Мини-проект

Сравнение эффективности графовых нейронных сетей и классических методов машинного обучения для классификации новостных статей

Студент группы ИУ6-13М
Левкович И.А.

Декабрь 2025

Аннотация

В данном мини-проекте исследуется эффективность графовых нейронных сетей (GNN) по сравнению с классическими методами машинного обучения (Random Forest) для задачи классификации новостных статей по странам. Гипотеза о превосходстве GNN благодаря учёту структурных связей между документами проверяется на датасете мировых новостей. Реализованы модели GCN (Graph Convolutional Network) и Random Forest, проведены эксперименты с оценкой по метрикам Accuracy и F1-Score. Результаты показывают, что Random Forest (Accuracy: 48.47%, F1-Score: 47.59%) превосходит GCN (Accuracy: 34.80%, F1-Score: 31.22%), что частично опровергает гипотезу. Обсуждаются возможные причины, включая особенности построения графа и размер датасета.

Содержание

1 Введение

1.1 Проблематика

Классификация текстовых данных, таких как новостные статьи, является ключевой задачей в обработке естественного языка (NLP). Традиционные методы машинного обучения, например Random Forest, часто используются для этой цели, но они не учитывают потенциальные структурные связи между документами (например, тематические или стилистические сходства). Графовые нейронные сети (GNN) предлагают подход к моделированию таких связей, что может повысить качество классификации.

1.2 Гипотеза исследования

Графовые нейронные сети (GNN) показывают лучшие результаты в классификации новостных статей по сравнению с классическими методами машинного обучения благодаря своей способности учитывать структурные связи между документами.

1.3 Цели и задачи

- Сравнить эффективность GNN (на основе архитектуры GCN) и Random Forest для классификации новостных статей по странам.
- Реализовать модели GCN и Random Forest на Python с использованием библиотек PyTorch Geometric и scikit-learn.
- Провести эксперименты на датасете мировых новостей с оценкой по метрикам Accuracy и F1-Score.
- Проанализировать результаты, включая визуализацию обучения, матрицы ошибок и проекции эмбеддингов.
- Обсудить причины наблюдаемых результатов.

2 Теоретическая основа

2.1 Графовые нейронные сети (GNN)

GNN — класс нейросетевых архитектур, предназначенных для обработки графовых данных. Основной механизм — передача сообщений (message passing), который позволяет узлам графа обмениваться информацией с соседями. Формально граф определяется как:

$$G = (V, E, X, A) \tag{1}$$

где V — множество узлов, E — множество рёбер, X — матрица признаков узлов, A — матрица смежности.

2.1.1 Механизм Message Passing

Обобщённая формула для GCN [1]:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{2}$$

где:

- $\tilde{A} = A + I$ — матрица смежности с добавленными self-loops,
- $\tilde{D}$ — диагональная матрица степеней узлов для $\tilde{A}$,
- $H^{(l)}$ — активации на слое l,
- $W^{(l)}$ — обучаемые веса слоя l,
- σ — функция активации (например, ReLU).

2.2 Random Forest

Random Forest — ансамблевый метод машинного обучения, строящий множество решающих деревьев и агрегирующий их предсказания. Он эффективен для задач классификации благодаря устойчивости к переобучению и способности работать с разнородными данными. Формально предсказание для Random Forest:

$$\hat{y} = \text{mode}\left(\{T_k(x)\}_{k=1}^{K}\right) \tag{3}$$

где T_k — k-е дерево в ансамбле, K — общее количество деревьев, mode — мода (наиболее частый класс).

3 Методология

3.1 Постановка задачи

Задача — многоклассовая классификация новостных статей по странам (24 класса). Входные данные:

- Тексты новостей (признаки извлечены с помощью TF-IDF).
- Граф, построенный на основе косинусной схожести текстов (рёбра между наиболее похожими статьями).

Выход: метка страны для каждой статьи.

3.2 Модели

3.2.1 GCN (Graph Convolutional Network)

Реализована 5-слойная модель GCN с архитектурой:

- Входной слой: размерность признаков TF-IDF (200-500).
- Скрытые слои: 64, 128, 64, 32, 16 нейронов.
- Выходной слой: 24 нейрона (по числу стран).
- Функция активации: ReLU, dropout=0.3.
- Функция потерь: Negative Log Likelihood (NLL).

3.2.2 Random Forest

Реализован с параметрами:

- Количество деревьев: 50.
- Критерий разделения: Gini impurity.
- Максимальная глубина: не ограничена.
- Использование bootstrap выборок.

3.3 Оптимизация

Для GCN используется оптимизатор Adam:

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4}$$

с параметрами: $\eta = 0.01$, weight_decay $= 5 \times 10^{-4}$, количество эпох: 1000.

