Zastosowanie grafowych sieci neuronowych w wykrywaniu mowy nienawiści i cyberprzemocy

Maciej Demucha, Kacper Wróblewski, Jakub Krupiński

Politechnika Wrocławska

1 Wstęp

1.1 Sieci neuronowe a przetwarzanie języka naturalnego

Ogrom informacji w połączeniu z potrzebą szybkiego dostępu do konkretnych, ale kompleksowych informacji skłonił do rozwoju badań nad przetwarzaniem języka naturalnego (NLP) i rozumieniem języka naturalnego (NLU).

Badania w tych obszarach przesunęły się w kierunku metod opartych na grafach. Mniejsza złożoność metod grafowych w porównaniu z metodami wektorowymi oferuje bardziej skompresowaną i wydajną reprezentację tekstu[9].

1.2 Grafowe sieci neuronowe

GNN to specjalnie zaprojektowane sieci neuronowe działające na danych o strukturze grafu. Celem GNN jest iteracyjna aktualizacja reprezentacji węzłów poprzez agregację reprezentacji sąsiadujących węzłów i ich własnej reprezentacji w poprzedniej iteracji. W literaturze zaproponowano wiele różnych sieci GNN, które można dalej podzielić na nadzorowane i nienadzorowane sieci GNN. Po nauczeniu się reprezentacji węzłów, podstawowym zadaniem na grafach jest klasyfikacja węzłów, która próbuje przydzielić węzły do jednej lub kilku zdefiniowanych klas [15].

Zaobserwowano wzrost zainteresowania rozwojem różnych typów sieci GNN i osiągnięto znaczny sukces w wielu zadaniach przetwarzania języka naturalnego od takich jak klasyfikacja zdań, do generacyjnych, takich jak tłumaczenie maszynowe lub generowanie pytań itp.[14] Pomimo tych sukcesów, głębokie uczenie się na grafach dla NLP nadal napotyka wiele wyzwań:

- automatyczna transformacja oryginalnych danych sekwencji tekstowej w dane o strukturze grafu,
- efektywne modelowanie danych, które obejmuje mapowanie między danymi wejściowymi opartymi na grafach i innymi wysoce ustrukturyzowanymi danymi wyjściowymi, takimi jak drzewa. Wiele zadań generacyjnych w NLP, takich jak SQL-to-Text, jest przykładem tego typu problemu.

Dane mogą być reprezentowane przez struktury grafowe w kilku obszarach zastosowań NLP. Najprostsze rodzaje struktur grafowych obejmują pojedyncze węzły i sekwencje. Jednak w wielu zastosowaniach informacje są organizowane w bardziej złożonych strukturach grafowych, takich jak drzewa, grafy acykliczne lub grafy cykliczne. W uczeniu maszynowym ustrukturyzowane dane są

często utożsamiane z celem (nadzorowanego lub nienadzorowanego) uczenia się na przykładach funkcji, która mapuje graf i jeden z jego węzłów na wektor liczb rzeczywistych[12].

Zaproponowano różne reprezentacje grafów do modelowania tekstu. Na podstawie różnych typów węzłów i krawędzi grafu większość istniejących prac można uogólnić do poniższych kategorii:

- Grafy tekstowe wykorzystują słowa, zdania, akapity lub dokumenty jako węzły i ustalają krawędzie na podstawie współwystępowania słów, lokalizacji lub podobieństwa tekstu. Grafy tekstowe można szybko utworzyć, ale nie mogą one charakteryzować składniowej lub semantycznej struktury zdań [3].
- Grafy składniowe podkreślają zależności składniowe między słowami w zdaniu. Takie strukturalne reprezentacje zdań uzyskuje się poprzez parsowanie, które konstruuje strukturę zdania zgodnie z formalną gramatyką.
- Grafy semantyczne mają na celu przedstawienie przekazywanego znaczenia. Są one pomocne w ujednoznacznieniu znaczenia zdania, gdy wiele interpretacji jest poprawnych.
- Grafy hybrydowe zawierają wiele typów węzłów i krawędzi, aby zintegrować informacje różnego typu [1]. W ten sposób różne atrybuty tekstu i relacje mogą być wspólnie wykorzystywane do zadań NLP [15].

1.3 Mowa nienawiści i cyberprzemoc

Mowa nienawiści stanowi wszechobecną i szkodliwą formę interakcji w sieci, często przejawiającą się w różnych wyzwiskach. W miarę jej rozprzestrzeniania, zaczyna stanowić znaczące zagrożenie społeczne, psychologiczne, a czasem nawet fizyczne dla osób i społeczności. Obecne podejścia NLP do zwalczania tego zjawiska opierają się na kolekcjonowaniu oznakowanych zbiorów danych z mediów społecznościowych używanych do trenowania modeli zdolnych do coraz skuteczniejszej detekcji toksyczności online [11].

