoki. Dokładność modelu wzrosła, a liczba błędnych klasyfikacji dla klasy 1 zmniejszyła się. Wartości precision, recall i F1-score dla obu klas pozostały na bardzo wysokim poziomie, z poprawą w precision i F1-score dla klasy 0 oraz recall dla klasy 1. Dodatkowy trening pomógł modelowi osiągnąć jeszcze lepsze wyniki w klasyfikacji.

Wynik dokładności i raport klasyfikacji dla szóstego eksperymentu dla modelu z LSTM, w którym do szkolenia użyto 6 epok przedstawiono na rysunku 7.11. Natomiast macierz pomyłek dla tego samego eksperymentu obrazuje rysunek 7.12.

```
accuracy_score(y_test, y_pred)

0.9956570155902005

print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4711
           1       1.00      1.00      1.00      4269

    accuracy                           1.00      8980
   macro avg       1.00      1.00      1.00      8980
weighted avg       1.00      1.00      1.00      8980
```

Rysunek 7.11 Dokładność i raport klasyfikacji dla modelu z LSTM – 6 epok



Rysunek 7.12 Macierz pomyłek dla modelu z LSTM – 6 epok

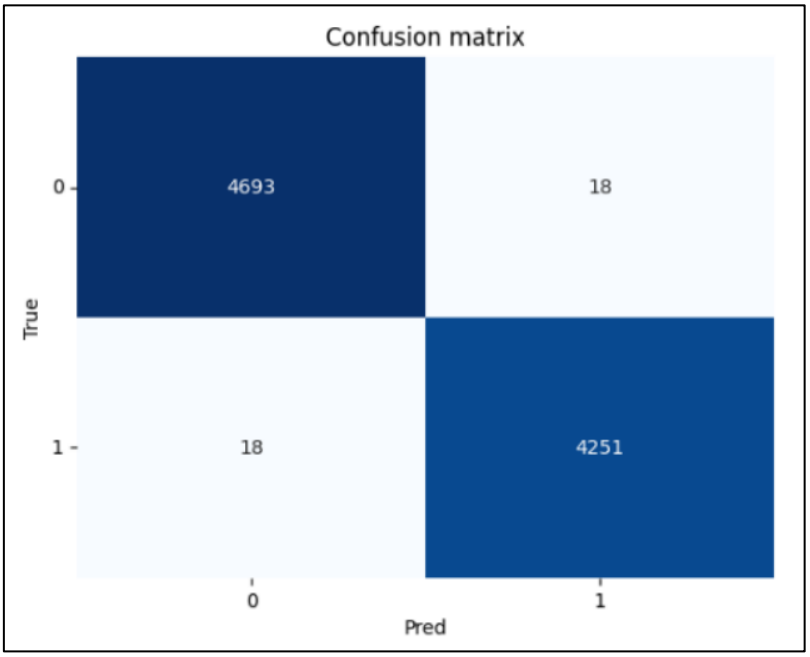
Dodatkowa, szósta epoka szkolenia modelu przyniosła dalszą poprawę wyników w porównaniu do treningu przez 5 epok. Dokładność modelu wzrosła, a wartości precision, recall i F1-score dla obu klas osiągnęły 1.00, co wskazuje na doskonałą wydajność modelu. Macierz pomyłek pokazuje zmniejszenie liczby błędnych klasyfikacji dla klasy 0, a to przyczynia się do ogólnej poprawy wyników. Model osiągnął idealne parametry po 6 epokach

treningu. Dodatkowy trening był korzystny i poprawił zdolność modelu do dokładnego rozróżniania między klasami.

