Graph Neural Networks for Hotel Recommendation and Quality Assessment

Iveta Mrázová[1], Marek Behún[2]

[1]*Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic*
[2]*Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic*

Abstract
Popular travel-related forums provide prodigious support to customers. We scraped 3,125,631 Tripadvisor reviews for 3,260 hotels from 2,296,247 unique authors. This data allows for an extensive exploration from the perspective of social networks. While the hotels and review authors correspond to the nodes of the corresponding network graph, the ratings represent edges. In this paper, we inspect the prospects of graph neural networks for recommender systems. The experiments conducted so far yielded promising results regarding modeling travel-related data despite disregarding textual or image-related parts of the reviews. We see the core contribution of our research in providing a proof of concept for amplifying the power of recommender systems with the principles of social networks and advanced machine learning.

Keywords
graph neural networks, social network analysis, link prediction, node´s score evolution, review rating prediction, hotel´s star category assessment

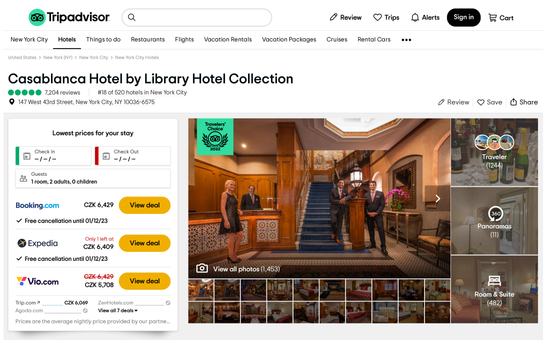
1. Introduction

Travel-related forums provide prodigious support to customers. In addition to possible accommodation mediation, these websites tender invaluable information on the equipment and overall quality of the hotels and their neighborhood. For example, Tripadvisor, Inc. [19] comprises about one billion reviews on eight million establishments. Successful recommender systems [15] would utilize the collected information, e.g., to suggest an alternative accommodation option.

Previous research in this field focused on text-based sentiment analysis of user opinions. Yet, the scraped data records allow for a study from the perspective of *social networks* (SNs). While the hotels and review authors correspond to nodes of an SN graph, the ratings represent edges. To facilitate learning in SNs, e.g., when modeling the evolution of a network, the concept of *graph neural networks* (GNNs) [17] might constitute a viable approach.

Our ultimate objective is thus to explore the prospects of GNNs in support of a travel-related recommender system. We see the core contribution of our research in providing a proof of concept for utilizing extensive data rich in structure in a way that allows autonomous learning of their mutually intertwined relationships.

Section 2 explains the principles of SNs, GNNs, and visualization of high-dimensional data. Section 3 outlines the construction of relevant SNs. Section 4 specifies heterogeneous GNNs [9] and graph convolutional recurrent networks (GCRN) [18] we have used. The analyzed tasks refer to review rating prediction, hotel category assessment, and hotel score evolution. Conclusions summarize the results and outline future enhancements.



Casablanca Hotel by Library Hotel Collection
7,204 reviews #18 of 520 hotels in New York City
147 West 43rd Street, New York City, NY 10036-6575

Lowest prices for your stay
Check In –/–/–
Check Out –/–/–
Guests 1 room, 2 adults, 0 children
Booking.com CZK 6,429 View deal
✓ Free cancellation until 01/12/23
Expedia Only 1 left at CZK 6,409 View deal
✓ Free cancellation until 01/12/23
Vio.com ~~CZK 6,429~~ CZK 5,708 View deal
Trip.com CZK 6,069 ZenHotels.com
Agoda.com View all 7 deals
Prices are the average nightly price provided by our partne...

Figure 1: Screenshot of the top section of the Tripadvisor page for the Casablanca Hotel in New York City [19].

2. Related Work

Static SNs are defined as a tuple (V, E, a_V, a_E). (V, E) denotes the graph of actors and relationships between them, a_V states the actor and a_E the edge attributes, resp. For the Tripadvisor data, a_V might reflect the star category of the hotels or the ID of review authors, and a_E can indicate the awarded review rating. Dynamic SNs consist of a finite sequence of static SNs.

A bipartite SN $(V_1, V_2, E, a_{V_1}, a_{V_2}, a_E)$ admits relationships only between two disjoint groups of actors, V_1

ITAT'23: Information technologies – Applications and Theory, Slovakia
iveta.mrazova@mff.cuni.cz (I. Mrázová);
behunmarek@gmail.com (M. Behún)
0000-0002-3765-1400 (I. Mrázová)

and V_2. a_{V_1} and a_{V_2} are the actor attributes, a_E specifies the edge attributes. The actual role of an actor is often related to the connectivity structure of the whole SN. E.g., the so-called *eigenvector centrality* [1] sums the importance of the neighbors of an actor with a damping factor $C_E(v) = \lambda^{-1} \sum_{u \in \mathrm{ne}(v)} C_E(u)$.

