Filip Wójcik

Grafowe sieci neuronowe

Teoria i praktyka

Wydawnictwo Naukowe Helion

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Grafika na okładce została wykorzystana za zgodą AdobeStock.com.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/siegra
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.
Kolorowe wersje wybranych rysunków dostępne są pod adresem:
https://ftp.helion.pl/przyklady/siegra.zip.

ISBN: 978-83-289-3390-3

Printed in Poland.

Spis treści

ROZDZIAŁ 6.

Przykłady zastosowań grafowych sieci neuronowych

Niniejszy rozdział zostanie poświęcony omówieniu praktycznych zastosowań grafowych sieci neuronowych w kontekście rozwiązywania wybranych problemów biznesowych. W każdej kategorii zadań, takich jak klasyfikacja wierzchołków, predykcja połączeń czy klasyfikacja całych grafów, przedstawiony zostanie konkretny problem wraz z odpowiadającym mu zbiorem danych. Dodatkowo zaprezentowane zostanie przykładowe rozwiązanie wraz z jego implementacją. Taki układ treści ma na celu ułatwienie utrwalenia wiedzy teoretycznej oraz zrozumienia architektury GNN, omówionych w poprzednich rozdziałach.

Ważne: różnice w wynikach

Biblioteki wykorzystane w przykładzie i dołączonych materiałach korzystają z obliczeń opartych na kartach graficznych (CUDA).

Pomimo zastosowania mechanizmów zapewniania deterministyczności procesu szkolenia (m.in. ustawianie ziarna losowego za pomocą biblioteki PyG, deterministyczne uczenie w PyTorch Lightning itp.) nie ma gwarancji, że uzyskane wyniki będą za każdym razem identyczne. Różnice mogą wynikać m.in. **z wersji używanego sterownika CUDA oraz innych systemowych sterowników karty** graficznej, kompilacji bibliotek dla konkretnego systemu operacyjnego (Linux, Windows, macOS).

6.1. Klasyfikacja wierzchołków

6.1.1. Postać formalna

Klasyfikacja wierzchołków w grafach to jedno z głównych zagadnień w analizie danych strukturalnych, które znajduje zastosowanie w wielu dziedzinach, takich jak sieci społecznościowe, biologia obliczeniowa czy analiza finansowa.

Formalnie zadanie to polega na przypisaniu każdemu wierzchołkowi $v_i \in V$ z grafu $G = (V, E)$ etykiety $y_i \in \mathcal{Y}$ spośród zbioru możliwych klas C.

Aby wykonać zadanie klasyfikacji wierzchołków, niezbędne są, oprócz grafu, następujące elementy:

- zbiór wierzchołków $V = \{v_1, v_2, \dots, v_N\}$, gdzie $N = |V|$;
- zbiór krawędzi $E \subseteq V \times V$, które reprezentują relacje między wierzchołkami;

- macierz cech $\boldsymbol{X} \in \mathbb{R}^{N \times d}$, gdzie każdy wiersz $x_i \in \mathbb{R}^d$ zawiera wektor cech opisujących wierzchołek v_i;
- zbiór etykiet $Y = \{y_1, y_2, \ldots, y_N\}$, gdzie $y_i \in Y = \{1,2, \ldots, C\}$.

Celem klasyfikacji jest nauczenie modelu:

$$\text{GNN}: (G, \boldsymbol{X}) \rightarrow Y, \tag{45}$$

który przewiduje etykiety dla wszystkich lub wybranych wierzchołków. W przypadku częściowo nadzorowanego uczenia (ang. *semi-supervised learning*) etykiety są znane jedynie dla podzbioru wierzchołków $\mathrm{V_L} \subset \mathrm{V}$, a zadaniem modelu jest przewidzenie etykiet dla pozostałych $V_U = V \setminus V_L$. Problem ten można formalnie zapisać jako minimalizację funkcji kosztu:

$$\mathcal{L} = \frac{1}{|V_L|} \sum_{v_i \in V_L} loss(\text{GNN}(v_i; G, \boldsymbol{X}), y_i)\,, \tag{46}$$

gdzie:

- $\mathcal{L}(\cdot,\cdot)$ to globalna funkcja kosztu (np. entropia krzyżowa);
- $loss(.)$ to funkcja kosztu dla pojedynczego przykładu uczącego;
- $\text{GNN}(v_i;\ G, \boldsymbol{X})$ to model splotu grafowego dokonujący predykcji klasy dla wierzchołka v_i,
- y_i to rzeczywista etykieta przypisana do tego wierzchołka.

W przeciwieństwie do klasycznych modeli uczenia maszynowego, grafowe sieci neuronowe stanowią efektywne podejście do tego problemu, gdyż wykorzystują zarówno lokalne informacje o sąsiedztwie każdego wierzchołka (struktura grafu), jak i jego cechy.

6.1.2. Znaczenie problemu i zastosowania

Klasyfikacja wierzchołków ma szerokie zastosowania praktyczne.

Jednym z najważniejszych jest wykrywanie botów w sieciach społecznościowych. W tym przypadku modele analizują grafy znajomości oraz wzorce interakcji, aby identyfikować konta automatyczne lub prowadzące złośliwą działalność. Techniki te są skuteczne nawet przy ograniczonych danych wejściowych i pozwalają na rozpoznawanie botów o różnych strategiach kamuflażu (Kalameyets, 2021).

