Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Informatyki

Sprawozdanie z realizacji projektu

Wykrywanie fałszywych wiadomośći z wykorzystaniem wielkich modeli językowych

Jerzy Muszyński
Numer albumu 327383

WARSZAWA 2026

Spis treści

1. Wstęp

W dobie powszechnego dostępu do mediów społecznościowych i gwałtownego przepływu informacji, zjawisko fałszywych wiadomości (*fake news*) stało się jednym z kluczowych zagrożeń dla debaty publicznej. Automatyzacja procesu weryfikacji treści jest niezbędna, gdyż ręczna moderacja nie nadąża za skalą generowanych danych. Celem niniejszego projektu jest opracowanie i porównanie skuteczności systemów automatycznej klasyfikacji tekstu, mających na celu odróżnienie wiadomości prawdziwych od fałszywych.

1.1. Definicja problemu

Problem został zdefiniowany jako zadanie binarnej klasyfikacji tekstu. Na wejściu system otrzymuje tekst artykułu (często połączony z tytułem), a na wyjściu ma wygenerować etykietę: `Real` (wiadomość prawdziwa) lub `Fake` (wiadomość fałszywa).

Przykład danych wejściowych:

Title: US to impose sanctions on... Content: The United States announced today that...

Oczekiwany rezultat:

Etykieta: `Real` (1)

1.2. Wkład własny

W ramach projektu zrealizowano następujące zadania:

1. Przygotowanie potoku przetwarzania danych dla zbioru *Fake News Detection Dataset*.
2. Implementacja i optymalizacja trzech podejść:
    - Metoda klasyczna: TF-IDF z Regresją Logistyczną.
    - Model typu Encoder-Only: Dostrojenie (Fine-tuning) modelu DistilBERT.
    - Model typu Decoder-Only: Adaptacja modelu Qwen-2.5 z techniką LoRA.
3. Przeprowadzenie eksperymentów optymalizacyjnych (Grid Search).
4. Eksperymentalne porównanie metod i wizualizacja przestrzeni cech (t-SNE/UMAP).

2. Opis rozwiązania

2.1. Zbiór danych

Wykorzystano zbiór danych ErfanMoosaviMonazzah/fake-news-detection-dataset-English dostępny w HuggingFace. Jest to zbiór zbalansowany. Struktura danych oparta jest na obiekcie `DatasetDict`. Liczebność podzbiorów przedstawiono w Tabeli 2.1.

Tabela 2.1. Statystyki liczebności zbioru danych.

Podzbiór	Rola	Liczba rekordów
`train`	Trenowanie modelu	30 000
`validation`	Dobór hiperparametrów	6 000
`test`	Ostateczna ocena	8 267
Suma		**44 267**

2.1.1. Przetwarzanie wstępne (Preprocessing)

Wstępne przetwarzanie obejmowało konkatenację kolumn `title` oraz `text`. Pozwala to modelom na analizę pełnego kontekstu.

Dane surowe:

Tytuł: *BIKERS FOR TRUMP: "Not Going To Put Up With" Violent Leftists Disrupting Cleveland GOP Convention... Will Protect Delegates "Right To Peacefully Assemble"*
Treść: *Veterans are the backbone of the biker community We are patriots We love our cops The antithesis of the Black Lives Matter radicals Breitbart Exclusive: A large group of patriotic motorcycle enthusiasts will be among the visitors to Cleveland Ohio next week for the Republican National Committee meeting that will nominate business mogul Donald Trump to be the Republican nominee for President of the United States...*
Etykieta: 0 (Fake)

Dane wejściowe dla modelu:

BIKERS FOR TRUMP: "Not Going To Put Up With" Violent Leftists Disrupting Cleveland GOP Convention... Will Protect Delegates "Right To Peacefully Assemble" Veterans are the backbone of the biker community...

3. TF-IDF + Regresja Logistyczna

Jako linię bazową (baseline) zastosowano wektoryzację TF-IDF. Przeprowadzono przeszukiwanie siatki (Grid Search) dla kluczowych parametrów.

3.1. Dobór hiperparametrów (Grid Search)

Zbadano następującą przestrzeń parametrów:

- `max_features` $\in \{10000, 30000, 50000\}$
- `ngram_range` $\in \{(1,1), (1,2)\}$
- `min_df` $\in \{2, 5, 10\}$
- `max_df` $\in \{0.8, 0.9, 0.95\}$

Tabela 3.1. 10 najlepszych konfiguracji modelu TF-IDF (posortowane wg Accuracy).

max_feat	ngram	min_df	max_df	Accuracy	Precis.	Recall	F1
10000	(1, 2)	2	0.80	**0.9863**	0.9816	0.9904	0.9860
10000	(1, 1)	2	0.90	0.9837	0.9766	0.9900	0.9833
10000	(1, 1)	10	0.90	0.9835	0.9766	0.9897	0.9831
10000	(1, 1)	2	0.95	0.9833	0.9776	0.9883	0.9829
10000	(1, 1)	5	0.90	0.9833	0.9763	0.9897	0.9829
10000	(1, 1)	5	0.95	0.9833	0.9776	0.9883	0.9829
10000	(1, 1)	10	0.95	0.9833	0.9776	0.9883	0.9829
10000	(1, 1)	2	0.80	0.9830	0.9763	0.9890	0.9826
10000	(1, 1)	5	0.80	0.9830	0.9763	0.9890	0.9826
10000	(1, 1)	10	0.80	0.9828	0.9763	0.9887	0.9824

3.2. Ewaluacja najlepszego modelu

W tej sekcji przedstawiono szczegółowe wyniki dla najlepszej znalezionej konfiguracji parametrów. Analizę przeprowadzono na trzech podzbiorach danych: treningowym, walidacyjnym oraz testowym.