Mowa nienawiści, znana również jako toksyczne zachowanie lub hejt, jest definiowana przez prawo Unii Europejskiej jako publiczne nawoływanie do przemocy lub nienawiści skierowane przeciwko grupom lub jednostkom na podstawie określonych cech, takich jak rasa, kolor skóry, religia, narodowość lub pochodzenie etniczne. W ostatniej dekadzie zainteresowanie akademickie mową nienawiści znacząco wzrosło. Badacze zajmujący się tym tematem opisują ją jako zestaw zachowań uznawanych za toksyczne w odniesieniu do stale renegocjowanych i ewoluujących norm społecznych [5].

Głoszenie jej jest coraz szerzej występującym przestępstwem, a konsekwencje są bagatelizowane [7]. Na ten stan wpływa kilka czynników. Z jednej strony w internecie, a w szczególności w mediach społecznościowych, ludzie częściej czują się anonimowi, w czym pomaga specyfika profili na mediach społecznościowych [4]. Z drugiej strony, ludzie są coraz bardziej skłonni do wyrażania swoich opinii w internecie [13], co również przyczynia się do propagacji mowy nienawiści. Tego rodzaju zachowania mogą być niezwykle szkodliwe dla społeczeństwa, rządy oraz

platformy społecznościowe mogą chcieć skorzystać z narzędzi do przetwarzania języka naturalnego, jakimi są GNN, do wykrywania i zapobiegania takim zjawiskom [6].

Wyzwania, przed którymi stoją automatyczne podejścia do wykrywania mowy nienawiści zostały zidentyfikowane i zbadane. Do trudności tych należą subtelności i manipulacje językowe [2], różne definicje mowy nienawiści oraz ograniczona dostępność danych do trenowania i testowania tych systemów. Ponadto wiele nowoczesnych podejść boryka się z problemem wyjaśnialności, to znaczy trudno jest zrozumieć, dlaczego systemy podejmują takie decyzje, a nie inne. W kontekście potrzeby skutecznej detekcji mowy nienawiści, zaproponowane zostało podejście oparte na klasyfikatorze SVM, które osiąga najlepsze wyniki, a jednocześnie jest prostsze i generuje bardziej zrozumiałe decyzje w porównaniu z metodami opartymi na sieciach neuronowych. Skuteczne narzędzia do wykrywania mowy nienawiści są niezbędne, aby wspierać zdrowe środowisko komunikacyjne w sieci i przeciwdziałać negatywnym skutkom tego zjawiska [8].

Jednym z głównych problemów badań nad cyberprzemocą jest brak danych, ponieważ badacze są tradycyjnie zmuszeni polegać na danych z ankiet, w których ofiary i sprawcy samodzielnie raportują swoje odczucia. W artykule *Machine Learning and Semantic Analysis of Ingame Chat for Cyber Bullying* [10] przedstawiono automatyczny system zbierania danych, który gromadzi dane z czatu w grze z jednej z popularniejszych gier wieloosobowych online: *World of Tanks*. Zebrane dane zostały połączone z innymi informacjami o graczach dostępnymi w serwisach internetowych *Wargaming* i zapisane w bazie danych. Prosta klasyfikacja SQL okazała się dość użyteczna w identyfikacji niektórych cech toksycznego czatu, takich jak używanie wulgarnych słów czy rasistowskich stwierdzeń. Następnie wyniki zostały przeanalizowane w celu uzyskania wglądu w cyberprzemoc w grze, wykazując, że możliwe jest znaczące ograniczenie cyberprzemocy w grze poprzez tymczasowe zablokowanie graczom możliwości komunikacji za pomocą czatu po ich śmierci w meczu. Wykazano również, że nowi gracze znacznie rzadziej angażują się w cyberprzemoc, co sugeruje, że może to być zachowanie nabyte od innych graczy [10].

To pokazuje, że toksyczne zachowania online można znacznie ograniczyć za pomocą narzędzi NLP, potrzebne są tylko zbiory danych, o które tak trudno. Firmy, które nie udostępniają informacji o komunikacji w swoich platformach (czy to w obszarze mediów społecznościowych, czy gier) możliwie pracują nad rozwiązaniem wykorzystującym przetwarzanie języka naturalnego oraz sieci neuronowe, podczas gdy drugie otwarcie udostępniają te dane, zachęcając społeczność do tworzenia narzędzi pomagających w filtracji mowy nienawiści i cyberprzemocy [10].

2 Przegląd literaturowy

2.1 Detection of hateful twitter users with graph convolutional network model [1]

Badanie to koncentruje się na wykrywaniu mowy nienawiści w mediach społecznościowych z zastosowaniem modelu grafowej sieci konwolucyjnej (GCN). Celem badania jest poprawa wykrywania mowy nienawiści przez uwzględnienie nie tylko treści, ale także struktury sieciowej użytkowników Twittera (takich jak powiązania z innymi użytkownikami albo ich retweety). W przeciwieństwie do tradycyjnych metod opierających się wyłącznie na analizie treści, podejście oparte o GCN wykorzystuje zarówno dane tekstowe jak i strukturalne sieci Twittera. W badaniu tym wykorzystano publicznie dostępny zbiór danych z Twittera obejmujący 100 368 użytkowników i ich aktywność na platformie.