Wynik dokładności i raport klasyfikacji dla siódmego eksperymentu dla modelu z LSTM, w którym do szkolenia użyto 7 epok przedstawiono na rysunku 7.13. Natomiast macierz pomyłek dla tego samego eksperymentu obrazuje rysunek 7.14.

```
accuracy_score(y_test, y_pred)

0.9959910913140312

print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4711
           1       1.00      1.00      1.00      4269

    accuracy                           1.00      8980
   macro avg       1.00      1.00      1.00      8980
weighted avg       1.00      1.00      1.00      8980
```

Rysunek 7.13 Dokładność i raport klasyfikacji dla modelu z LSTM – 7 epok



Rysunek 7.14 Macierz pomyłek dla modelu z LSTM – 7 epok

Dodatkowa, siódma epoka treningu dla modelu LSTM przyniosła minimalną poprawę wyników w porównaniu do treningu przez 6 epok. Dokładność modelu wzrosła nieznacznie, a liczba błędnych klasyfikacji dla klasy 0 zmniejszyła się – oznacza to lepszą wydajność modelu. Wartości precision, recall i F1-score dla obu klas pozostały na poziomie 1.00, co oznacza

doskonałą wydajność klasyfikatora. Model osiągnął bardzo wysoką jakość klasyfikacji po 6 epokach, a dodatkowa epoka przyniosła tylko minimalne ulepszenia, a to oznacza, że model jest już dobrze wytrenowany i dalsze epoki mogą przynieść jedynie nieznaczne poprawy.

7.3 Klasyfikacja przy użyciu BERT

W tym podrozdziale zostaną przedstawione wyniki działania modelu BERT. Udało się przeprowadzić trzy eksperymenty ze względu na ograniczenia spowodowane zasobami obliczeniowymi oraz czasem dostępnym na wykonanie obliczeń. Pomimo wykupienia wersji Google Colab Pro (za 12,29 USD na miesiąc), nie udało się przeprowadzić treningu na więcej niż 3 epokach. Google Colab, nawet w wersji Pro, posiada pewne ograniczenia związane z zasobami obliczeniowymi dostępnymi dla użytkowników. Wersja Pro oferuje zwiększoną ilość pamięci RAM oraz dostęp do lepszych jednostek GPU, jednak nadal istnieją limity dotyczące długości sesji oraz maksymalnego wykorzystania zasobów. Długotrwałe obliczenia mogą powodować automatyczne zakończenie sesji, co w praktyce ogranicza możliwość przeprowadzenia długich i zasobochłonnych eksperymentów, takich jak właśnie trening modelu BERT na większej liczbie epok.

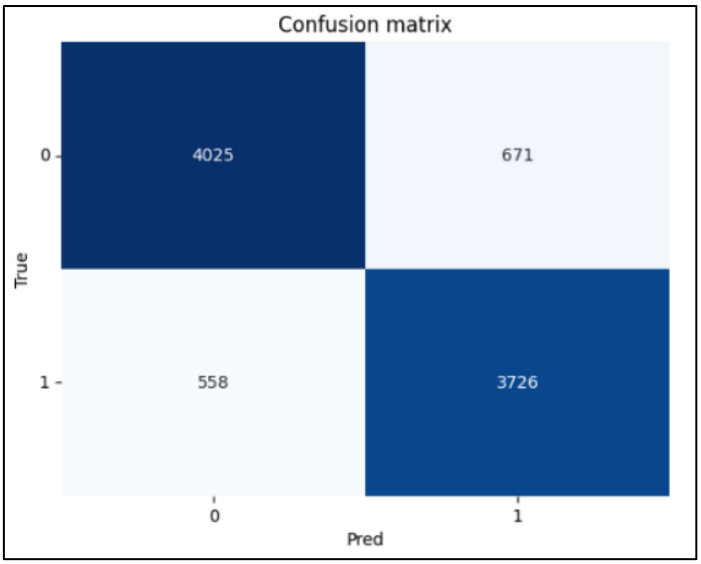
Wynik dokładności dla pierwszego eksperymentu dla modelu BERT, w którym do szkolenia użyto 1 epoki przedstawiono na rysunku 7.15. Raport klasyfikacji widnieje na rysunku 7.16, natomiast macierz pomyłek dla tego samego eksperymentu obrazuje rysunek 7.17.

```
281/281 [==============================] - 987s 4s/step - loss: 0.3439 - accuracy: 0.8631
```