3.3. Wyniki metryk klasyfikacji

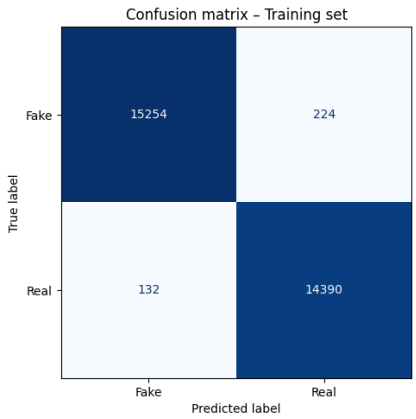
W Tabeli 3.2 zestawiono wartości precyzji, czułości (recall) oraz miary F1 dla obu klas (`Fake` i `Real`).

Tabela 3.2. Szczegółowe wyniki klasyfikacji dla najlepszej konfiguracji.

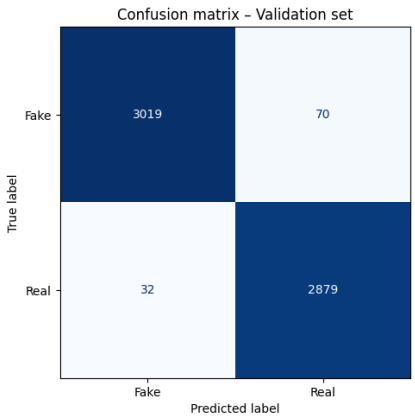
Zbiór danych	Klasa	Precyzja	Czułość (Recall)	F1-Score
Treningowy	Fake	0.991	0.986	0.988
(Acc: 0.988)	Real	0.985	0.991	0.988
Walidacyjny	Fake	0.990	0.977	0.983
(Acc: 0.983)	Real	0.976	0.989	0.983
Testowy	Fake	0.986	0.979	0.983
(Acc: 0.982)	Real	0.977	0.985	0.981

3.4. Macierze pomyłek

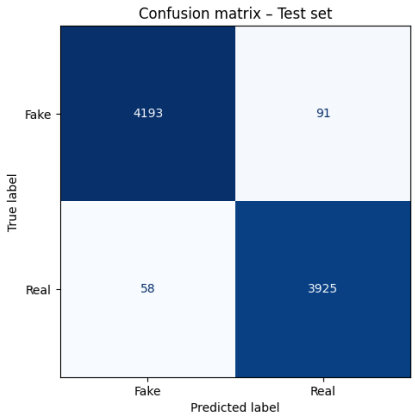
Poniższe rysunki przedstawiają macierze pomyłek (Confusion Matrices), które obrazują liczbę poprawnych i błędnych dopasowań dla każdego ze zbiorów.



(a) Zbiór Treningowy



(b) Zbiór Walidacyjny



(c) Zbiór Testowy

Rysunek 3.1. Macierze pomyłek dla poszczególnych zbiorów danych.

3.5. Wnioski z ewaluacji

Na podstawie powyższych danych można sformułować następujące wnioski:

- **Wysoka skuteczność:** Model osiągnął bardzo wysoką dokładność (*Accuracy*) na poziomie około 98% we wszystkich zbiorach danych. Oznacza to, że system poprawnie klasyfikuje niemal każdy artykuł.
- **Brak przeuczenia (Overfitting):** Różnica w wynikach między zbiorem treningowym (Accuracy 0.988) a zbiorem testowym (Accuracy 0.982) jest minimalna i wynosi mniej niż 1 punkt procentowy. Świadczy to o tym, że model dobrze generalizuje wiedzę i nie nauczył się danych treningowych "na pamięć".
- **Balans błędów:** Analizując macierz pomyłek dla zbioru testowego (Rysunek 3.1c), widzimy, że:
  — Model pomylił się 91 razy klasyfikując `Fake` jako `Real` (False Negative).
  — Model pomylił się 58 razy klasyfikując `Real` jako `Fake` (False Positive).

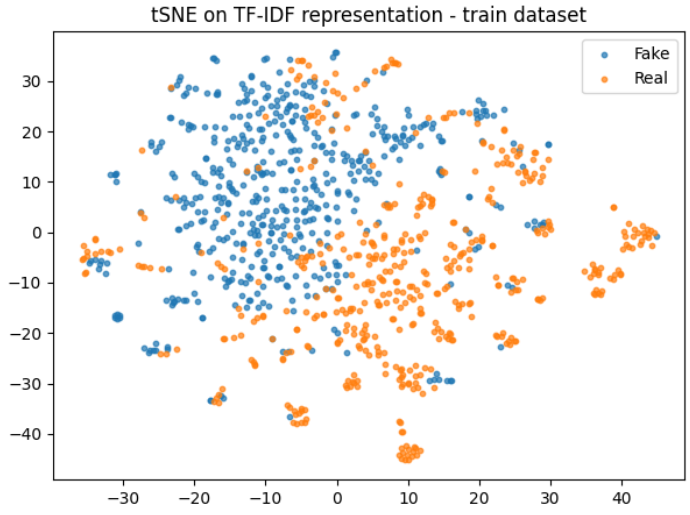
  Liczby te są niskie i zbliżone do siebie, co sugeruje, że model nie jest stronniczy w kierunku żadnej z klas.
- **Stabilność:** Wyniki na zbiorze walidacyjnym i testowym są niemal identyczne (F1-score dla klasy Fake wynosi 0.983 w obu przypadkach), co potwierdza wiarygodność przeprowadzonej optymalizacji parametrów.