W sektorze finansowym klasyfikacja wierzchołków wspiera wykrywanie oszustw poprzez analizę relacji między transakcjami, punktami sprzedaży i użytkownikami kart kredytowych. Sieci GNN umożliwiają identyfikację złożonych wzorców oszustw znacznie skuteczniej niż tradycyjne metody (Johannessen, Jullum, 2023; Pereira, Murai, 2021).

W biologii obliczeniowej klasyfikacja genów na podstawie sieci molekularnych pozwala przewidywać funkcje nieznanych genów oraz ich związki z chorobami. Metody te wykorzystują podejście „wina przez skojarzenie" (ang. *guilt-by-association*), zakładające, że geny silnie połączone w sieci mają podobne funkcje (R. Liu, Mancuso, Yannakopoulos et al., 2020; Mancuso, Johnson, Liu et al., 2024).

W bieżącym podrozdziale zajmiemy się klasyfikacją tematyczną stron internetowych na Facebooku na podstawie ich sąsiedztwa (wzajemnych linków) oraz zawartości. Analiza ta wspiera personalizację treści i optymalizację kampanii reklamowych. Przykładowo klasyfikacja interakcji na stronach marek pozwala zrozumieć wzorce zaangażowania użytkowników oraz segmentować strony według ich popularności i aktywności fanów (Chiu, 2021).

6.1.3. Klasyfikacja tematyczna stron na Facebooku

Eksperyment 11. Klasyfikacja tematyczna stron na Facebooku

Implementacja eksperymentu znajduje się w notatniku *rozdzial_6_zastosowania_sieci_gnn/01_facebook_klasyfikacja_wierzcholkow.ipynb*.

Ze względu na stopień skomplikowania i objętość omawianego przykładu w części książkowej zostaną omówione założenia, parametry i wyniki. Detale implementacyjne znajdują się w dołączonych materiałach (notatniku).

W tym podrozdziale przedstawiamy szczegółową analizę studium przypadku opartego na zbiorze danych *Facebook Large Page-Page Network*, który jest częścią repozytorium **Stanford SNAP**[1]. Zbiór ten ilustruje zastosowanie grafowych sieci neuronowych do wieloklasowej klasyfikacji wierzchołków i koncentruje się na rzeczywistych danych z sieci społecznościowej (Rozemberczki, Allen, Sarkar, 2021).

Opisywany zbiór danych to graf reprezentujący zweryfikowane strony na Facebooku, gdzie:

- **Wierzchołki** odpowiadają oficjalnym stronom.
- **Krawędzie** reprezentują wzajemne „polubienia" między nimi.

Każdy wierzchołek należy do jednej z czterech kategorii zdefiniowanych przez Facebook:

1. Firmy (~30% wartości).
2. Organizacje rządowe (~28% wartości).
3. Politycy (~25% wartości).
4. Programy telewizyjne (~14% wartości).

Problemem badawczym jest **wieloklasowa klasyfikacja wierzchołków** — przewidzenie kategorii każdej strony na podstawie jej cech oraz struktury grafu.

Analizowany graf składa się z 22 470 wierzchołków, 171 002 krawędzi, jest strukturą spójną (tzn. z każdego wierzchołka można dotrzeć do każdego innego), a średni stopień wierzchołka wynosi 15,22, co wskazuje na intensywną siatkę połączeń. Dla porównania maksymalny stopień wierzchołka to 709. Na zbiór treningowy/walidacyjny i testowy wybrano losowo odpowiednio 15 760, 3370 i 3370 wierzchołków.

[1] *https://snap.stanford.edu/data/facebook-large-page-page-network.html* [dostęp: 20 listopada 2025].

Na zbiór cech wierzchołków składa się macierz 31 zanonimizowanych wartości numerycznych, reprezentujących rozmaite cechy poszczególnych witryn.

Ze względu na wielkość grafu i wektora cech oraz społecznościowy charakter problemu do treningu wybrano metodę próbkowania wycinków uczących w oparciu o społeczności (omawianą w podrozdziale 5.2.2).

Pierwszą fazą eksperymentu był wybór hiperparametrów i optymalizacja modeli. Optymalizator dokonywał wyboru spośród:

- architektur splotu SAGE, GAT, GCN oraz ClusterGCN (będącego odmianą GCN przeznaczoną dla problemów społecznościowych);
- agregacji pojedynczych (suma/średnia/maksimum) oraz ich kombinacji dwu- i trzyelementowych;
- mechanizmu połączeń skokowych z różnymi sposobami łączenia poprzednich warstw (konkatenacja/maksimum/LSTM);
- ewentualnie dodatkowych hiperparametrów przeznaczonych dla poszczególnych modeli splotu (takich jak normalizacja SAGE lub liczba „głów" uwagi).

Optymalizacja hiperparametrów była prowadzona za pomocą biblioteki Optuna z wykorzystaniem zbiorów treningowego i walidacyjnego. W jej wyniku wybrane zostały najlepsze hiperparametry (tabela 6.1).