2.2 Graph-Based Methods to Detect Hate Speech Diffusion on Twitter [2]

Autorzy tego artykułu koncentrują się na opracowaniu uzupełniających metod wykrywaniu mowy nienawiści z wykorzystaniem modeli grafowych śledzących dyfuzję takich treści w sieci, które stanowiłyby wsparcie dla istniejących rozwiązań tekstowych. Wykorzystany jest zbiór danych zawierający około 10 000 tweetów, które zostały ręcznie oznaczone jako nienawistne lub normalne. Badanie to pomogło również podkreślić słabości tradycyjnych modeli tekstowych przy rozwiązywaniu tego typu problemów, jako że mogą one być bardziej podatne na ataki oraz manipulacje językowe.

2.3 Hate Speech and Offensive Content Identification with Graph Convolutional Networks [3]

W tym artykule przedstawiono metodę wykrywania mowy nienawiści na platformach społecznościowych przy użyciu grafowych sieci konwolucyjnych (GCN), jako alternatywę dla tradycyjnych modeli (np. LSTM czy BERT). Badanie to zostało przeprowadzone w ramach wyzwania HASOC 2021, gdzie zaproponowane tu rozwiązanie zajęło 2 miejsce. Opracowana przez autorów sieć przedstawia dokumenty jako graf, w którym węzły odpowiadają tweetom oraz unikalnym słowom, krawędzie tworzone są na podstawie współwystępowania słów oraz ich ważności w dokumencie, a wagi obliczane są przy użyciu wskaźników: TF-IDF oraz PMI. Architektura pełnego modelu składa się z dwuwarstwowej sieci GCN i klasyfikatora softmax.

3 Projekt i implementacja

3.1 Pytania badawcze

1. Jak prezentuje się wydajność grafowych sieci neuronowych w porównaniu do klasycznych sieci neuronowych?
2. Jakie metody konstrukcji grafu (np. reprezentowanie węzłów jako użytkowników, postów lub słów) mają największy wpływ na wyniki modelu w detekcji mowy nienawiści?
3. Czy uwzględnienie kontekstu sieciowego (np. zależności między postami, interakcjami użytkowników) poprawia skuteczność modeli opartych na grafowych sieciach neuronowych w porównaniu z klasycznymi estymatorami probabilistycznymi na przykładzie lasu losowego?

3.2 Plan eksperymentów

Metody konstrukcji grafów Pierwszym eksperymentem jest sprawdzenie wyników klasyfikacji dla różnych metod konstrukcji grafów. Pierwszą metodą jest konstrukcja grafów na podstawie pojedynczych zdań, gdzie węzły będą reprezentować słowa w zdaniu, które będą połączone krawędziami na podstawie tego, które słowa są ze sobą sąsiednie w konkretnym zdaniu. Drugą metodą jest konstrukcja grafów, gdzie węzły będą reprezentować poszczególne wypowiedzi dotyczące tego samego targetu.

Kontekst sieciowy Celem drugiego eksperymentu jest porównanie działania tradycyjnego modelu BERT ignorującego kontekst sieciowy z działaniem sieci grafowych. Embeddingi zostaną wykorzystane w klasyfikacji za pomocą Multi-Layer Perceptron (MLP).

Metryka: Metryką użytą w eksperymencie będzie *accuracy* liczone poprzez porównanie liczby poprawnie przewidzianych próbek do całości datasetu branego pod uwagę w eksperymencie (dla sieci). W przypadku lasu losowego zastosowaliśmy *balanced_ accuracy.*

Zbiór danych Do badań zostanie wykorzystany zbiór danych "A Benchmark Dataset for Learning to Intervene in Online Hate Speech". Zawiera zbiór wypowiedzi oraz komentarzy zebranych z Reddita i Gab. Każdy element składa się z jednego posta w języku angielskim z własnym ID i przypiętych komentarzy również posiadających swoje ID. Każdy post i komentarz jest oznaczony etykietą 0 lub 1 informujący czy jest to mowa nienawiści. W zbiorze zawarte są również wygenerowane odpowiedzi na dany element mowy nienawiści.

Pole	Opis
`id`	ID postu lub komentarza w konwersacji.
`text`	Tekst postu lub komentarza.
`hate_speech_idx`	Lista etykiet 0 lub 1 kolejnością odpowiadająca postom oraz przypiętym komentarzom.
`response`	Tekst wygenerowanej odpowiedzi na mowę nienawiści.