3.6. Wizualizacja przestrzeni cech (t-SNE)

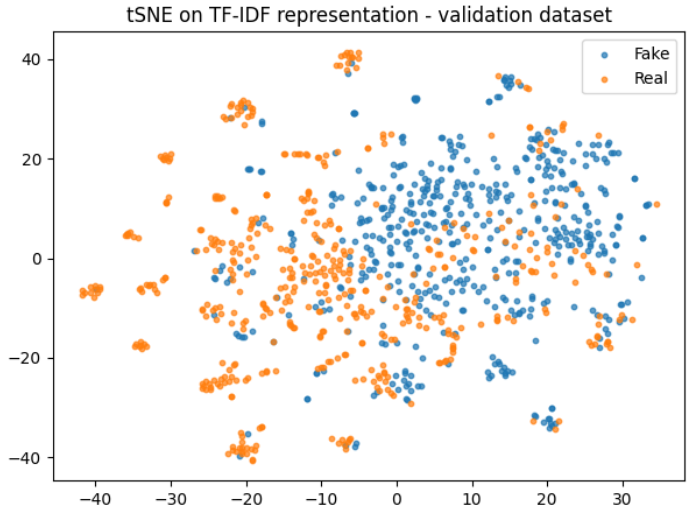
Aby lepiej zrozumieć, jak model "widzi"artykuły, zastosowano technikę t-SNE. Pozwala ona spłaszczyć skomplikowane dane matematyczne do prostego wykresu 2D, gdzie każdy punkt reprezentuje jeden artykuł.

Zasada interpretacji jest prosta:

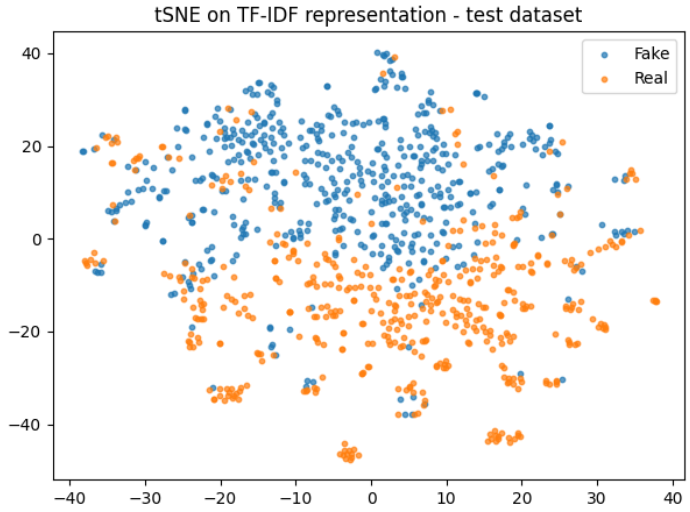
- Jeśli punkty o różnych kolorach są wymieszane, modelowi trudno odróżnić prawdę od fałszu.
- Jeśli punkty tworzą oddzielne skupiska (klastry) dla każdego koloru, model łatwo rozróżnia klasy.



(a) Zbiór Treningowy



(b) Zbiór Walidacyjny



(c) Zbiór Testowy

Rysunek 3.2. Wizualizacja t-SNE dla cech TF-IDF. Kolor niebieski to `Fake`, pomarańczowy to `Real`.

3.6.1. Wnioski z wizualizacji

Analiza powyższych wykresów (Rysunek 3.2) pozwala na wyciągnięcie następujących wniosków:

1. **Wyraźna separacja:** Widoczne jest bardzo wyraźne oddzielenie punktów niebieskich (wiadomości fałszywe) od pomarańczowych (wiadomości prawdziwe). Tworzą one dwie osobne "wyspy". Oznacza to, że słownictwo używane w Fake Newsach jest na tyle specyficzne, że nawet prosta metoda TF-IDF potrafi je skutecznie oddzielić od rzetelnych artykułów.
2. **Spójność danych:** Wykresy dla zbioru treningowego, walidacyjnego i testowego wyglądają bardzo podobnie. Struktura "chmur"jest zachowana. Potwierdza to, że model jest stabilny i będzie działał poprawnie na nowych, nieznanych wcześniej danych.
3. **Obszary niepewności:** Na granicy obu chmur widać niewielką strefę, gdzie punkty niebieskie i pomarańczowe się mieszają. To właśnie te artykuły są najtrudniejsze do sklasyfikowania i to tam model popełnia nieliczne błędy, które widzieliśmy wcześniej w macierzach pomyłek.