TABELA 6.1. Hiperparametry dla modelu klasyfikacji stron na Facebooku

Hiperparametr	Wartość
Architektura splotu	SAGE
Liczba neuronów w pojedynczej warstwie	32
Liczba warstw	3
Agregacja	Maksimum
Połączenia skokowe	Konkatenacja
Normalizacja	Tak
Projekcja	Tak
Normalizacja próbki treningowej	Tak

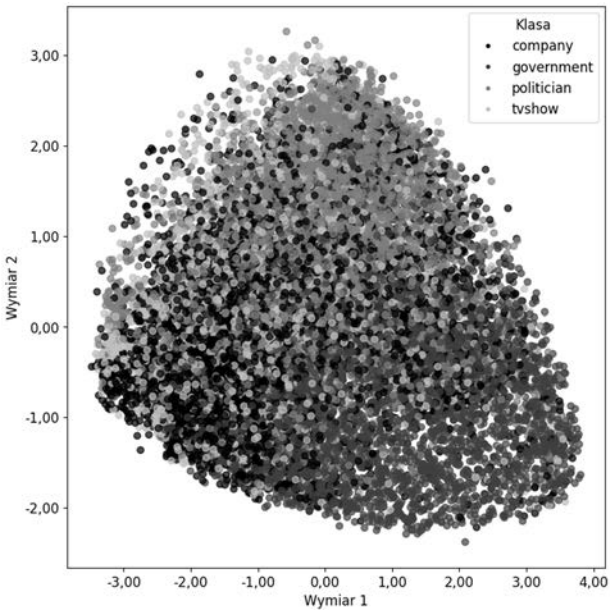
Po zakończeniu tego procesu przystąpiono do szkolenia modelu z pomocą biblioteki PyTorch Lightning. Trening prowadzony był przez 50 epok z uruchomionym mechanizmem wczesnego przerywania po 3 epokach w przypadku pogorszenia wyników oraz bieżącym zapisywaniem najlepszego modelu.

Tabela 6.2 podsumowuje uzyskane wyniki. Widać ewidentną poprawę zarówno dla metryki F1, jak i AUC dla zbioru walidacyjnego — przed treningiem i po nim. Ostateczny wynik dla zbioru testowego nie odbiega znacząco od wyników walidacyjnych.

TABELA 6.2. Wyniki klasyfikacji zbioru danych stron na Facebooku

Metryka	Zbiór	Faza	
		Przed treningiem	Po treningu
F1	Walidacyjny	0,164	0,686
	Testowy	–	0,671
AUC	Walidacyjny	0,493	0,839
	Testowy	–	0,841

Oprócz klasyfikacji za pomocą modelu można go także wykorzystać do wizualizacji zbioru danych. Budując wektory reprezentacji wierzchołków, a następnie poddając je redukcji wymiarowości, można stworzyć reprezentację wektorów osadzenia na płaszczyźnie. Jest to niezwykle przydatne w sytuacji, gdy chcemy sprawdzić jakość wykonania zadania i separację poszczególnych klas. Rysunek 6.1 przedstawia uzyskany rezultat. Widać wyraźnie, że klasy takie jak `government`, `company` oraz `politician` zostały prawidłowo odseparowane i tworzą odróżniające się klastry na przeciwległych krańcach chmury punktów. Klasa `tvshow` jest wymieszana z pozostałymi. Model ewidentnie miał trudności z odróżnieniem jej od pozostałych.



RYSUNEK 6.1. Wizualizacja wyniku klasyfikacji stron na Facebooku

Przeprowadzony eksperyment pokazuje, że model sieci grafowej relatywnie dobrze poradził sobie z klasyfikacją stron na Facebooku. Polem do poprawy jest zdecydowanie jakość predykcji dla klasy `tvshow`, która zaniża wynik ogólny i (co jest widoczne w wizualizacji na płaszczyźnie) nie odróżnia się wyraźnie od pozostałych. Możliwe jest przeprowadzenie dalszego strojenia parametrów i wykorzystanie bardziej złożonej architektury.

6.2. Klasyfikacja krawędzi

6.2.1. Postać formalna

Klasyfikacja krawędzi w grafach to zadanie, w którym celem jest przypisanie każdemu połączeniu (krawędzi) e_{uv} między wierzchołkami v oraz u etykiety y_{uv} należącej do zbioru Y. W zależności od specyfiki problemu etykieta może być binarna (0 — krawędź nie występuje, 1 — krawędź występuje) lub mieć wiele wartości odzwierciedlających określone zjawiska. Dla uproszczenia oraz w związku z prezentowanym dalej przykładem omówiona zostanie formalizacja tego działania dla klasyfikacji binarnej.

W rozwiązywaniu tego zadania stosuje się podejście dwuetapowe:

1. **Krok 1.** Budowanie reprezentacji wierzchołków z wykorzystaniem grafowych sieci neuronowych:

$$\boldsymbol{h}_v = \mathrm{GNN}(G, \boldsymbol{X}), v \in V, \tag{47}$$

$$\mathcal{H} = \{\boldsymbol{h}_v \mid \forall v \in V\}, \tag{48}$$

   gdzie G to graf z wierzchołkami i krawędziami, $\boldsymbol{X}$ to macierz cech wierzchołków, a GNN to model splotu grafowego.

2. **Krok 2.** Predykcja etykiety dla par wierzchołków $\{(u, v) \mid u, v \in V\}$ za pomocą wyspecjalizowanego modelu LinkPred. Jako wejście wykorzystuje on wektory reprezentacji skonstruowane przez sieć GNN:

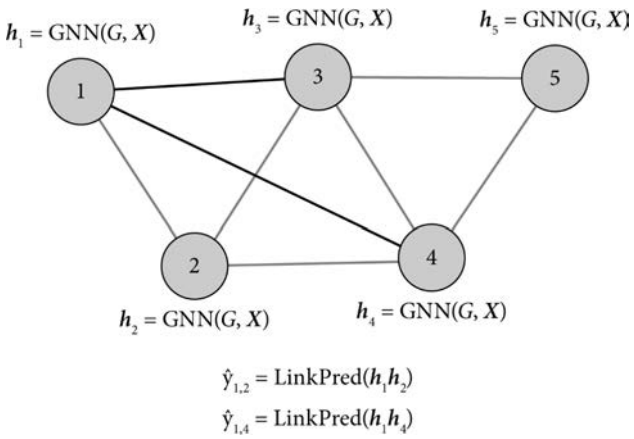
$$\hat{y}_{uv} = \mathrm{LinkPred}(\boldsymbol{h}_v, \boldsymbol{h}_u), \hat{y}_{uv} \in [0, 1]. \tag{49}$$