4 Экспериментальная часть

4.1 Датасет

Использован датасет мировых новостей (World News Dataset) со следующими характеристиками:

- **Количество статей**: 93,444 (после очистки).
- **Количество классов**: 24 страны.
- **Признаки**: TF-IDF векторы (максимум 500 признаков).
- **Граф**: Построен на основе косинусной схожести текстов (рёбра между 5 ближайшими соседями, порог схожести > 0.15). Количество узлов: до 1000, рёбер: 4000.
- **Разбиение**: Для Random Forest — 70% train, 30% test. Для GCN — 70% train, 15% validation, 15% test.

4.2 Метрики оценки

- **Accuracy**: Доля правильно классифицированных статей.
- **F1-Score**: Среднее гармоническое precision и recall (взвешенное по поддержке классов).
- **Confusion Matrix**: Визуализация ошибок классификации.
- **Проекции эмбеддингов**: t-SNE и PCA для анализа распределения классов.

5 Результаты и обсуждение

5.1 Сравнение моделей

Таблица 1: Сравнение Random Forest и GCN на тестовой выборке

Модель	Тип	Accuracy	F1-Score	Время обучения
Random Forest	Классическая	0.4847	0.4759	2 мин
GCN	GNN	0.3480	0.3122	1 мин

Random Forest показал Accuracy 48.47% и F1-Score 47.59%, превосходя GCN (Accuracy 34.80%, F1-Score 31.22%). Это частично опровергает гипотезу о превосходстве GNN.

5.2 Визуализация обучения

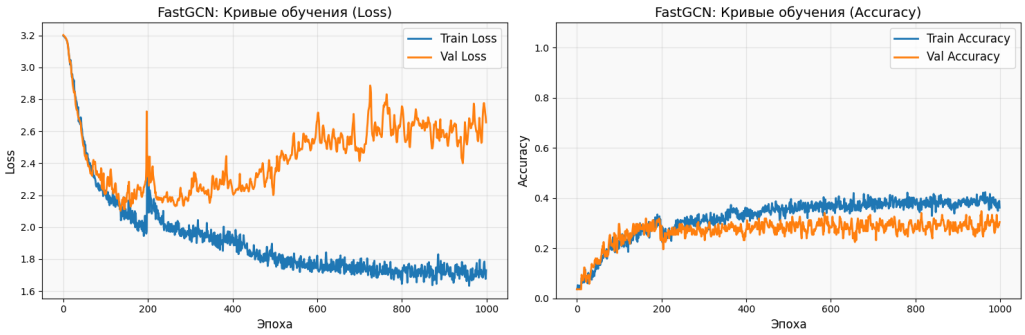

Рис. 1: Кривые обучения GCN: Loss и Accuracy на train/validation. Видно, что модель сходится, но accuracy на validation остаётся низкой(35%), а Loss на validation после 200 эпохи начинает расти вверх.

5.3 Визуализация графа после обучения GCN

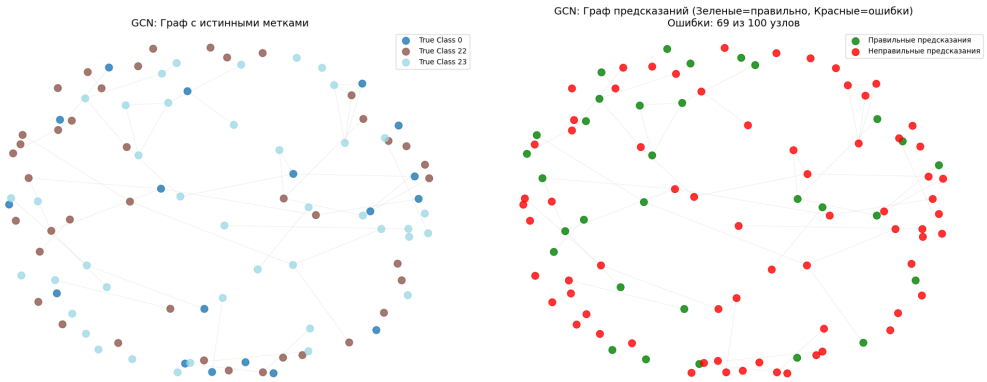

Рис. 2: Визуализация графа после обучения GCN.

5.4 Проекции эмбеддингов

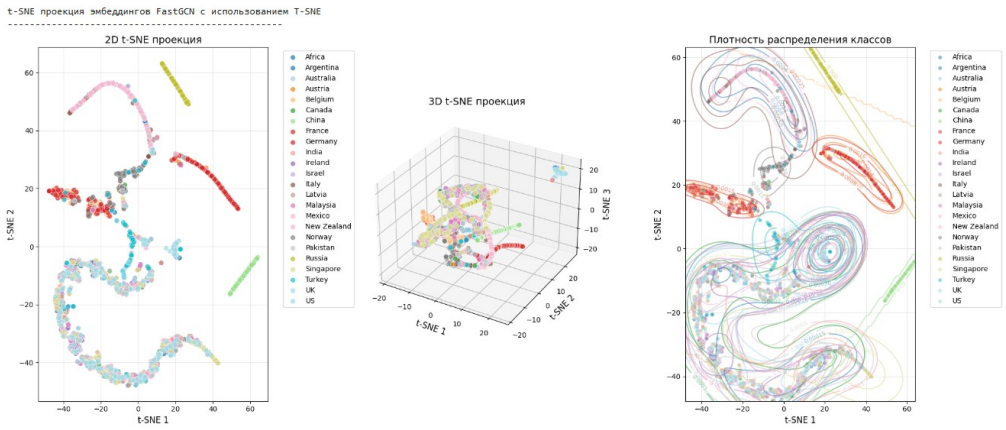

Рис. 3: PCA проекция эмбеддингов GCN с использованием PCA.