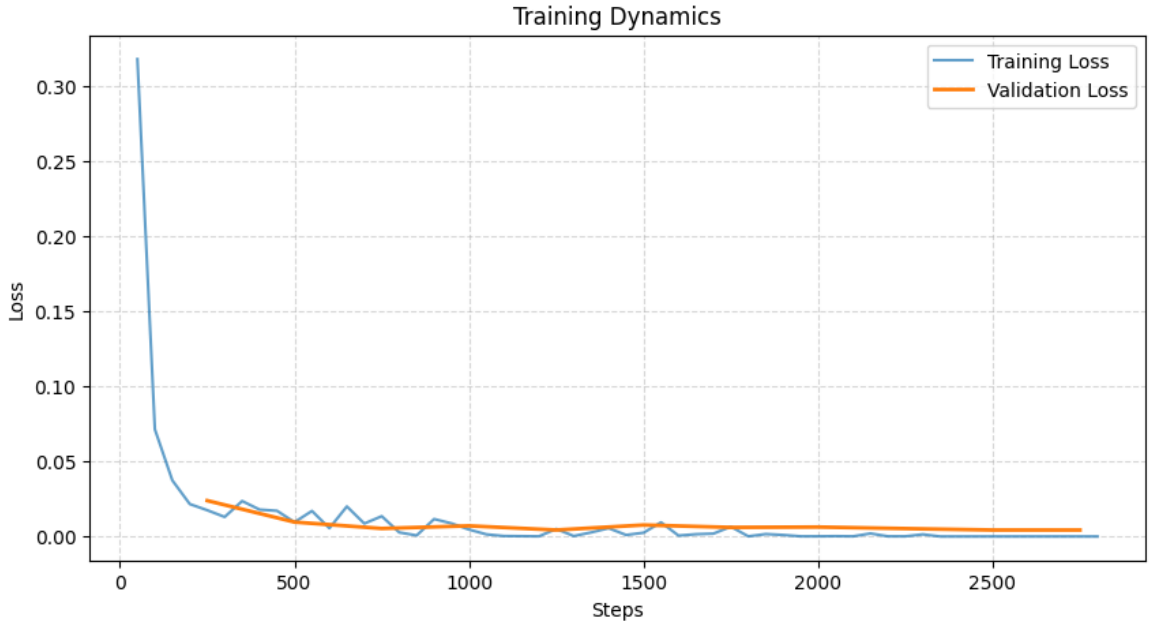
4. DistilBERT (Model Encoder-Only)

W drugim podejściu zastosowano model oparty na architekturze Transformer – DistilBERT. Jest to model typu *Encoder-Only*, który potrafi analizować głębokie znaczenie (semantykę) tekstu, a nie tylko występowanie pojedynczych słów.

Model został poddany procesowi dostrajania (*Fine-Tuning*) na zbiorze treningowym. Poniżej przedstawiono analizę przebiegu treningu oraz uzyskane wyniki.

4.1. Przebieg treningu

Wykres 4.1 przedstawia dynamikę uczenia się modelu (Training Dynamics).



Rysunek 4.1. Wykres funkcji straty (Loss) dla zbioru treningowego i walidacyjnego.

Można zaobserwować, że:

- Funkcja straty (*Loss*) spada bardzo gwałtownie już w pierwszych krokach treningu.
- Strata walidacyjna (linia pomarańczowa) utrzymuje się na bardzo niskim poziomie i nie rośnie, co świadczy o tym, że model uczy się stabilnie i nie traci zdolności do generalizacji.

4.2. Wyniki ewaluacji

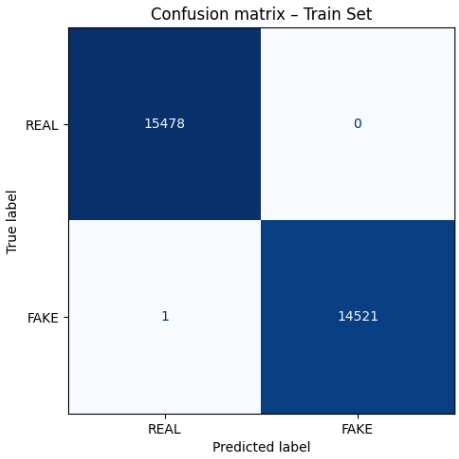
Model DistilBERT osiągnął niemal perfekcyjne wyniki na wszystkich zbiorach danych. Szczegółowe metryki przedstawiono w Tabeli 4.1.

Tabela 4.1. Wyniki klasyfikacji dla modelu DistilBERT.

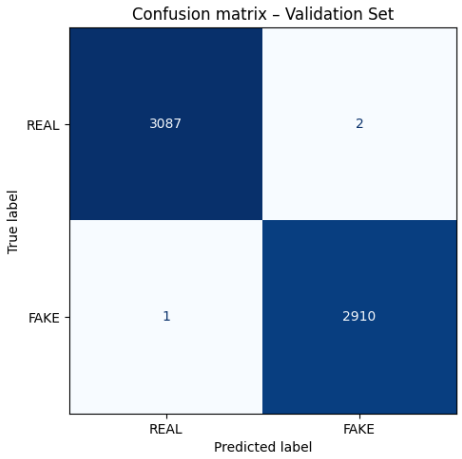
Zbiór danych	Klasa	Precyzja	Czułość (Recall)	F1-Score
Treningowy	Fake	1.000	1.000	1.000
(Acc: 1.000)	Real	1.000	1.000	1.000
Walidacyjny	Fake	0.999	1.000	0.999
(Acc: 1.000)	Real	1.000	0.999	1.000
Testowy	Fake	0.999	1.000	0.999
(Acc: 0.999)	Real	1.000	0.999	0.999

4.2.1. Macierze pomyłek

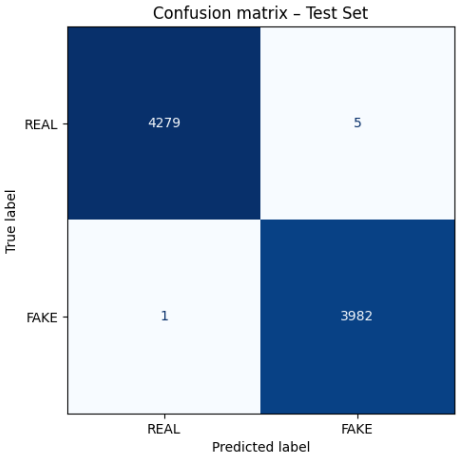
Analiza macierzy pomyłek (Rysunek 4.2) potwierdza niemal bezbłędne działanie modelu.