Wynik predykcji, będący prawdopodobieństwem istnienia/nieistnienia krawędzi, jest następnie podstawą do obliczenia funkcji kosztu, najczęściej entropii krzyżowej lub binarnej:

$$\mathcal{L}_{BCE} = \frac{1}{|E|} \sum_{(u,v)\in E} \left[y_{uv} \log \hat{y}_{uv} + (1 - y_{uv}) \log(1 - \hat{y}_{uv}) \right]. \tag{50}$$

Opisane zadanie można przedstawić graficznie jak na ilustracji poniżej (rysunek 6.2). Początkowo model GNN tworzy reprezentację wektorową każdego wierzchołka. Następnie pary takich reprezentacji (dla wierzchołków połączonych krawędziami) są wykorzystywane do klasyfikacji połączeń (prawdziwe/istniejące — jasne; nieprawdziwe/nieistniejące — ciemne).

W tak przedstawionej formalizacji problemu kryje się jednak pewna pułapka. Standardowo bowiem graf G zawiera tylko krawędzie realnie istniejące — „prawdziwe". Aby przeprowadzić proces nauki, konieczne jest sztuczne dodanie krawędzi „negatywnych" — oznaczonych etykietą „0" (Hamilton, 2020; Hamilton, Ying, Leskovec, 2017). Taki proces nosi nazwę „próbkowania negatywnego" (ang. *negative sampling*) i jest szeroko stosowany w zadaniach takich jak systemy rekomendacyjne (Liu, Zhong, Che et al., 2022), a nawet analiza języka naturalnego (Yang, Ding, Huang et al., 2024). Generowane negatywne próbki mogą być całkowicie losowe lub uwzględniać bardziej zaawansowane kryteria (np. odległość topologiczną między wierzchołkami).



RYSUNEK 6.2. Wizualizacja dwustopniowego procesu klasyfikacji wierzchołków

Biblioteka PyG oferuje kilka narzędzi wspomagających proces próbkowania negatywnego:

- `RandomLinkSplit` — umożliwia jednorazowe i statyczne dodanie krawędzi negatywnych do zbiorów treningowego, walidacyjnego i testowego. Narzędzie to pozwala na kontrolowanie proporcji negatywnych próbek względem pozytywnych poprzez parametr `neg_sampling_ratio`.
- `LinkNeighborLoader` — stosowany podczas konstruowania próbek uczących (ang. *batch*). Dynamicznie generuje negatywne próbki dla każdej iteracji treningowej, co pozwala na bardziej efektywne wykorzystanie danych i zwiększenie różnorodności próbek.

Mając na uwadze powyższe, po zastosowaniu próbkowania negatywnego zbiór krawędzi grafu E zawiera zarówno krawędzie „prawdziwe" (realnie istniejące, których etykieta klasy to 1), jak i krawędzie „fałszywe" (nieistniejące, których etykieta klasy to 0): $E = E^+ \cup E^-$, co umożliwia przeprowadzenie procesu uczenia i klasyfikacji binarnej.

6.2.2. Znaczenie problemu i zastosowania

Klasyfikacja krawędzi odgrywa kluczową rolę w wielu dziedzinach nauki i biznesu, umożliwiając analizę relacji między obiektami w sieciach grafowych. Jednym z istotnych zastosowań jest wykrywanie interakcji między lekami (ang. *Drug-Drug Interaction*, DDI), co ma szczególne znaczenie w farmakologii i opiece zdrowotnej. Identyfikacja potencjalnie niebezpiecznych interakcji pozwala na zwiększenie bezpieczeństwa terapii oraz opracowanie skuteczniejszych rekomendacji medycznych (Han, Xie, Li et al., 2022). Problem ten zostanie szerzej omówiony w kolejnym podrozdziale.

W analizie sieci społecznościowych klasyfikacja krawędzi służy do wykrywania powiązań między użytkownikami, co znajduje zastosowanie w systemach rekomendacji znajomości, identyfikacji osób o kluczowym wpływie w danej społeczności oraz detekcji fałszywych kont i botów (Silva, Correia, Maziero, 2023). W sektorze finansowym natomiast umożliwia wykrywanie anomalii poprzez analizę nietypowych

transakcji i powiązań między podmiotami, co wspomaga przeciwdziałanie oszustwom oraz nieprawidłowościom w systemach bankowych i płatniczych (Johannessen, Jullum, 2023; Pereira, Murai, 2021).

Biologia obliczeniowa wykorzystuje klasyfikację krawędzi do badania zależności między białkami i genami. Grafowe sieci neuronowe pozwalają na identyfikację nowych połączeń w sieciach biologicznych, co może prowadzić do głębszego zrozumienia mechanizmów chorób oraz wskazania potencjalnych celów terapeutycznych (Muzio, O'Bray, Borgwardt, 2021; Wu, Gao, Zeng et al., 2022).

6.2.3. Badanie oddziaływania pomiędzy lekami — klasyfikacja krawędzi

Eksperyment 12. Klasyfikacja interakcji pomiędzy lekami

Implementacja eksperymentu znajduje się w notatniku *rozdzial_6_zastosowania_sieci_gnn/02_interakcje_lekow_klasyfikacja_wierzcholkow.ipynb*.