Table 1. Zbiór danych

Proces uczenia Posty z reddita i gaba zostały przeformatowane na etapie preprocessingu na format `{'text':'label'}` oraz podane modelowi BERT (`all-Mi--niLM-L6-v2`) w celu wytworzenia osadzeń. Następnie *embeddingi* zostały podane do odpowiedniej metody tworzenia grafów, opisanych w powyżej. W przypadku metod porównawczych (MLP, las losowy), chcąc wykorzystać obliczone już osadzenia, podawaliśmy je bezpośrednio na wejściu metod (w przypadku MLP w porcjach o rozmiarze 16).

Dla metod grafowych naszym optymalizatorem jest *Adam*, a krytrium *CrossEntropyLoss*, natomiast w przypadku klasyfikatora MLP jako kryterium użyliśmy *BinaryCrossEntropyLoss*. Implementacja tych klasy pochodzi z biblioteki *PyTorch*.

4 Implementacja

4.1 Klasyfikacja węzłów

Model grafowej sieci konwolucyjnej (GCN) przedstawiony na listingu 1.1 jest wykorzystywany do klasyfikacji węzłów w grafach. Sieci GCN umożliwiają propagację informacji między połączonymi węzłami, co pozwala na efektywne uwzględnienie relacji w grafie.

- Pierwsza warstwa GCN transformuje cechy węzłów do przestrzeni ukrytej, uwzględniając ich lokalny kontekst w grafie.
- Druga warstwa agreguje informacje i dokonuje predykcji klas.
- Warstwa dropout redukuje przetrenowanie, zwiększając zdolność uogólniania modelu.

Listing 1.1. Architektura sieci GCN do klasyfikacji węzłów

```
class GCN(torch.nn.Module):
    def __init__(self, hidden_channels):
        super().__init__()
        torch.manual_seed(1234567)
        self.conv1 = GCNConv(input_dim, hidden_channels)
        self.conv2 = GCNConv(hidden_channels, 2)
```

```
    def forward(self, x, edge_index):
        x = self.conv1(x, edge_index)
        x = x.relu()
        x = F.dropout(x, p=0.5, training=self.training)
        x = self.conv2(x, edge_index)
        return x
```

4.2 Klasyfikacja grafów

Listing 1.2 przedstawia model GCN jest wykorzystany przez nas do klasyfikacji całych grafów. W porównaniu do architektury GCN dla klasyfikacji węzłów, wprowadza on:

- Dodatkową warstwę GCN (`conv3`), co pozwala na głębszą propagację informacji w grafie.
- Mechanizm *global mean pooling*, który agreguje informacje z całego grafu do jednej reprezentacji cech.
- Warstwę w pełni połączoną (`self.lin`), dokonującą klasyfikacji na poziomie całych grafów.

Listing 1.2. Architektura sieci GCN do klasyfikacji grafów

```
class GCN(torch.nn.Module):
    def __init__(self, hidden_channels):
        super(GCN, self).__init__()
        torch.manual_seed(12345)
        self.conv1 = GCNConv(input_dim, hidden_channels)
        self.conv2 = GCNConv(hidden_channels, hidden_channels)
        self.conv3 = GCNConv(hidden_channels, hidden_channels)
        self.lin = Linear(hidden_channels, num_classes)

    def forward(self, x, edge_index, batch):
        x = self.conv1(x, edge_index)
        x = x.relu()
        x = self.conv2(x, edge_index)
        x = x.relu()
        x = self.conv3(x, edge_index)
        x = global_mean_pool(x, batch)
        x = F.dropout(x, p=0.5, training=self.training)
        x = self.lin(x)
        return x
```

4.3 Implementacja klasy *MLPClassifier*

W celu porównania skuteczności metody zaimplementowaliśmy niegrafową metodę w postaci klasyfikatora MLP. Listing 1.3 przedstawia kod klasy *MLPClassifier*.

Opisuje on klasę MLPClassifier, która dziedziczy po *nn.Module* z biblioteki *PyTorch*. Klasa ta reprezentuje prostą sieć neuronową typu Multi-Layer Perceptron (MLP) używaną do klasyfikacji binarnej tekstu. Posiada ona dwie warstwy liniowe w pełni połączone ze sobą, a pomiędzy nimi dwie funkcje aktywacji, *ReLU* oraz *Sigmoid*.

Listing 1.3. Architektura klasy MLPClassifier

```
class MLPClassifier(nn.Module):
    def __init__(self, input_dim, hidden_dim=128, output_dim=1):
        super(MLPClassifier, self).__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_dim, output_dim)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.fc1(x)
        x = self.relu(x)
        x = self.fc2(x)
        return self.sigmoid(x)
```

4.4 Las losowy

Inną metodą zaimplementowaną jako porównawcza, był las losowy. Listing 1.4 przedstawia implementację pętli uczenia maszynowego.