(a) Zbiór Treningowy



(b) Zbiór Walidacyjny



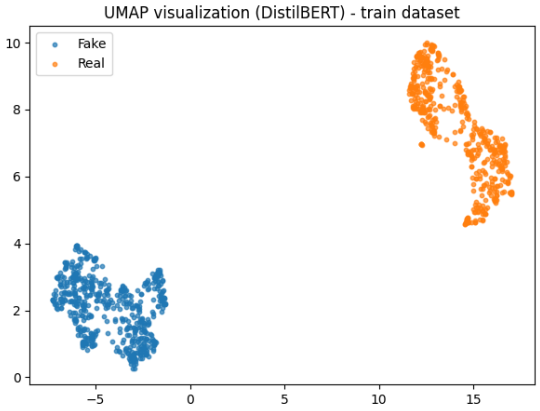
(c) Zbiór Testowy

Rysunek 4.2. Macierze pomyłek dla modelu DistilBERT.

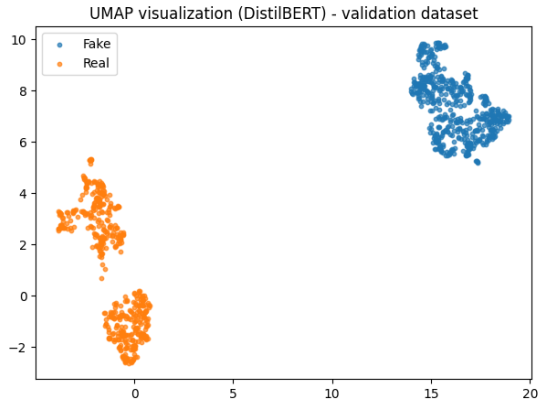
Na zbiorze testowym, liczącym ponad 8000 artykułów, model popełnił zaledwie **6 błędów** (1 fałszywy pozytyw i 5 fałszywych negatywów). Na zbiorze treningowym nie popełnił ani jednego błędu.

4.3. Wizualizacja przestrzeni cech (UMAP)

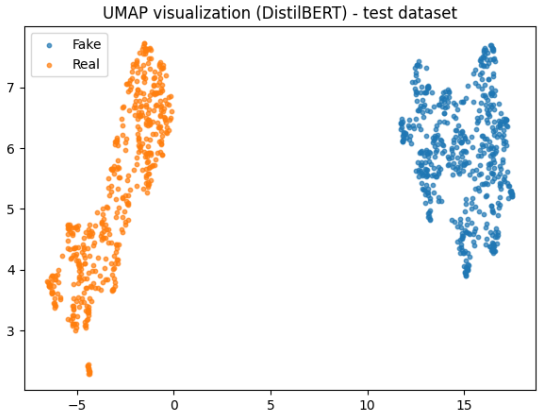
Dla modelu DistilBERT zastosowano algorytm UMAP do wizualizacji wektorów osadzeń (embeddings).



(a) Zbiór Treningowy



(b) Zbiór Walidacyjny



(c) Zbiór Testowy

Rysunek 4.3. Wizualizacja UMAP reprezentacji wygenerowanych przez DistilBERT.

Wnioski z wizualizacji (Rysunek 4.3):

- **Idealna separacja:** W przeciwieństwie do metody TF-IDF, tutaj chmury punktów dla klasy `Real` i `Fake` są całkowicie rozdzielone. Pomiędzy nimi znajduje się duża pusta przestrzeń.
- **Grupowanie tematyczne:** Wewnątrz klastrów widać mniejsze podgrupy. Sugeruje to, że model nie tylko rozróżnia prawdę od fałszu, ale także grupuje artykuły o podobnej tematyce lub stylu wewnątrz tych kategorii.

4.4. Wnioski dla modelu DistilBERT

1. **Skuteczność:** Model osiągnął skuteczność na poziomie 99.9%-100%. Jest to znacząca poprawa względem metody TF-IDF.
2. **Jakość uczenia:** Osiągnięcie 100% na zbiorze treningowym często sugeruje przeuczenie (*overfitting*). Jednak w tym przypadku wynik na zbiorze testowym jest niemal identyczny (99.9%), co oznacza, że model faktycznie "zrozumiał"różnice między klasami, a nie nauczył się ich na pamięć.
3. **Potencjał produkcyjny:** Ze względu na znikomą liczbę błędów, model ten jest idealnym kandydatem do wdrożenia w systemie produkcyjnym, gdzie priorytetem jest niezawodność.

5. Qwen-2.5 (Generatywny Model Językowy)

W trzecim podejściu wykorzystano model typu *Decoder-Only* – Qwen-2.5 o rozmiarze 1.5 miliarda parametrów. W przeciwieństwie do modelu DistilBERT, który jest klasyfikatorem, modele LLM są generatorami tekstu. Ich zadaniem jest przewidywanie kolejnego słowa (tokena).