Ze względu na stopień skomplikowania i objętość omawianego przykładu w części książkowej zostaną omówione założenia, parametry i wyniki. Detale implementacyjne znajdują się w dołączonych materiałach (notatniku).

W tym podrozdziale analizujemy praktyczne zastosowanie klasyfikacji krawędzi w grafach na przykładzie sieci interakcji leków (DDI). Studium przypadku opiera się na zbiorze danych `ogbl-ddi`, który pochodzi z platformy Open Graph Benchmark (OGB)[2] i został zaprojektowany do oceny metod uczenia maszynowego na grafach dla celów branży farmaceutycznej (Guney, 2017). Zbiór ten przedstawia jednorodny, nieskierowany i nieważony graf, w którym:

1. **Wierzchołki** reprezentują leki zatwierdzone przez FDA lub będące w fazie eksperymentalnej.
2. **Krawędzie** odzwierciedlają interakcje między lekami, czyli sytuacje, w których wspólne stosowanie dwóch leków prowadzi do efektu istotnie różniącego się od oczekiwanego działania każdego z nich osobno.

Problemem badawczym jest przewidywanie interakcji między lekami na podstawie już znanych powiązań. W tym celu wykorzystuje się zarówno miary F1, AUC i czułość jako metryki ewaluacji, jak i *K* najlepszych trafień (ang. *Hits@K*), która ocenia zdolność modelu do budowania poprawnego rankingu: prawdziwe interakcje (krawędzie) powinny uzyskiwać wyższe prawdopodobieństwa niż losowo wybrane próbki negatywne. Dla tego zbioru danych przyjęto specyficzny podział na zestawy treningowy, walidacyjny i testowy, bazujący na oddziaływaniach leków na białka. Dzięki temu testowane są możliwości modeli w generowaniu praktycznie użytecznych prognoz dla leków o odmiennych mechanizmach biologicznego oddziaływania w porównaniu do substancji obecnych w zbiorze treningowym.

[2] *https://ogb.stanford.edu/docs/linkprop/#ogbl-ddi* [dostęp: 20 listopada 2025].

Zbiór `ogbl-ddi` cechuje się dużą objętością: zawiera **4267 wierzchołków** oraz **1 334 889 krawędzi**, co czyni go wymagającym pod względem obliczeniowym. Na potrzeby tej książki oraz towarzyszącego przewodnika zdecydowano się na zmniejszenie jego wielkości. W tym celu podjęto następujące kroki:

1. Przeanalizowano spójne składowe grafu, czyli połączone elementy sieci.
2. Spośród wszystkich spójnych składowych wybrano tę, która zawiera około 20% wszystkich wierzchołków grafu.
3. Ograniczono graf do tej wybranej składowej, zachowując jedynie około 20% wierzchołków oraz krawędzie łączące je ze sobą.
4. Podzielono dane na zestawy: zbiory treningowy (80%), walidacyjny (10%) oraz testowy (10%). Dodatkowo dodano krawędzie negatywne (brak interakcji) w proporcji 1:1 względem krawędzi pozytywnych (rzeczywistych interakcji).

Po przeprowadzeniu powyższych kroków uzyskano następujące próbki:

1. **Liczba wierzchołków**: 845 we wszystkich zbiorach (treningowym, walidacyjnym i testowym).
2. **Zbiór treningowy**: 136 384 krawędzi (pozytywnych i negatywnych).
3. **Zbiór walidacyjny**: 17 046 krawędzi (pozytywnych i negatywnych).
4. **Zbiór testowy**: 17 046 krawędzi (pozytywnych i negatywnych).

Pomimo redukcji rozmiaru danych proces treningu pozostaje czasochłonny ze względu na dużą liczbę krawędzi i złożoność modelu.

Szkolenie modelu odbywało się z wykorzystaniem mechanizmów próbkowania w oparciu o sąsiedztwo (opisanych w podrozdziale 5.2.1), dobranych w taki sposób, aby zapewnić kompromis między efektywnością uczenia a jego czasem. Szczególnie warto zwrócić uwagę na fragment kodu zaprezentowany poniżej, ukazujący konfigurację klasy `LinkNeighborLoader`:

```
1. train_loader = pyg.loader.LinkNeighborLoader(
2.     train_data,
3.     batch_size=1024,
4.     num_neighbors=[-1, 15, 10, 5],
5.     shuffle=True,
6.     edge_label_index=train_data.edge_label_index,
7.     edge_label=train_data.edge_label,
8.     num_workers=3)
```

Przedstawiona metoda próbkowania zbioru treningowego rekurencyjnie pobiera sąsiadów w czterech krokach (wiersz 4.):

- wszystkich bezpośrednich sąsiadów danego wierzchołka (wartość –1);
- 15 sąsiadów dla każdego sąsiada bezpośredniego;
- kolejnych 10 w odległości dwóch skoków;
- kolejnych 5 w odległości trzech skoków.

Takie próbkowanie może szybko doprowadzić do gwałtownego wzrostu liczby wierzchołków. Z tego względu ograniczono całkowitą wielkość próbki do 1024 elementów (wiersz 3.).

W przypadku zbiorów walidacyjnego i testowego, ze względu na mniejsze rozmiary, możliwe jest ich całościowe załadowanie do pamięci od razu.