Rysunek 7.15 Dokładność dla modelu BERT – 1 epoka

	precision	recall	f1-score	support
0	0.88	0.86	0.87	4696
1	0.85	0.87	0.86	4284
accuracy			0.86	8980
macro avg	0.86	0.86	0.86	8980
weighted avg	0.86	0.86	0.86	8980

Rysunek 7.16 Raport klasyfikacji dla modelu BERT – 1 epoka



Rysunek 7.17 Macierz pomyłek dla modelu BERT – 1 epoka

Model BERT po jednej epoce szkolenia wykazuje dobrą skuteczność w klasyfikacji, z równowagą między precyzją a czułością dla obu klas. Wysoka dokładność i zbalansowane wyniki w raporcie klasyfikacji oraz macierzy pomyłek sugerują, że model dobrze nauczył się rozróżniać obie klasy. Dalsze szkolenie modelu przez więcej epok mogłoby poprawić jego wydajność, zwłaszcza jeśli chodzi o minimalizowanie liczby błędnych klasyfikacji.

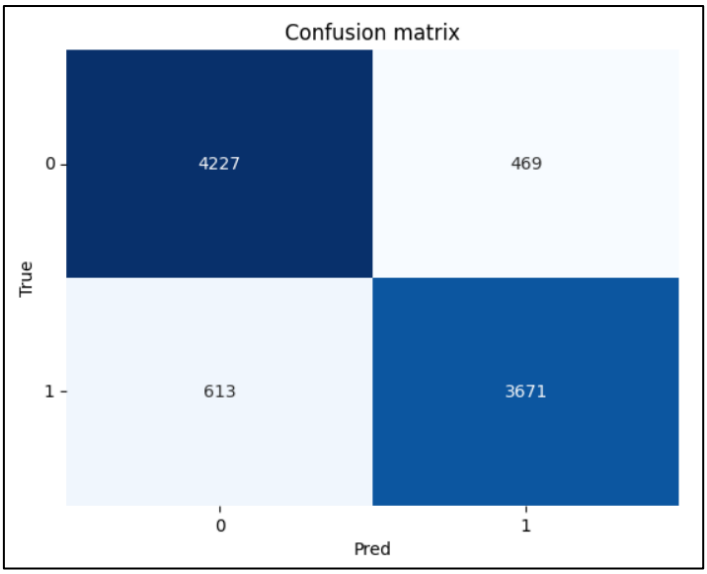
Wynik dokładności dla drugiego eksperymentu dla modelu BERT, w którym do szkolenia użyto 2 epok przedstawiono na rysunku 7.18. Raport klasyfikacji widnieje na rysunku 7.19, natomiast macierz pomyłek dla tego samego eksperymentu obrazuje rysunek 7.20.

```
281/281 [==============================] - 4129s 15s/step - loss: 0.3014 - accuracy: 0.8795
```

Rysunek 7.18 Dokładność dla modelu BERT – 2 epoki

	precision	recall	f1-score	support
0	0.87	0.90	0.89	4696
1	0.89	0.86	0.87	4284
accuracy			0.88	8980
macro avg	0.88	0.88	0.88	8980
weighted avg	0.88	0.88	0.88	8980

Rysunek 7.19 Raport klasyfikacji dla modelu BERT – 2 epoki



Rysunek 7.20 Macierz pomyłek dla modelu BERT – 2 epoki

Trenowanie modelu przez dwie epoki znacząco poprawiło jego skuteczność, zwłaszcza jeśli chodzi o równowagę między precision a recall dla obu klas. Wyższa dokładność oraz lepsze wyniki precision, recall i f1-score potwierdzają, że dłuższy czas szkolenia modelu doprowadził do lepszych wyników w zadaniach klasyfikacyjnych. Chociaż liczba błędnych klasyfikacji wzrosła nieznacznie przy dwóch epokach, ogólna skuteczność modelu wzrosła. Wskazane jest kontynuowanie trenowania modelu, aby dalej poprawiać jego wyniki.