Real-world SNs form massive graphs with millions of nodes and edges [2] that would require full-batch training for a traditional memory setting comprising the entire graph. [17] introduced the graph neural network model (GNN) to facilitate learning on graphs. GNNs capture mutual dependencies of graph nodes via message passing. [23] proposes a general pipeline design for GNN models and reviews state-of-the-art methods relevant to GNNs and their applications. A comprehensive survey on recent GNN models can be found in [22].

A generalization of the efficient convolutional neural network (CNN) like information processing to graphs involves the so-called *graph Fourier transform* and is principal to constructing graph convolutions [3]. To produce networks universal to any graph structure, the ChebNet model uses Chebyshev polynomials of order K to determine the filters on graphs [5].

Graph Convolutional Networks (GCNs) [11] represent a first-order approximation to ChebNets. Yet, due to vanishing gradients, GCNs are limited to shallow models (3 or 4) layers. To boost the scalability of GNNs, the GraphSAGE model (SAmple and aggreGatE) [8] introduces a recursive node-wise sampling. Recursive neighborhoods may, on the other hand, impact exponential memory complexity that grows with the number of GNN layers.

The Graph Attention Networks (GATs) [21] further learn to pay different attention to the respective node's neighbors. The Graph Autoencoders (GAEs) [12] leverage the GCNs for encoding with ReLU used as an activation function. Decoding may be implemented as an inner product of the node embeddings. GCRN models combine the ChebNet convolution with LSTM or GRU units to process temporal SN data [18].

2.1. The UMAP Visualisation Technique

To grasp how the GNNs represent the high-dimensional graph data, we will use the so-called *Uniform Manifold Approximation and Projection* (UMAP) visualization technique [13] that maps n originally high-dimensional data points $\mathbf{x}_1 \ldots \mathbf{x}_n \in \mathbb{R}^f$ to lower-dimensional (usually 2D) data points $\mathbf{y}_1 \ldots \mathbf{y}_n \in \mathbb{R}^{f'}$ in a non-linear way. In this paragraph, $\mathbf{X} = (\mathbf{x}_1 \ldots \mathbf{x}_n)^T \in \mathbb{R}^{n \times f}$ represents the high-dimensional data point matrix and $\mathbf{Y} = (\mathbf{y}_1 \ldots \mathbf{y}_n)^T \in \mathbb{R}^{n \times f'}$ the data point matrix after dimensionality reduction.

Neighbor similarities visualized in the low-dimensional space should match the similarities existing for the high-dimensional data points as closely as possible. Dimensionality reduction techniques adjust the position of low-dimensional data points through gradient descent to comply with this effort. The loss function evaluates the discrepancy between high- and low-dimensional neighbor similarities.

To visualize of the data points, UMAP considers the neighbor similarities $p_{j|i}$ only for k nearest neighbors of $\mathbf{x}_i$. For $i \neq j$, we define the conditional neighbor similarity of $\mathbf{x}_j$ to $\mathbf{x}_i$ ($\sigma_i > 0; 1 \leq i \leq n$) as:

$$p_{j|i} = \begin{cases} \exp\left(\frac{-(d(\mathbf{x}_i,\mathbf{x}_j)-\rho_i)}{\sigma_i}\right) & \text{if } d(\mathbf{x}_i,\mathbf{x}_j) \geq \rho_i, \\ 1 & \text{otherwise,} \end{cases}$$

with a (not necessarily Euclidean) metric $d(\cdot,\cdot)$ in the high-dimensional space and ρ_i being the distance to the nearest neighbor in this metric. The symmetrized neighbor similarities p_{ij} are then determined as $p_{ij} = p_{i|j} + p_{j|i} - p_{i|j} \cdot p_{j|i}$. The formula for differentiable low-dimensional similarities q_{ij} has the form of:

$$q_{ij} = \left(1 + a \left\|\mathbf{y}_i - \mathbf{y}_j\right\|^{2b}\right)^{-1}$$

with the parameters a and b to be fitted by a non-linear least squares method.