Podobnie jak wcześniej, pierwszą fazę eksperymentu stanowił wybór hiperparametrów i optymalizacja modeli. Tym razem (głównie ze względów wydajnościowych) wykorzystana została tylko architektura SAGE, po której następował model dokonujący klasyfikacji wierzchołka. Hiperparametry brane podczas optymalizacji to:

- liczba warstw i neuronów dla modelu SAGE;
- sposoby agregacji sąsiednich wierzchołków (suma/średnia/maksimum oraz ich kombinacje 2- i 3-elementowe);
- mechanizm połączeń skokowych z różnymi sposobami łączenia poprzednich warstw (konkatenacja/maksimum/LSTM);
- architektura sieci klasyfikującej połączenia (liczba warstw i neuronów).

Optymalizacja hiperparametrów była prowadzona za pomocą biblioteki Optuna z wykorzystaniem zbiorów treningowego i walidacyjnego.

Tabela 6.3 przedstawia końcowe hiperparametry.

TABELA 6.3. Przykładowe zoptymalizowane hiperparametry dla predykcji połączeń

Hiperparametr	Wartość
Liczba neuronów w warstwie splotu	128
Liczba warstw	3
Agregacja	Suma, maksimum, średnia
Połączenia skokowe	Konkatenacja
Normalizacja	Tak
Projekcja	Tak
Normalizacja próbki treningowej	Tak
Liczba warstw klasyfikatora połączeń	3
Liczba neuronów w warstwie klasyfikatora	256

Po zakończeniu tego procesu przystąpiono do szkolenia modelu. Trening prowadzony był przez 25 epok z uruchomionym mechanizmem wczesnego przerywania (po 3 epokach) w przypadku pogorszenia wyników oraz bieżącym zapisywaniem najlepszego modelu.

Tabela 6.4 podsumowuje uzyskane wyniki. Widać ewidentną poprawę zarówno dla metryki F1, jak i AUC dla zbioru walidacyjnego — przed treningiem i po nim. Ostateczny wynik dla zbioru testowego nie odbiega znacząco od wyników walidacyjnych.

Model wykazał znaczącą poprawę wyników w procesie uczenia, co jest widoczne we wszystkich analizowanych metrykach. W zadaniu predykcji istnienia lub braku krawędzi, opierającym się na rozróżnianiu próbek negatywnych od rzeczywistych, kluczowe znaczenie mają współczynniki trafień. Określają one, jak często spośród K predykcji o najwyższym prawdopodobieństwie rzeczywiste krawędzie uzyskują wyższe wskazania niż obiekty nieistniejące. W przeprowadzonym eksperymencie zauważalna jest istotna różnica między wskaźnikami trafień dla top 10 i top 20 predykcji dla zbioru testowego. Średnio 55% spośród 10 najwyżej ocenionych predykcji poprawnie wskazuje istniejące połączenia. W przypadku rozszerzenia analizy do 20 najlepszych predykcji odsetek ten wzrasta do około 78%.

TABELA 6.4. Wyniki klasyfikacji połączeń w modelu interakcji leków

Metryka	Zbiór	Faza	
		Przed treningiem	Po treningu
F1	Walidacyjny	0,0	0,861
	Testowy	–	0,665
AUC	Walidacyjny	0,395	0,93
	Testowy	–	0,92
Śr. trafień w top 10	Walidacyjny	0,0	0,57
	Testowy	–	0,55
Śr. trafień w top 20	Walidacyjny	0,0	0,79
	Testowy	–	0,78

Dłuższy trening oraz bardziej agresywne strojenie hiperparametrów mogłyby istotnie poprawić wyniki w zakresie top 10.

Opisane podejście znajduje szerokie zastosowanie w systemach rekomendacyjnych, gdzie kluczowe jest określenie, jaka część proponowanych użytkownikowi sugestii rzeczywiście spełnia jego oczekiwania.

6.3. Klasyfikacja grafów

6.3.1. Postać formalna

Klasyfikacja całych grafów jest kluczowym zagadnieniem w analizie danych strukturalnych, znajdującym szerokie zastosowanie w wielu gałęziach nauki i biznesu. W przeciwieństwie do klasyfikacji wierzchołków czy krawędzi, gdzie predykcje dotyczą lokalnych komponentów grafu, klasyfikacja grafów koncentruje się na jego globalnych właściwościach.

Celem tego zadania jest przypisanie pojedynczej etykiety $y \in \mathcal{Y}$ do całego grafu $G = (V, E)$, gdzie $\mathcal{Y}$ oznacza zbiór możliwych klas. Proces ten można podzielić na dwa główne etapy:

1. **Konstrukcja reprezentacji grafu**, w której globalne właściwości strukturalne i atrybuty wierzchołków są transformowane do zwartych reprezentacji numerycznych.
2. **Klasyfikacja oparta na tych reprezentacjach**, gdzie model uczący dokonuje predykcji etykiety na podstawie osadzenia całego grafu.

Pierwszym krokiem jest wyznaczenie reprezentacji wierzchołków przy użyciu dowolnej grafowej sieci neuronowej (ang. *Graph Neural Network*, GNN). Proces ten jest analogiczny do tego stosowanego w klasyfikacji krawędzi i wierzchołków:

$$\boldsymbol{h}_v = \mathrm{GNN}(G, \boldsymbol{X}), v \in V,$$

$$\mathcal{H} = \{\, \boldsymbol{h}_v \mid \forall v \in V \,\}.$$

Następnie uzyskane osadzenia wierzchołków są agregowane do jednej reprezentacji opisującej cały graf:

$$\boldsymbol{h}_G = \text{GlobalPool}(\{\boldsymbol{h}_v \mid \forall v \in V\}). \tag{51}$$

Funkcja GlobalPool(·) pełni rolę agregatora i może przyjmować różne formy, takie jak średnia, maksimum, suma czy mechanizmy bardziej zaawansowane, np. uwzględniające stopnie wierzchołków.