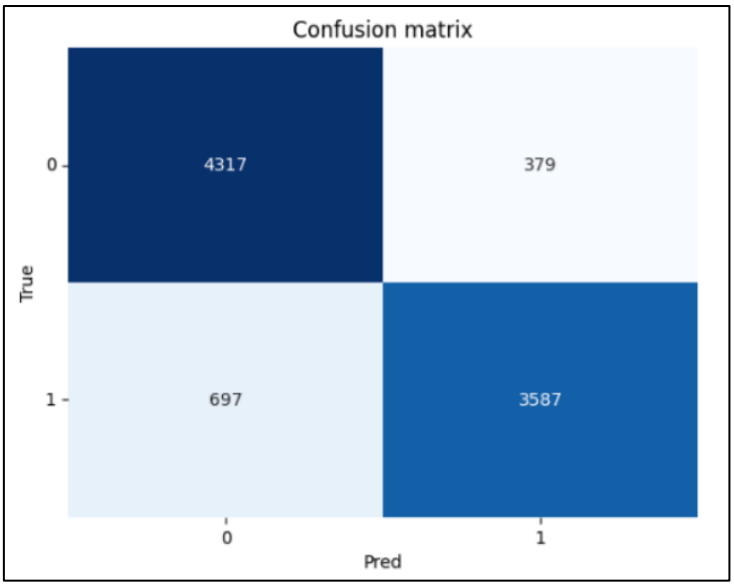
Wynik dokładności dla trzeciego eksperymentu dla modelu BERT, w którym do szkolenia użyto 3 epok przedstawiono na rysunku 7.21. Raport klasyfikacji widnieje na rysunku 7.22, natomiast macierz pomyłek dla tego samego eksperymentu obrazuje rysunek 7.23.

```
281/281 [==============================] - 510s 2s/step - loss: 0.2906 - accuracy: 0.8802
```

Rysunek 7.21 Dokładność dla modelu BERT – 3 epoki

	precision	recall	f1-score	support
0	0.86	0.92	0.89	4696
1	0.90	0.84	0.87	4284
accuracy			0.88	8980
macro avg	0.88	0.88	0.88	8980
weighted avg	0.88	0.88	0.88	8980

Rysunek 7.22 Raport klasyfikacji dla modelu BERT – 3 epoki



Rysunek 7.23 Macierz pomyłek dla modelu BERT – 3 epoki

Dodatkowa trzecia epoka szkolenia nieznacznie poprawiła skuteczność modelu, szczególnie w przypadku wykrywania klasy 0, co zostało odzwierciedlone wyższą wartością recall dla tej klasy. Ogólna dokładność nieznacznie wzrosła, tak jak i precyzja dla klasy 1. Natomiast recall nieco spadł, a to może wskazywać na potrzebę dalszej optymalizacji modelu.

8. Wnioski z przeprowadzonych eksperymentów

Przeprowadzone badania dotyczące budowy dwóch modeli do klasyfikacji fake newsów – LSTM oraz BERT – wykazały różne wyniki skuteczności w zależności od zastosowanej architektury oraz liczby epok treningowych. Oba modele przeprowadzono na tym samym zbiorze danych, co pozwala na bezpośrednie porównanie ich wyników. Wyniki pokazały, że model zbudowany na podstawie LSTM osiągnął wyższą dokładność klasyfikacji w porównaniu do modelu BERT, co jest zaskakujące biorąc pod uwagę teoretyczne założenia i potencjał BERT.