Before the gradient descent like adjustment of low-dimensional data points, a weighted graph $G_{\mathbf{X}} = (\mathbf{X}, E_{\mathbf{X}}, \mathbf{P})$ is constructed for the high-dimensional data point matrix $\mathbf{X}$ with the weights p_{ij}. The so-called *spectral embedding method* initializes the low-dimensional data point matrix $\mathbf{Y}$. UMAP uses the loss function $\mathcal{L}$:

$$\mathcal{L} = - \sum_{\{i,j\} \in E_{\mathbf{X}}} \left(p_{ij} \log q_{ij} + (1 - p_{ij}) \log (1 - q_{ij})\right).$$

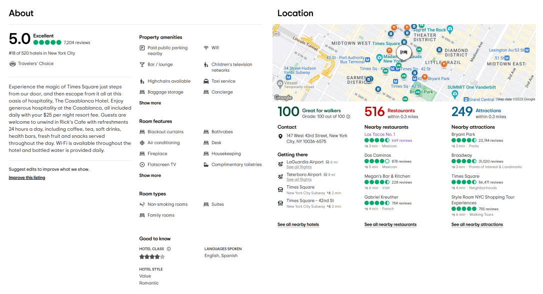
For performance reasons, UMAP uses stochastic gradient descent with negative sampling to optimize $\mathcal{L}$. In each iteration, the algorithm updates the low-dimensional points $\mathbf{y}_i$ by the gradient of $\log q_{ij}$ and randomly selects several $\mathbf{y}_k$ as negative samples and updates $\mathbf{y}_i$ by the gradient of $\log(1 - q_{ik})$.

3. Data Source

The Tripadvisor [19] portal provides helpful information on travel destinations, such as descriptions and photos of the hotels and their prices, availability, and booking options. It is also a popular travel review website that offers millions of reviews and ratings from travelers worldwide. Figures 1 and 2 illustrate the type of data available on Tripadvisor.

3.1. Data Acquisition

On the Tripadvisor platform, each served HTML document contains the definition of a JavaScript Object Notation (JSON) object that contains all the information



About

5.0 Excellent 7,204 reviews

#18 of 520 hotels in New York City

Travelers' Choice

Experience the magic of Times Square just steps from our door, and then escape from it all at this oasis of hospitality, The Casablanca Hotel. Enjoy generous hospitality at the Casablanca, all included daily with your $25 per night resort fee. Guests are welcome to unwind in Rick's Cafe with refreshments 24 hours a day, including coffee, tea, soft drinks, health bars, fresh fruit and snacks served throughout the day. Wi-Fi is available throughout the hotel and bottled water is provided daily.

Suggest edits to improve what we show.
Improve this listing

Property amenities

Paid public parking nearby
Wifi
Bar / lounge
Children's television networks
Highchairs available
Taxi service
Baggage storage
Concierge

Show more

Room features

Blackout curtains
Bathrobes
Air conditioning
Desk
Fireplace
Housekeeping
Flatscreen TV
Complimentary toiletries

Show more

Room types

Non-smoking rooms
Suites
Family rooms

Good to know

HOTEL CLASS
★★★★

LANGUAGES SPOKEN
English, Spanish

HOTEL STYLE
Value
Romantic

Location

100 Great for walkers
Grade: 100 out of 100

516 Restaurants
within 0.3 miles

249 Attractions
within 0.3 miles

Contact
147 West 43rd Street, New York City, NY 10036-6575

Getting there
LaGuardia Airport 6 mi
See all flights
Teterboro Airport 8 mi
See all flights
Times Square
New York City Subway 2 min
Times Square - 42nd St
New York City Subway 2 min

See all nearby hotels

Nearby restaurants
Los Tacos No. 1
669 reviews
3 min · Mexican
Dos Caminos
878 reviews
5 min · Mexican
Megan's Bar & Kitchen
228 reviews
6 min · Irish
Gabriel Kreuther
754 reviews
4 min · French

See all nearby restaurants

Nearby attractions
Bryant Park
23,744 reviews
3 min · Parks
Broadway
31,020 reviews
3 min · Points of Interest & Landmarks
Times Square
56,471 reviews
4 min · Neighborhoods
Style Room NYC Shopping Tour Experiences
755 reviews
6 min · Walking Tours

See all nearby attractions

Figure 2: Screenshot of the About section for the Casablanca Hotel in New York City from Tripadvisor [19].

Table 1
Left: information about the scraped data (r is # reviews). Right: number of authors who provided 1 to 7 reviews (r).

quantity	value
# of hotels	3,260
# of reviews	3,125,631
# of authors	2,296,247
avg r per author	1.36
max r per author	134
avg r per hotel	958.78
max r per hotel	19,534

r	# authors
1	1,895,027
2	239,682
3	77,155
4	34,374
5	17,649
6	10,247
7	6,328

displayed on the web page in a structured format. The data scraping utility we have developed for this study[1] stores the scraped information in one of the respective JSON files listed below:

- `hotels.json` contains information about the reviewed hotels, i.e., the name, description, location, contact information, star rating, a list of languages the staff at the reception desk is able to speak, and a list of amenities/perks (e.g., a non-smoking hotel, WiFi, laundry service),
- `authors.json` embraces review author fields, such as ID, username, displayed name, home town, and home town ID, and
- `reviews.json` comprises the reviews themselves inclusive the review ratings, dates of stay, writing and publication, the type of the trip (family, couples, friends, business, solo, or none of them), review title, text, and a potential room tip.