Aby wykorzystać taki model do detekcji fake newsów, zdefiniowano problem jako zadanie generowania odpowiedzi w formacie czatu (Instruct), gdzie model na podstawie treści artykułu ma wygenerować słowo `"Real"` lub `"Fake"`.

Ze względu na duże wymagania sprzętowe modeli generatywnych, zastosowano techniki optymalizacyjne:

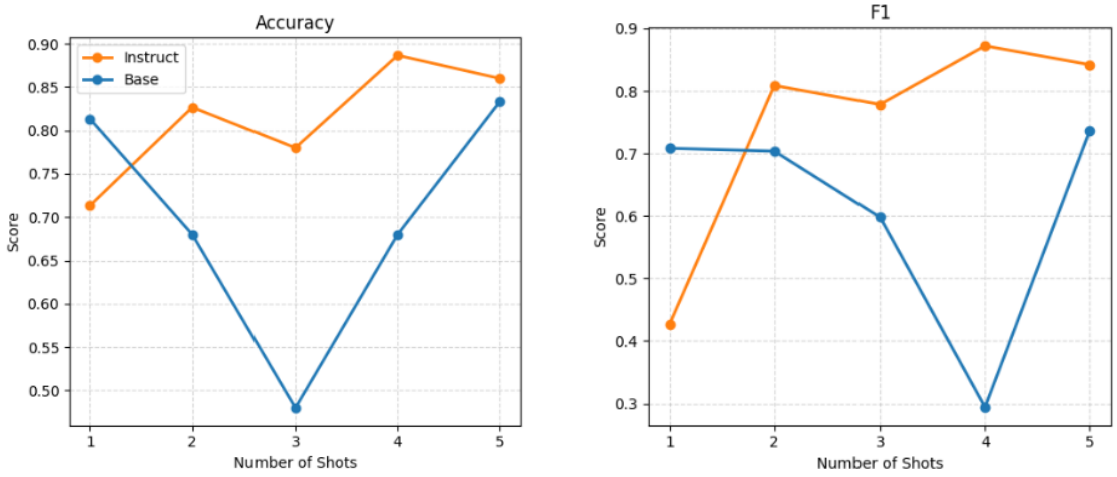
- **4-bit Quantization (bitsandbytes):** Kompresja wag modelu w celu zmniejszenia zużycia pamięci VRAM.
- **QLoRA (PEFT):** Metoda polegająca na zamrożeniu głównego modelu i trenowaniu jedynie niewielkich adapterów (macierzy niskiego rzędu).

5.1. Badanie wstępne: Few-Shot Prompting

Przed rozpoczęciem treningu przeprowadzono eksperyment mający na celu sprawdzenie zdolności modelu do uczenia się z kontekstu (*In-Context Learning*). Porównano dwa warianty modelu:

1. **Base:** Model surowy, zadanie sformułowane jako uzupełnianie tekstu.
2. **Instruct:** Model przystosowany do podążania za instrukcjami w formacie czatu.

Zbadano wpływ liczby podanych przykładów (k-shot, gdzie $k \in \{1,2,3,4,5\}$) na skuteczność modelu bez trenowania.



(a) Dokładność (Accuracy)

(b) Miara F1 (F1-Score)

Rysunek 5.1. Szczegółowa analiza wpływu liczby przykładów (shots) na kluczowe metryki. Widać przewagę modelu Instruct (linia pomarańczowa) przy 4 przykładach.

5.1.1. Wyniki analizy Few-Shot

Wykres 5.1 oraz Tabela 5.1 pokazują, że model w wersji **Instruct** radzi sobie znacznie lepiej niż wersja Base. Najlepsze wyniki osiągnięto przy podaniu w prompcie **4 przykładów** (4-shot learning). Ta konfiguracja została wybrana jako punkt wyjścia do dalszego treningu.

Tabela 5.1. Wyniki eksperymentu Few-Shot (wybrane konfiguracje).

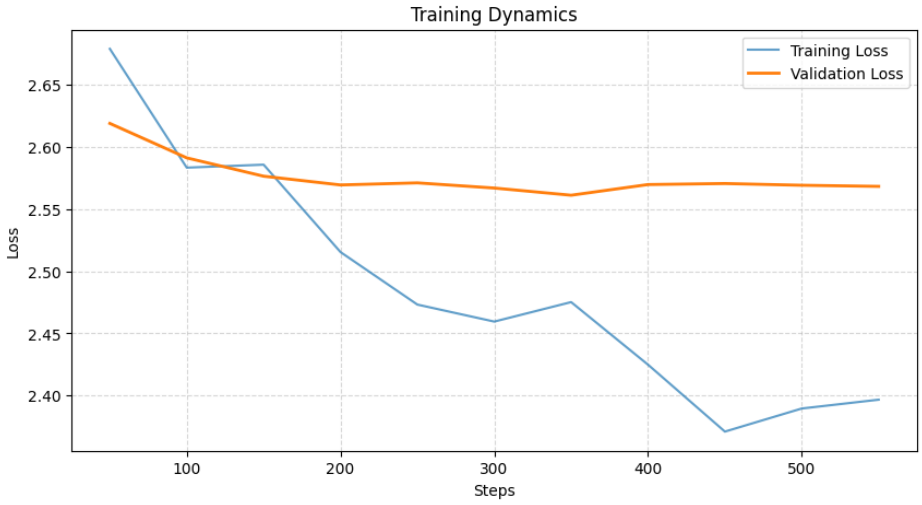
Wariant	Shots	Accuracy	F1-Score	Precision	Recall
Instruct	**4**	**0.887**	**0.872**	0.773	**1.000**
Instruct	5	0.860	0.842	0.747	0.966
Base	5	0.833	0.737	0.946	0.603
Instruct	0 (Zero-shot)	0.713	0.427	0.941	0.276
Base	3	0.480	0.598	0.426	1.000