Listing 1.4. Implementacja klasyfikatora Random Forest

```
rskf = RepeatedStratifiedKFold(n_splits=5, n_repeats=2)
rf = RandomForestClassifier(n_estimators=50, max_depth=10)
cv_scores = []

for i, (train_idx, test_idx) in enumerate(rskf.split(X_train, y_train)):
    X_train_fold, X_test_fold = X_train[train_idx], X_train[test_idx]
    y_train_fold, y_test_fold = y_train[train_idx], y_train[test_idx]

    model = clone(rf)
    model.fit(X_train_fold, y_train_fold)
    y_pred = model.predict(X_test_fold)
    accuracy = balanced_accuracy_score(y_test_fold, y_pred)
    cv_scores.append(accuracy)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

5 Rezultaty eksperymentu

Eksperymenty przeprowadzono za pomocą Grafowej Konwolucyjnej Sieci Neuronowej, wielowarstwowego perceptronu oraz lasu losowego. W przypadku grafów ze słów dokonano zadania klasyfikacji grafów, gdzie każdy graf posiadał etykietę klasyfikującą go jako mowa nienawiści lub nie. Na grafach utworzonych z postów oraz komentarzy dokonano zadania klasyfikacji wierzchołków jednego grafu, który składał się ze wszystkich utworzonych grafów w ramach eksperymentu. Dla MLP i lasu losowego użyto samych embeddingów jako inputu.

5.1 Konstrukcja grafów ze słów

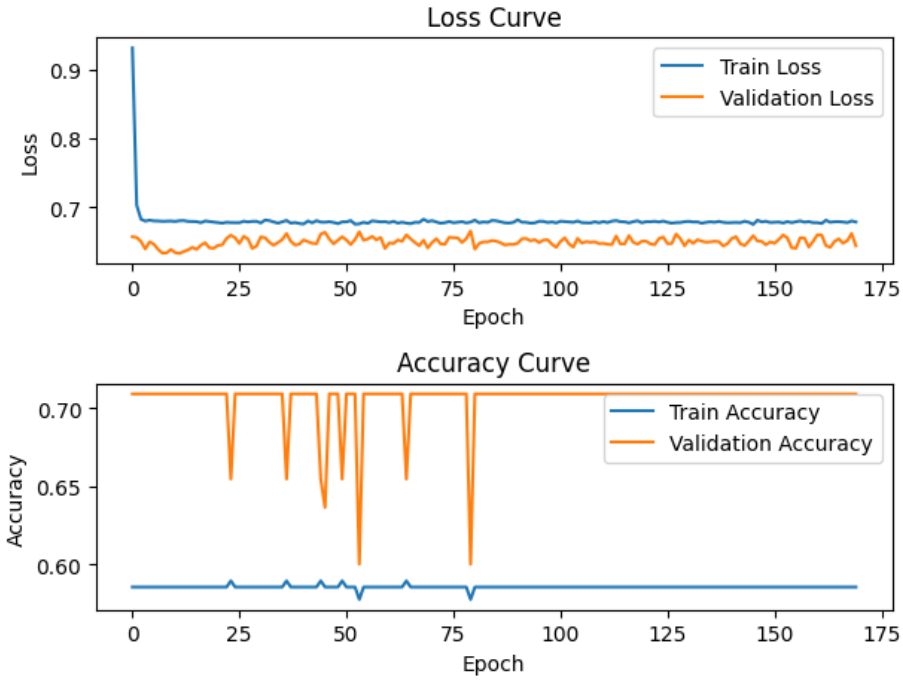

Fig. 1. Krzywa uczenia dla zbioru z serwisu Gab – Test Accuracy: 0.7091

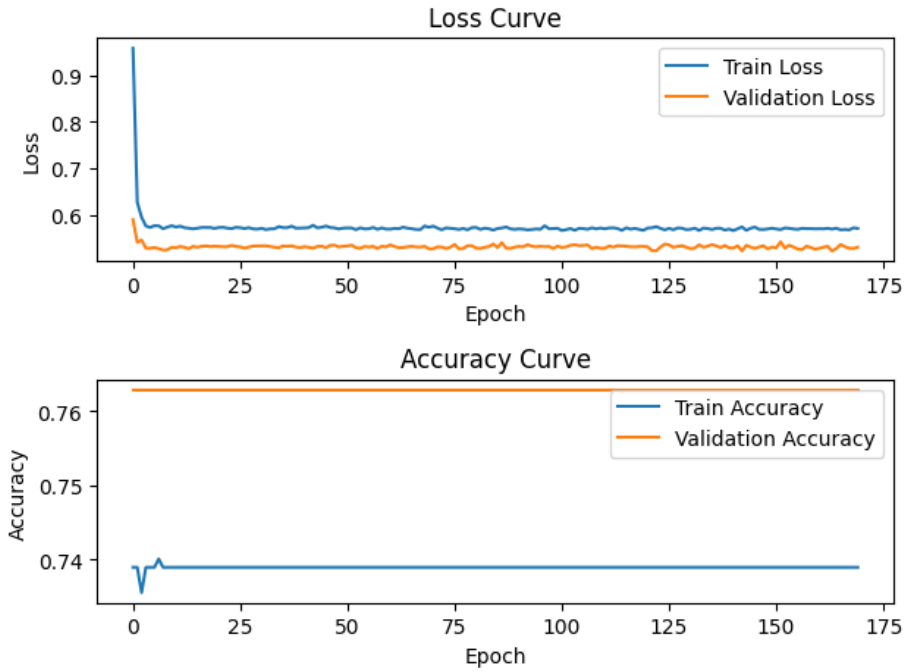

Fig. 2. Krzywa uczenia dla zbioru z serwisu Reddit – Test Accuracy: 0.7629

5.2 Konstrukcja grafów z postów oraz komentarzy

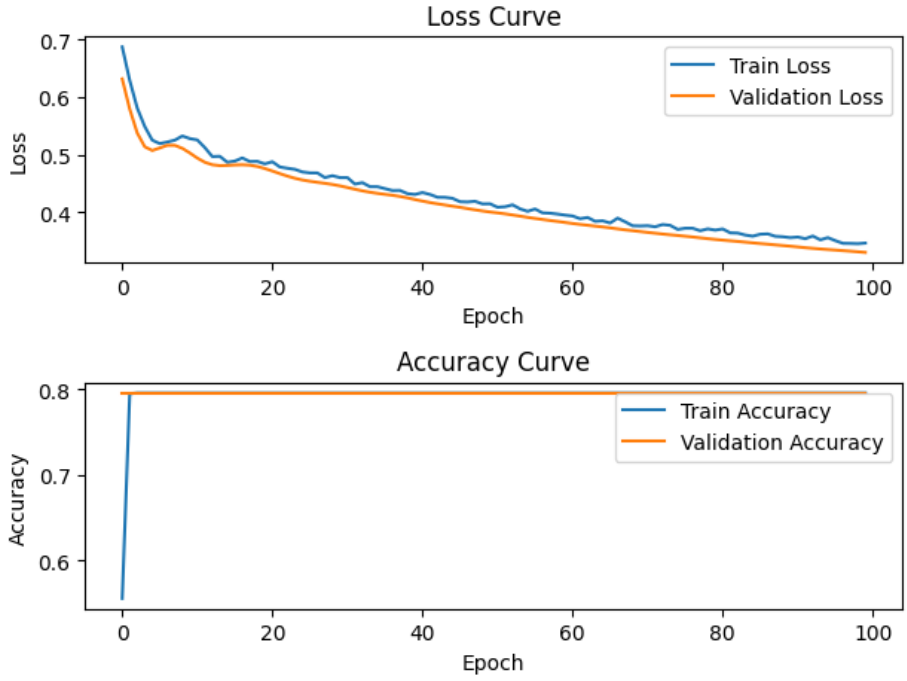

Fig. 3. Krzywa uczenia dla zbioru z serwisu Gab – Test Accuracy 0.7960

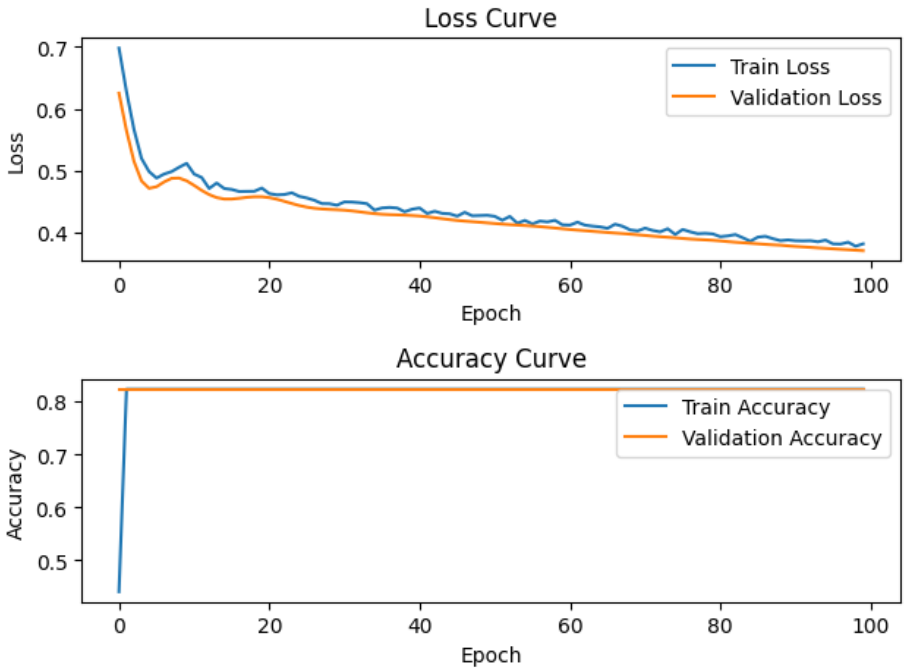

Fig. 4. Krzywa uczenia dla zbioru z serwisu Reddit – Test Accuracy: 0.8227