Model z LSTM wykazał stały wzrost dokładności wraz z liczbą epok, osiągając najwyższą dokładność 0.995991 po siedmiu epokach. Wysoka dokładność LSTM była wynikiem odpowiedniego doboru parametrów oraz skutecznego przetwarzania sekwencji tekstowych. Do imponującego wyniku szkolenia bezsprzecznie przyczyniło się użycie Word2Vec, który uwzględnia kontekstowe zależności między słowami, co pomaga modelowi z LSTM lepiej zrozumieć strukturę oraz znaczenie tekstu.

Pomimo teoretycznej przewagi architektury BERT w zadaniach przetwarzania języka naturalnego, jego dokładność była niższa niż modelu z LSTM, z maksymalnym wynikiem 0.8802 po trzech epokach. BERT jest zaawansowanym modelem, który zazwyczaj wymaga większej liczby epok treningowych oraz dostrojenia, aby osiągnąć optymalne wyniki. Trening przez trzy epoki może być niewystarczający dla tak skomplikowanego modelu jak BERT, który zwykle wymaga więcej czasu na konwergencję (osiągnięcie stabilnego stanu, w którym zmiany wartości funkcji kosztu stają się minimalne z każdą kolejną iteracją treningową). Dodatkowo BERT jest modelem o dużej liczbie parametrów i mógł wymagać bardziej złożonego procesu strojenia hiperparametrów. Warto się zastanowić również nad technikami optymalizacji – model BERT może wymagać bardziej złożonych technik optymalizacji, takich jak regularyzacja czy zastosowanie bardziej zaawansowanych schematów uczenia, aby osiągnąć lepsze wyniki.

Oba modele wykazały wysoką skuteczność w klasyfikacji fake newsów, jednak model z LSTM osiągnął wyższą dokładność w krótszym czasie treningowym. Pomimo swojej teoretycznej przewagi, model BERT wymagałby dalszych badań i dostrojenia, aby zbliżyć się do wyników uzyskanych przez LSTM.

9. Podsumowanie

Celem niniejszej pracy magisterskiej było opracowanie, implementacja oraz porównanie modeli sztucznej inteligencji w kontekście skutecznego rozpoznawania fake newsów. W ramach badań przeprowadzono analizę efektywności dwóch modeli – z użyciem LSTM oraz BERT, stosując zaawansowane techniki przetwarzania języka naturalnego. Wprowadzenie obu modeli umożliwiło automatyczne rozpoznawanie wzorców w danych tekstowych, co znacząco zwiększyło skuteczność wykrywania dezinformacji.

Do realizacji celu praca skupiła się na kilku kluczowych zadaniach: zdobyciu wiedzy na temat metod wykrywania fake newsów, zapoznaniu się z technikami przetwarzania języka naturalnego, przygotowaniu i oczyszczeniu zbiorów danych, wyborze odpowiednich technik tokenizacji i osadzania słów, a także implementacji modeli z LSTM i BERT. Następnie przeprowadzono eksperymenty, w których oceniano efektywność modeli w zależności od liczby epok, analizując wyniki za pomocą dokładności, raportów klasyfikacji oraz macierzy pomyłek.

Wnioski z przeprowadzonych eksperymentów wskazują, że model LSTM osiągnął wyższą dokładność klasyfikacji fake newsów niż model BERT, co było zaskakujące, biorąc pod uwagę teoretyczny potencjał BERT. Model LSTM, dzięki odpowiedniemu doborowi parametrów oraz zastosowaniu Word2Vec, uzyskał najwyższą dokładność 0.995991 po siedmiu epokach. Z kolei model BERT, mimo swojej zaawansowanej architektury, osiągnął dokładność 0.8802 po trzech epokach, co sugeruje, że wymagałby dalszego dostrojenia i większej liczby epok treningowych, aby osiągnąć optymalne wyniki. Dodatkowo, należy zauważyć, że za niskie parametry środowiska wykonawczego ograniczyły możliwość przeprowadzenia większej liczby eksperymentów na modelu BERT. Niemniej jednak, teoretyczne założenia oraz zaawansowane właściwości BERT pozycjonują go jako potencjalnie bardzo mocne narzędzie do detekcji dezinformacji. Dalsze badania i optymalizacja parametrów mogą przynieść lepsze wyniki, umożliwiając mu pełne wykorzystanie swojego potencjału.