Zdefiniowana reprezentacja $\boldsymbol{h}_G$ stanowi wejście do modelu klasyfikacyjnego, który przypisuje grafowi etykietę:

$$\hat{y}_G = \text{GraphClf}(\mathbf{h}_G), \tag{52}$$

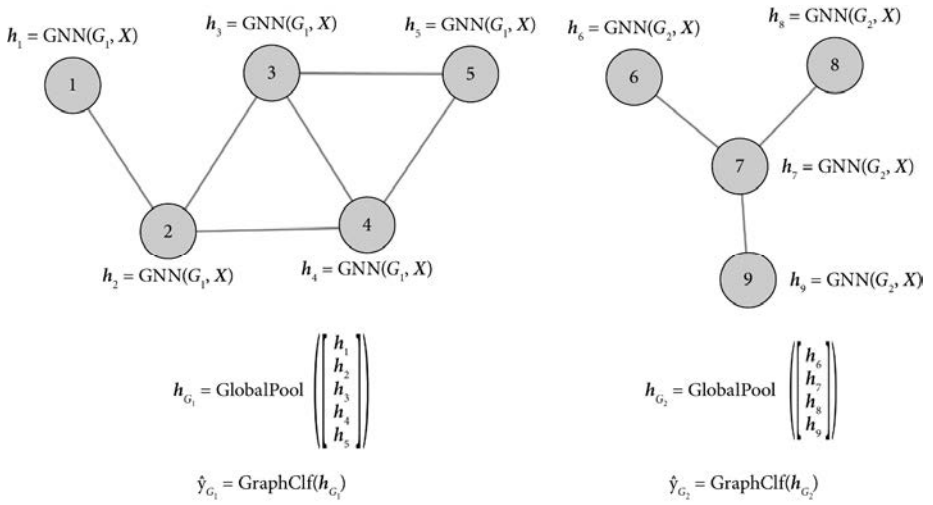
gdzie GraphClf(·) to dowolny klasyfikator, np. wielowarstwowa sieć neuronowa MLP. Proces optymalizacji przebiega poprzez minimalizację funkcji kosztu, np. entropii krzyżowej:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} loss\left(\text{GraphClf}\left(\boldsymbol{h}_{G_i}\right), y_i\right), \tag{53}$$

gdzie:

- $loss(\cdot,\cdot)$ to funkcja kosztu dla pojedynczego przykładu uczącego,
- $\boldsymbol{h}_{G_i}$ to wektor reprezentacji grafu G_i,
- y_i to rzeczywista etykieta przypisana do grafu G_i,
- N to liczba grafów w zbiorze uczącym.

Rysunek 6.3 ilustruje opisany proces.



RYSUNEK 6.3. Ilustracja procesu budowania reprezentacji całego grafu

W zadaniach klasyfikacji grafów często pracujemy z wieloma grafami, tj. zbiorem $\{G_1, G_2, \ldots, G_N\}$, gdzie każdy graf $G_i = (V_i, E_i)$ ma własny zestaw wierzchołków V_i i krawędzi E_i. Aby umożliwić efektywne uczenie modeli, grafy te muszą zostać

scalone w jedną wspólną strukturę. PyG oferuje wyspecjalizowane mechanizmy umożliwiające ich efektywne połączenie, obejmujące:

1. Konkatenację macierzy cech wierzchołków:

$$X = \text{concat}(\boldsymbol{X}_1, \boldsymbol{X}_2, \dots, \boldsymbol{X}_N), \tag{54}$$

gdzie $\boldsymbol{X_i} \in \mathbb{R}^{|V_i| \times \boldsymbol{d}}$ to macierz cech wierzchołków dla grafu G_i.

2. Łączenie macierzy sąsiedztwa w globalną strukturę:

$$A = \begin{bmatrix} \boldsymbol{A_1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & 0 & \boldsymbol{A}_N \end{bmatrix}, \tag{55}$$

gdzie każda macierz $\boldsymbol{A}_i$ jest macierzą sąsiedztwa w grafie G_i.

3. Dodanie identyfikatora przynależności każdego wierzchołka do konkretnego grafu:

$$\boldsymbol{batch} = \left[\underbrace{1,1,\dots,1}_{|V_1|}, \underbrace{2,2,\dots,2}_{|V_2|}, \dots, \underbrace{N,N,\dots,N}_{|V_N|}\right], \tag{56}$$

gdzie wartość w wektorze $\boldsymbol{batch}$ wskazuje numer grafu źródłowego dla danego wierzchołka.

Posługując się grafem z przywołanego wcześniej przykładu (patrz rysunek 6.3), możemy przyjąć, że macierze cech wierzchołków dla grafu 1 (wierzchołki 1 – 5) oraz grafu 2 (wierzchołki 6 – 9) wyglądają następująco:

$$\boldsymbol{X}_1 = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \\ 4 & 4 \\ 5 & 5 \end{bmatrix},$$

$$\boldsymbol{X}_2 = \begin{bmatrix} 6 & 6 \\ 7 & 7 \\ 8 & 8 \\ 9 & 9 \end{bmatrix}.$$

Dla uproszczenia przyjęto, że cechy wierzchołków to dwuelementowe wektory o wartościach odpowiadających indeksowi wierzchołka.