5.2. Trening QLoRA i ewaluacja końcowa

Po wyborze najlepszego formatu promptów, model poddano procesowi dostrajania (*Supervised Fine-Tuning*) z użyciem techniki QLoRA.

Uwaga metodologiczna: Ze względu na bardzo długi czas inferencji modeli generatywnych, ewaluację końcową przeprowadzono na losowym podzbiorze 300 próbek z każdego zbioru danych.

5.2.1. Dynamika uczenia

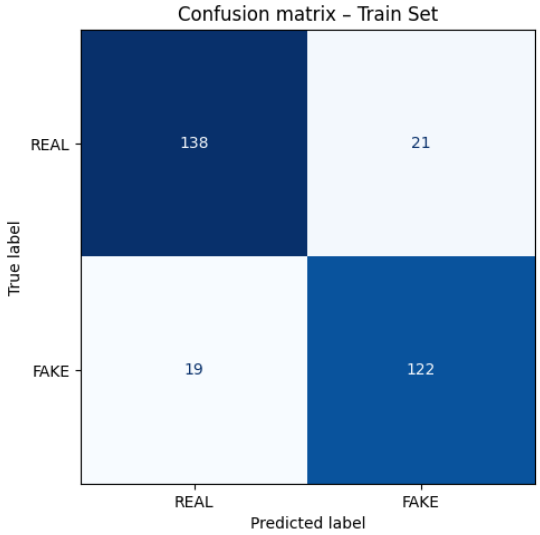


Rysunek 5.2. Przebieg funkcji straty podczas treningu QLoRA.

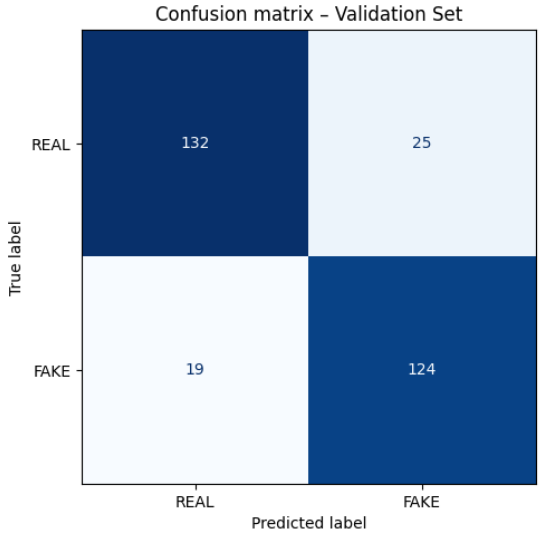
Wykres 5.2 pokazuje, że strata treningowa (*Training Loss*) systematycznie spadała, podczas gdy strata walidacyjna utrzymywała się na stabilnym poziomie, co sugeruje brak zjawiska przeuczenia przy zastosowaniu silnej regularyzacji, jaką zapewnia metoda LoRA.

5.2.2. Wyniki klasyfikacji i Macierze Pomyłek

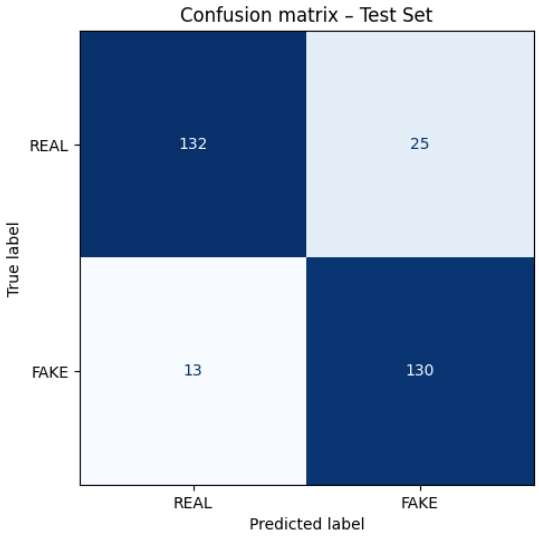
Ostateczna skuteczność modelu Qwen-2.5 po dostrojeniu oscyluje w granicach **87%**. Jest to wynik niższy niż w przypadku modelu DistilBERT, co może wynikać z faktu, że model 1.5B jest modelem stosunkowo małym jak na standardy LLM, a metoda 4-bitowej kwantyzacji wprowadza pewną utratę informacji.