Podsumowując, oba modele wykazały wysoką skuteczność w klasyfikacji fake newsów, jednak model LSTM okazał się bardziej efektywny w krótszym czasie treningowym. W przyszłości, model BERT mógłby wymagać bardziej zaawansowanych technik optymalizacji oraz dłuższego treningu, aby dorównać wynikom uzyskanym przez LSTM.

BIBLIOGRAFIA

[Antoun W., 2020] Antoun W., et al. *"State of the Art Models for Fake News Detection Tasks."* IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT'20), April 2020, DOI: https://doi.org/10.1109/ICIoT48696.2020.9089487

[Attention Is All You Need, 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. "Attention Is All You Need". arXiv, 15 pages, 5 figures. DOI: https://doi.org/10.48550/arXiv.1706.03762.

[Architecture of a LSTM Unit, 2024] *An Intuitive Explanation of LSTM.* [online] Dostępne na: <https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c>[Udostępniono 20.06.2024].

[BERT model architecture, 2024] *BERT model architecture.* [online] Dostępne na: <https://www.researchgate.net/figure/BERT-model-architecture_fig2_348214408>[Udostępniono 20.06.2024].

[CNN, 2024] *Convolutional Neural Network (CNN).* [online] Dostępne na: <https://www.linkedin.com/pulse/convolutional-neural-network-cnn-aaron-shajan-oxwfc/>[Udostępniono 20.06.2024].

[**CSV, 2022]** What is a CSV file? [online] Dostępne na: <https://docs.fileformat.com/spreadsheet/csv/> [Udostępniono 20.06.2024].

[**Data Cleaning, 2024]** *Data Cleaning: Techniques & Best Practices for 2023.* [online] Dostępne na: <https://technologyadvice.com/blog/information-technology/datacleaning/>[Udostępniono 20.06.2024].

[Fact-checking, 2024] *Czym jest fact-checking? – Zarys inicjatyw na świecie i w Polsce.* [online] Dostępne na: < https://cyberpolicy.nask.pl/czym-jest-fact-checking-zarys-inicjatyw-na-swiecie-i-w-polsce/>[Udostępniono 20.06.2024].

[Fake newsy, 2024] *Rozpoznawanie nieprawdziwych informacji.* Dostępne na: <https://www.gov.pl/web/baza-wiedzy/rozpoznawanie-nieprawdziwych-informacji>[Udostępniono 20.06.2024].

[FastText, 2024] *FastText.* Dostępne na: <https://fasttext.cc/ >[Udostępniono 20.06.2024].

[Gensim, 2024] *Gensim*. [online] Dostępne na: <https://pypi.org/project/gensim/> [Udostępniono 20.06.2024].

[GloVe, 2024] *GloVe: Global Vectors for Word Representation*. [online] Dostępne na: <https://nlp.stanford.edu/projects/glove/> [Udostępniono 20.06.2024].

[Google BERT, 2024] *Google BERT: Understanding the Architecture*. [online] Dostępne na: <https://www.theaidream.com/post/google-bert-understanding-the-architecture> [Udostępniono 20.06.2024].

[Google Colab, 2024] *Google Colaboratory*. [online] Dostępne na: <https://colab.google/> [Udostępniono 20.06.2024].

[**Harvard, 2024**] *Anglia Ruskin University Library – Harvard System*. [online] Dostępne na: <https://library.aru.ac.uk/referencing/harvard.htm> [Udostępniono 20.06.2024].

[Influence of fake news, 2024] *Influence of fake news in Twitter during the 2016 US presidential election*. [online] Dostępne na: < https://www.nature.com/articles/s41467-018-07761-2> [Udostępniono 20.06.2024].