5.3 Porównanie z metodą niegrafową

Wielowarstwowy perceptron: Wykres 5. przedstawia krzywą uczenia klasyfikatora MLP na tym samym zbiorze danych co sieć grafowa.

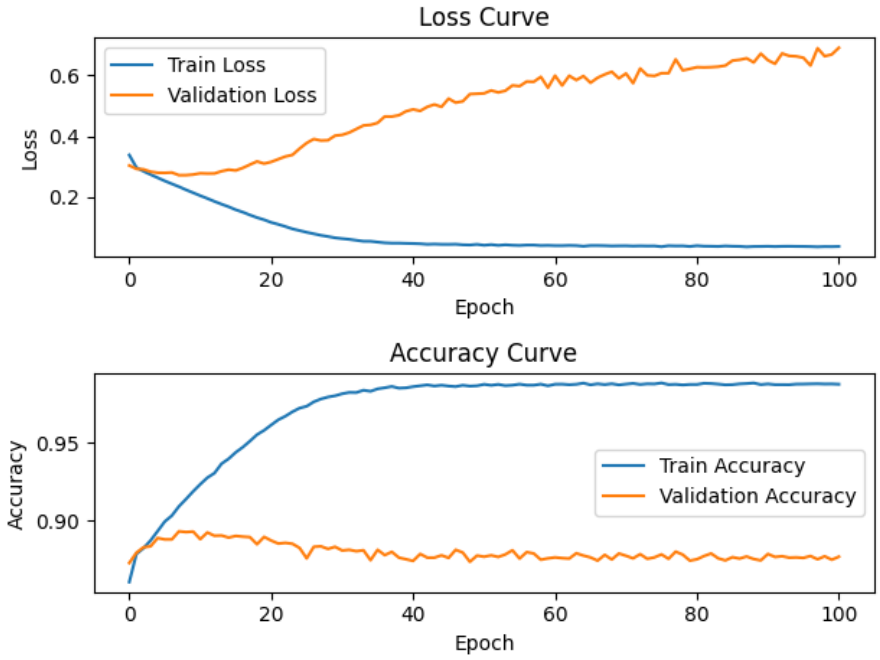

Fig. 5. Krzywe uczenia dla klasyfikatora *Multi-layer Perceptron* – Test Accuracy: 0.8770

Las losowy: Wykres 6. przedstawia krzywe uczenia dla drugiej niegrafowej metody.

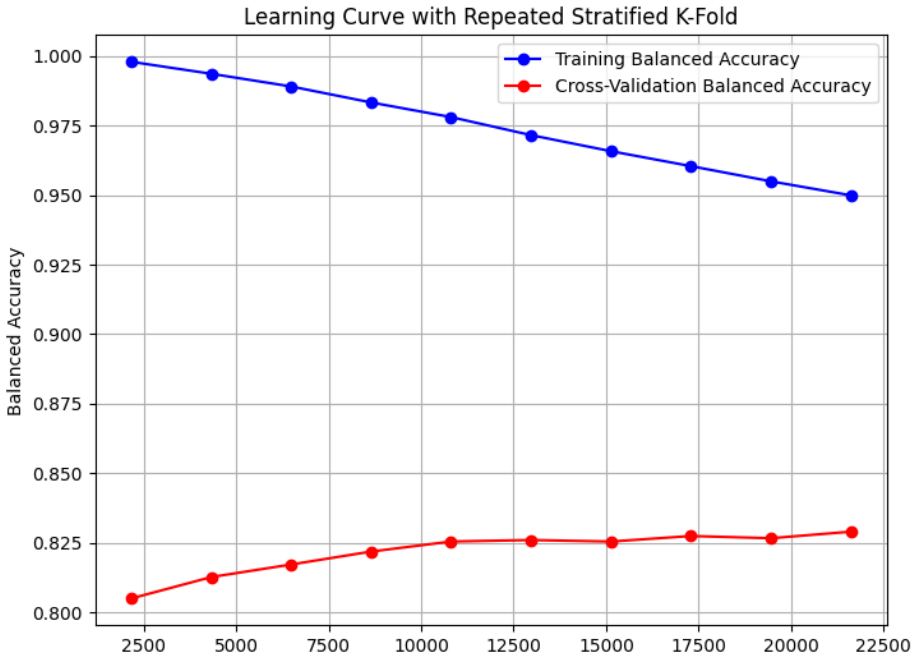

Fig. 6. Krzywe uczenia dla klasyfikatora Random Forest

6 Podsumowanie i wnioski

Wykorzystanie Grafowych Sieci Neuronowych nie przyniosło zadowalających rezultatów. Krzywa `accuracy` w poszczególnych epokach przyjmowała zawsze podobne wartości i nie ulegała istotnej poprawie, a efektywność tych modeli nie różniła się znacząco od zaimplementowanego przez nas estymatora *Random Forest*. Powodem tego problemu może być brak kontekstu sieciowego w zbiorze danych. Posiadając wyłącznie informację na temat identyfikatora postu, jego komentarzy, ich treści oraz etykiety nie udało się znaleźć sposobu budowy grafów, który pozwoliłby na poprawę rezultatów.

W przypadku klasyfikatora MLP szybko dochodzi do zjawiska przeuczenia, co można zaobserwować na krzywej 5, gdzie parametr `Train Loss` szybko zmniejsza swoją wartość podczas gdy `Validation` Loss rośnie. Mimo że estymator MLP osiągnął lepsze rezultaty, to nie są to wyniki wiarygodne, ponieważ stracił on możliwość generalizacji, przez co wskazane jest prowadzenie dalszych badań nad jego konfiguracją.