5.5 Анализ графа

- **Плотность графа**: Низкая (рёбра только между сильно похожими статьями), что может ограничивать способность GCN связи.
- **Размер датасета**: Для GCN использовано только 1000 статей (из-за вычислительных ограничений), тогда как Random Forest обучен на всём датасете (65,000+ train). Это могло повлиять на результаты.
- **Признаки**: TF-IDF векторы могут быть недостаточно информативны для GCN, которая полагается на структуру графа.

6 Выводы

6.1 Частичное опровержение гипотезы

Результаты эксперимента не подтверждают гипотезу о превосходстве GNN над классическими методами:

1. Random Forest показал более высокую Accuracy (48.47% против 34.80%) и F1-Score (47.59% против 31.22%).
2. GCN не смогла эффективно использовать структурные связи, возможно, из-за низкой плотности графа или недостаточного количества данных.
3. Визуализации показывают слабое разделение классов в эмбеддингах GCN.

6.2 Практическая значимость

- Random Forest остаётся эффективным методом для классификации текстов, особенно при ограниченных вычислительных ресурсах.
- GNN могут быть полезны в задачах, где связи между документами явные и плотные (например, социальные сети или цитирования).
- Построение графа на основе семантической схожести (например, с использованием BERT) могло бы улучшить результаты GCN.

6.3 Ограничения и дальнейшие исследования

1. **Ограничения**:
   - Датасет несбалансирован (некоторые страны представлены слабо).
   - Граф построен только на основе TF-IDF схожести, что может не отражать семантические связи.
   - Для GCN использована уменьшенная выборка из-за ограничений памяти.
2. **Дальнейшие исследования**:
   - Использование более богатых признаков (эмбеддинги BERT, Word2Vec).
   - Эксперименты с другими архитектурами GNN (GAT, GraphSAGE).
   - Построение графа на основе дополнительных данных.
   - Применение техник обработки несбалансированных данных (oversampling, weighting).

Заключение

В ходе мини-проекта проведено сравнение графовой нейронной сети (GCN) и классического метода (Random Forest) для классификации новостных статей по странам. Результаты показали, что Random Forest превосходит GCN по метрикам Accuracy и F1-Score, что частично опровергает гипотезу о преимуществе GNN благодаря учёту структурных связей. Возможные причины включают низкую плотность графа, ограниченный размер данных для GCN и простоту признаков TF-IDF. Для улучшения результатов GNN рекомендуется использовать семантические эмбеддинги и более сложные архитектуры графов. Исследование подчёркивает важность выбора метода в зависимости от характеристик данных и доступных ресурсов.

Список литературы

[1] Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[2] Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

[3] Fey, M., & Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428*.

[4] Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation, 28*(1), 11–21.

[5] Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.

A Ключевые фрагменты кода

A.1 Построение графа

```
from sklearn.metrics.pairwise import cosine_similarity
similarity_matrix = cosine_similarity(X_text_gnn)
edges = []
edge_weights = []
k = min(5, len(df_gnn) - 1)
for i in range(len(df_gnn)):
    similarities = similarity_matrix[i]
    similarities[i] = -1
    top_k_indices = np.argsort(similarities)[-k:]
    for j in top_k_indices:
        if similarities[j] > 0.15:
            edges.append([i, j])
            edge_weights.append(similarities[j])
```

A.2 Модель GCN

```
class GCNModel(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim, dropout=0.3):
        super(GCNModel, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, hidden_dim * 2)
        self.conv3 = GCNConv(hidden_dim * 2, hidden_dim)
        self.conv4 = GCNConv(hidden_dim, hidden_dim // 2)
        self.conv5 = GCNConv(hidden_dim // 2, hidden_dim // 4)
        self.lin1 = nn.Linear(hidden_dim // 4, hidden_dim // 8)
        self.lin2 = nn.Linear(hidden_dim // 8, output_dim)
        self.dropout = dropout

    def forward(self, data):
        x, edge_index, edge_weight = data.x, data.edge_index,
           data.edge_attr
        x = self.conv1(x, edge_index, edge_weight)
        x = F.relu(x)
        x = F.dropout(x, p=self.dropout, training=self.training)
        # ... remaining layers ...
        return F.log_softmax(x, dim=1)
```

A.3 Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=50, random_state=42,
   n_jobs=-1)
rf.fit(X_train, y_train)
predictions = rf.predict(X_test)
```