[Jolly A., Kumar S., 2022] Jolly A., Kumar S. *"Fake News Detection in Different Applications: State-of-the-art and Future Directions."* Delhi Technological University. Research Article. 7 September 2022. DOI: https://doi.org/10.21203/rs.3.rs-2024922/v1

[Kaggle, 2024] *Kaggle*. [online] Dostępne na: <https://www.kaggle.com/>[Udostępniono 20.06.2024]

[Keras, 2024] *Keras*. [online] Dostępne na: <https://keras.io/>[Udostępniono 20.06.2024]

[LSTM, 2024] *Understanding LSTM Networks*. Dostępne na: <https://colah.github.io/posts/2015-08-Understanding-LSTMs> [Udostępniono 20.06.2024]

[Matplotlib, 2024] *Matplotlib*. [online] Dostępne na: < https://matplotlib.org/>[Udostępniono 20.06.2024]

[NLP, 2024] *What is NLP (natural language processing)?* [online] Dostępne na: <https://www.ibm.com/topics/natural-language-processing>[Udostępniono 20.06.2024]

[NLTK, 2024] *NLTK (Natural Language Toolkit)*. [online] Dostępne na: <https://www.nltk.org/>[Udostępniono 20.06.2024]

[Numpy, 2024] *Numpy*. [online] Dostępne na: < https://numpy.org/>[Udostępniono 20.06.2024]

[Pandas, 2024] *Pandas*. [online] Dostępne na: < https://pandas.pydata.org/>[Udostępniono 20.06.2024]

[Patel S., 2024] *Fake News Detection*. [online] Dostępne na: <https://www.kaggle.com/code/therealsampat/fake-news-detection>[Udostępniono 20.06.2024]

[Preprocess_kgptalkie, 2024] *Preprocessing Text Python Package*. [online] Dostępne na: <https://github.com/laxmimerit/preprocess_kgptalkie>[Udostępniono 20.06.2024]

[Python, 2024] *Python*. [online] Dostępne na: < https://www.python.org/>[Udostępniono 20.06.2024]

[Random Forest, 2024] *Random Forests.* [online] Dostępne na: <https://medium.com/@roiyeho/random-forests-98892261dc49>[Udostępniono 20.06.2024]

[re, 2024] *re (Regular Expressions operations).* [online] Dostępne na: <https://docs.python.org/3/library/re.html>[Udostępniono 20.06.2024]

[Scikit-learn, 2024] *Scikit-learn*. [online] Dostępne na: < https://scikit-learn.org/stable/>[Udostępniono 20.06.2024]

[Seaborn, 2024] *Seaborn*. [online] Dostępne na: <https://seaborn.pydata.org/>[Udostępniono 20.06.2024]

[Sztuczna inteligencja, 2024] *What Is Artificial Intelligence? Definition, Uses, and Types*. [online] Dostępne na: <https://www.coursera.org/articles/what-is-artificial-intelligence?>[Udostępniono 20.06.2024]

[TensorFlow, 2024] *TensorFlow*. [online] Dostępne na: <https://www.tensorflow.org/?hl=pl>[Udostępniono 20.06.2024]

[TensorFlow Hub, 2024] *TensorFlow Hub*. [online] Dostępne na: <https://www.tensorflow.org/hub?hl=pl>[Udostępniono 20.06.2024]

[Word2vec, 2024] *Word2Vec For Word Embeddings -A Beginner's Guide*. [online] Dostępne na: <https://www.analyticsvidhya.com/blog/2021/07/word2vec-for-word-embeddings-a-beginners-guide/>[Udostępniono 20.06.2024]

[Yuan L., 2023] Yuan L., et al. *"Sustainable Development of Information Dissemination: A Review of Current Fake News Detection Research and Practice."*. Systems, 11(9), 458, 4 September 2023. DOI: https://doi.org/10.3390/systems11090458.