Lista sąsiedztwa dla obu grafów przedstawia się następująco:

$$\boldsymbol{A}_1 = \begin{bmatrix} 1 & 2 & 2 & 3 & 3 \\ 2 & 3 & 4 & 4 & 5 \end{bmatrix},$$

$$\boldsymbol{A}_2 = \begin{bmatrix} 6 & 8 & 9 \\ 7 & 7 & 7 \end{bmatrix}.$$

Mimo że oba grafy są nieskierowane, dla uproszczenia przedstawiono jednokierunkowe listy sąsiedztwa.

Podczas budowania próbki treningowej PyG skonstruowałby struktury zbliżone do przedstawionych poniżej, łącząc ze sobą grafy:

$$\boldsymbol{X} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \\ 4 & 4 \\ 5 & 5 \\ 6 & 6 \\ 7 & 7 \\ 8 & 8 \\ 9 & 9 \end{bmatrix}$$

$$\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 2 & 3 & 3 & 6 & 8 & 9 \\ 2 & 3 & 4 & 4 & 5 & 7 & 7 & 7 \end{bmatrix}$$

$$\mathbf{batch} = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 2 \quad 2 \quad 2]$$

Indeksy w macierzy **batch** odpowiadają przypisaniu przykładów do oryginalnego grafu. W ten sposób nie trzeba w żaden sposób modyfikować standardowego kodu służącego do uczenia modeli sieci grafowych.

Fragment kodu poniżej ilustruje przykład skonstruowania opisywanej próbki (pamiętajmy o indeksowaniu numerów wierzchołków od zera).

```
g1 = pyg.data.Data()
g1.x = th.tensor([[i, i] for i in range(1, 6)], dtype=th.float)
g1.edge_index = th.tensor([
    [0, 1, 1, 2, 2, 3],
    [1, 2, 3, 3, 4, 4]
])

g2 = pyg.data.Data()
g2.x = th.tensor([[i, i] for i in range(6, 10)], dtype=th.float)
g2.edge_index = th.tensor([
    [0, 2, 3],
    [1, 1, 1]
])

small_data_loader = pyg.loader.DataLoader([g1, g2], batch_size=10)
batch = next(iter(small_data_loader))
batch
>> DataBatch(x=[9, 2], edge_index=[2, 9], batch=[9], ptr=[3])
batch.x
>> tensor([[1., 1.],
>>         [2., 2.],
>>         [3., 3.],
>>         [4., 4.],
>>         [5., 5.],
>>         [6., 6.],
>>         [7., 7.],
>>         [8., 8.],
>>          [9., 9.]])
batch.edge_index
>> tensor([[0, 1, 1, 2, 2, 3, 5, 7, 8],
>>         [1, 2, 3, 3, 4, 4, 6, 6, 6]])
batch.batch
>> tensor([0, 0, 0, 0, 0, 1, 1, 1, 1])
```

Skorowidz

Cicha rewolucja, która nadeszła

Grafowe sieci neuronowe (GNN, ang. *graph neural networks*) to klasa modeli uczenia głębokiego przeznaczona do analizy danych o strukturze grafowej. W początkowym okresie ich rozwój był ograniczony przez brak efektywnych metod projektowania i optymalizacji; w ostatnich latach bariery te w dużej mierze zostały pokonane, co przełożyło się na dynamiczny postęp teorii i praktyki. Modele GNN znajdują zastosowanie między innymi w analizie sieci społecznościowych, optymalizacji procesów logistycznych, marketingu i pracy z bazami wiedzy.

Ta książka zawiera kompleksowe opracowanie tematyki sieci grafowych w kontekście uczenia maszynowego. Tym samym wypełnia istotną lukę na polskim rynku wydawniczym, oferując połączenie solidnych podstaw teoretycznych z praktycznym zastosowaniem GNN. To przewodnik, który systematycznie przeprowadza przez kolejne zagadnienia związane z sieciami grafowymi:

- od narzędzi klasycznej analizy grafów w środowisku Pythona i wybranych zagadnień teorii grafów
- przez wprowadzenie do grafowych sieci neuronowych, a także przegląd wybranych warstw splotu grafowego i dobrych praktyk ich projektowania
- po zagadnienia związane ze szkoleniem sieci GNN i praktyczne przykłady ich zastosowań

Wydawnictwo Naukowe Helion

helion.pl

HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-289-3390-3

Cena: 79,00 zł

Dr inż. **Filip Wójcik**

— analityk danych, zajmuje się systemami uczenia maszynowego i sztucznej inteligencji od 2012 roku. Projektował i wdrażał rozwiązania ML/AI dla międzynarodowych i polskich przedsiębiorstw w sektorach finansowym, telekomunikacyjnym i logistycznym, pracując jako analityk danych (data scientist), kierownik zespołu badawczego i lider techniczny. Łączy działalność komercyjną z pracą akademicką; od 2017 roku związany z Uniwersytetem Ekonomicznym we Wrocławiu, gdzie specjalizuje się w optymalizacji procesów decyzyjnych z wykorzystaniem modeli uczenia maszynowego, w szczególności głębokich sieci neuronowych. Chętnie dzieli się wiedzą i doświadczeniem ze słuchaczami w ramach studiów podyplomowych, kursów, a także Biznesowego Indywidualnego Programu Studiów (BIPS). Autor licznych recenzowanych publikacji naukowych, prelegent międzynarodowych konferencji poświęconych najnowszym trendom w dziedzinie uczenia maszynowego.