Przeprowadzone przez nas badania wskazują wyraźnie na istotność kontekstu społecznego przy implementacji grafowych sieci neuronowych. Metody te bez wątpienia mogą posiadać wiele zastosowań, istotne jest jednak, by zbiór danych pozwolił na pełne wykorzystanie potencjału tych rozwiązań. Znalezienie ogólnodostępnych zbiorów danych, które pozwalają na dogłębne badanie tego aspektu stanowi niemałe wyzwanie, jako że informacje zawarte w zbiorze *A Benchmark Dataset for Learning to Intervene in Online Hate Speech* okazały się niewystarczające do zagłębienia się w ten temat dla zadania detekcji mowy nienawiści.

References

1. Anil Utku, U.C., Aslan, S.: Detection of hateful twitter users with graph convolutional network model. Earth Science Informatics (2023). https://doi.org/https://doi.org/10.1007/s12145-023-00940-w
2. Beatty, M.: Graph-based methods to detect hate speech diffusion on twitter. In: 2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). pp. 502–506 (2020). https://doi.org/10.1109/ASONAM49781.2020.9381473
3. Bölücü, N., Canbay, P.: Hate speech and offensive content identification with graph convolutional networks. In: Mehta, P., Mandl, T., Majumder, P., Mitra, M. (eds.) Working Notes of FIRE 2021 - Forum for Information Retrieval Evaluation, Gandhinagar, India, December 13-17, 2021. CEUR Workshop Proceedings, vol. 3159, pp. 44–51. CEUR-WS.org (2021), https://ceur-ws.org/Vol-3159/T1-4.pdf
4. Burnap, P., Williams, M.L.: Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. Policy Internet **7**(2), 223–242 (2015)
5. Costa, S., Mendes da Silva, B., Tavares, M.: Online Hate Speech in Video Games Communities: A Counter Project. Counterspeech, Routledge, 1st edition edn. (2023), open Access funded by Knowledge Unlatched GmbH
6. Fortuna, P., Nunes, S.: A survey on automatic detection of hate speech in text. ACM Computing Surveys **51**(4), 1–30 (2018). https://doi.org/10.1145/3232676