(a) Zbiór Treningowy



(b) Zbiór Walidacyjny



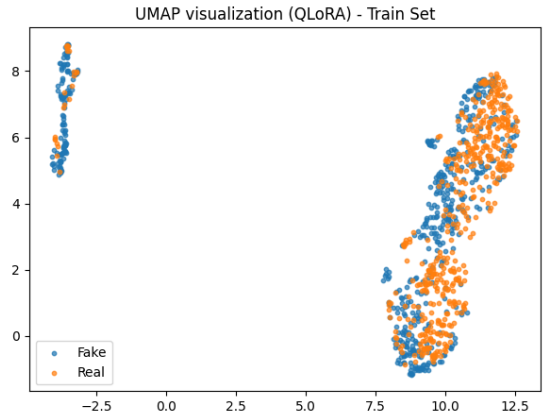
(c) Zbiór Testowy

Rysunek 5.3. Macierze pomyłek dla modelu Qwen-2.5 (na próbce n=300).

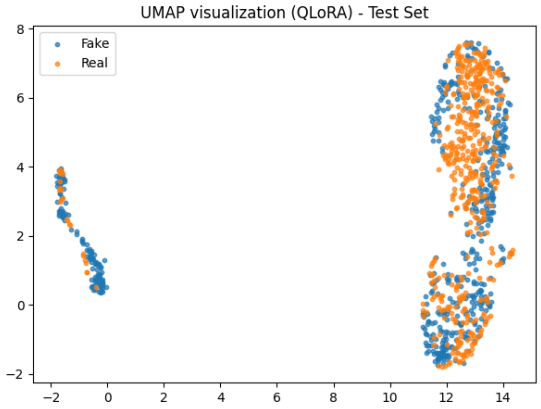
W analizie macierzy pomyłek (Rysunek 5.3) widać, że model zachowuje dobry balans między klasami, choć popełnia więcej błędów niż dedykowany model Encoder-Only.

5.3. Wizualizacja przestrzeni cech (LLM Embeddings)

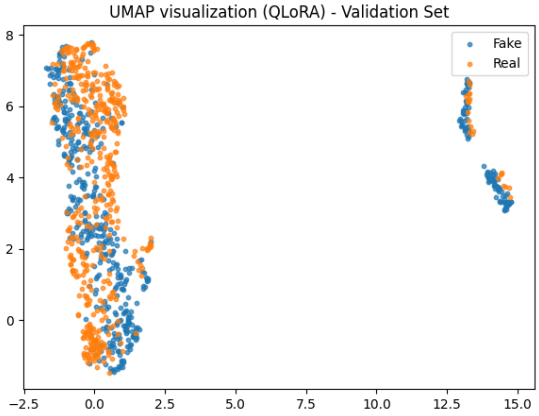
Analiza wykresów UMAP dla modelu QLoRA pokazuje, że mimo niższej dokładności numerycznej, model potrafi zbudować semantyczną reprezentację oddzielającą wiadomości prawdziwe od fałszywych.



(a) Zbiór Treningowy



(b) Zbiór Testowy



(c) Zbiór Walidacyjny

Rysunek 5.4. Wizualizacja UMAP dla reprezentacji modelu Qwen-2.5.

Separacja klastrów (Rysunek 5.4) jest wyraźna, choć granice między klasami są bardziej rozmyte niż w przypadku modelu DistilBERT, co koresponduje z wynikami tabelarycznymi.

6. Podsumowanie i wnioski końcowe

W ramach projektu przebadano trzy różne architektury pod kątem detekcji fake newsów. Celem było sprawdzenie, jak stopień skomplikowania modelu przekłada się na jego skuteczność.

6.1. Zestawienie wyników

Poniższa tabela przedstawia ostateczne porównanie skuteczności badanych metod na zbiorze testowym (dane, których model nigdy wcześniej nie widział).

Tabela 6.1. Końcowe zestawienie wyników na zbiorze testowym.

Metoda	Typ modelu	Dokładność (Accuracy)	F1-Score
TF-IDF + Regresja	Baseline (Statystyczny)	0.982	0.981
DistilBERT	**Encoder (Transformer)**	**0.999**	**0.999**
Qwen-2.5 + LoRA	Decoder (Transformer)	0.873	0.873

6.2. Wnioski

Na podstawie Tabeli 6.1 oraz przeprowadzonych analiz wizualnych można sformułować następujące wnioski:

1. **DistilBERT jest bezkonkurencyjny:** Model ten osiągnął niemal 100% skuteczności. Jako architektura dwukierunkowa (*Encoder-Only*), widzi on całe zdanie jednocześnie, co pozwala mu idealnie wyłapywać kontekst i subtelne cechy fałszywych wiadomości. Jest to najlepszy wybór do wdrożenia produkcyjnego.
2. **Proste metody są zaskakująco dobre:** Metoda TF-IDF osiągnęła wynik ponad 98%. Sugeruje to, że zbiór danych zawiera bardzo silne słowa kluczowe (np. specyficzne nazwiska, sensacyjne zwroty), które jednoznacznie wskazują na fake newsy.
3. **Duże modele (tylko-dekoder) nie zawsze są najlepsze:** Model Qwen-2.5, mimo że jest najnowocześniejszy i największy ("inteligentny"), poradził sobie najgorzej (87%). Wynika to z dwóch powodów:
    - Modele generatywne są stworzone do pisania tekstu, a nie do klasyfikacji (wyboru 0 lub 1).
    - Konieczność silnej kompresji (kwantyzacja 4-bitowa) w celu uruchomienia go na dostępnym sprzęcie mogła obniżyć jego jakość.

6.3. Podsumowanie

Podsumowując, do zadania binarnej klasyfikacji tekstu, wyspecjalizowane i lżejsze modele typu BERT są znacznie bardziej efektywne i ekonomiczne niż gigantyczne modele generatywne.