7. Kwak, H., Blackburn, J.: Linguistic analysis of toxic behavior in an online video game. Springer International Publishing Switzerland pp. 209–217 (2015). https://doi.org/10.1007/978-3-319-15168-7_26
8. MacAvaney, S., Yao, H.R., Yang, E., Russell, K., Goharian, N., Frieder, O.: Hate speech detection: Challenges and solutions. PLoS One **14**(8), e0221152 (Aug 2019). https://doi.org/10.1371/journal.pone.0221152
9. Mills, M.T., Bourbakis, N.G.: Graph-based methods for natural language processing and understanding—a survey and analysis. IEEE Transactions on Systems, Man, and Cybernetics: Systems **44**(1), 59–71 (2014). https://doi.org/10.1109/tsmcc.2012.2227472
10. Murnion, S., Buchanan, W.J., Smales, A., Russell, G.: Machine learning and semantic analysis of in-game chat for cyberbullying. CoRR **abs/1907.10855** (2019), http://arxiv.org/abs/1907.10855
11. Piot, P., Martín-Rodilla, P., Parapar, J.: Metahate: A dataset for unifying efforts on hate speech detection. Proceedings of the International AAAI Conference on Web and Social Media **18**(1), 2025–2039 (May 2024). https://doi.org/10.1609/icwsm.v18i1.31445, https://ojs.aaai.org/index.php/ICWSM/article/view/31445
12. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Transactions on Neural Networks **20**(1), 61–80 (2009). https://doi.org/10.1109/tnn.2008.2005605
13. Wendling, M.: 2015: The year that angry won the internet (2015), http://www.bbc.com/news/blogs-trending-35111707, stan strony z dnia 5.11.2024
14. Wu, L., Chen, Y., Ji, H., Li, Y.: Deep learning on graphs for natural language processing **20**(1), 61–80 (2021). https://doi.org/10.1109/tnn.2008.2005605
15. Wu, L., Cui, P., Pei, J., Zao, L.: Graph neural networks: Foundation, frontiers and applications (2022)