After running the scraping utility for several days nonstop (without triggering any Denial of Service protection), we have acquired records of 3,125,631 Tripadvisor reviews for 3,260 hotels from 2,296,247 unique authors, see Table 1. Yet the scraped dataset contains only hotel reviews from several US cities. Furthermore, the Covid pandemic of the last few years seriously disturbed traveling globally. Both these facts may have introduced a specific bias into the studied data. Overall, the scraped data elucidates two intriguing observations:

- The number of submitted reviews per year peaked in 2016 and began to fall afterward.
- In 2020, the number of reviews fell even more due to the Covid pandemic but resumed growth in 2021 and 2022.

After data acquisition, the next step is to transform the data so that each entry is flattened and contains only the information we will use for our experiments. Preprocessing decreases the size of the dataset JSON files significantly and involves:

- **Flattening.** Both hotel and review records contain normalized information: hotels list their amenities and languages, while reviews contain lists of ratings. All these fields are flattened to facilitate deep learning computations with tensors.
- **Filtering.** We remove redundant textual information from the records, such as names, descriptions, URLs, and addresses. Similarly, we consider just a pre-selected number of the most frequent amenities and languages the scraped hotels provide. For the actual experiment settings, refer to Section 4.

3.2. Graph construction

Based on the preprocessed dataset, we can build a bipartite social network $G = (A, H, R, a_A, a_H, r)$, where:

- A is the set of review authors,
- H is the set of reviewed hotels,
- $R \subseteq A \times H$ is the set of reviews—author-hotel pairs associated with a rating,
- a_A and a_H are author and hotel attribute functions that assign to each of the respective author or hotel nodes an attribute value, e.g., author ID or eigenvector centrality (for further examples, refer to Section 4),
- $r : R \to \{1, 2, 3, 4, 5\}^k$ is the edge attribute function—a function that maps author-hotel pairs to rating values: $r(a, h) = v$ means that author a gave rating v to hotel h.

Because of limited cluster machine memory, the dataset preprocessing utility allows to specify the minimum number of reviews for each hotel m_h and author m_a. Only those authors and hotels fulfilling this requirement will be kept. The resulting graph will be called the *Filtered Review Graph*. Figure 3 illustrates the process.

[1]The scraped data and the entire code we wrote for the study experiments are publicly available at https://github.com/elkablo/gnn-social-tripadvisor.

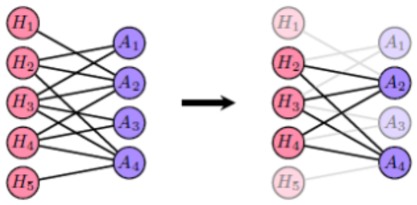


Figure 3: An illustration for the process of review graph filtering with $m_a = 3$ and $m_h = 2$. Pink nodes represent hotels, and violet nodes stand for review authors.

Definition 1 (Filtered Review Graph).
Let $G = (A, H, R, a_A, a_H, r)$ be a bipartite social network of authors A, hotels H and reviews R, further let $m_a, m_h \in \mathbb{N}$. The filtered review graph *of G with parameters m_a, m_h is the maximum subnetwork $G_f = (A_f \cup H_f, R_f, a_{A_f}, a_{H_f}, r_f); A_f \subseteq A, H_f \subseteq H$, and $E_f \subseteq E$; $\forall a \in A_f : \deg(a) \geq m_a$ and $\forall h \in H_f : \deg(h) \geq m_h$.*

We need to work with monopartite networks when analyzing the eigenvector centralities of the hotels and authors. However, we will use the so-called *bipartite network projections* in such a case. We create a network of authors, where each author is connected to another if they have reviewed the same hotel. The strength of such an association grows with more reviews on hotels visited by both authors. For an illustration of a projection, see Figure 4. A monopartite network for the hotels can be created in a similar way.

Definition 2 (Projection to Authors and Hotels).
Let $G = (A, H, R, a_A, a_H, r)$ be a bipartite SN of authors A, hotels H, and reviews R, and let $m_r, m_c, m_n \in \mathbb{N}$ be the minimum number of reviews, common associations, and neighbors, respectively.

The projection G_A of G to the authors A with the parameters m_r, m_c, m_n is constructed by:

- *first removing from G all author nodes $v \in A$ with $\deg(v) < m_r$,*
- *then constructing the bipartite network projection $\widetilde{G_A} = (A, E_A, a_A, a_{E_A})$ of G to A with the set of edges between the authors $E_A \subseteq A \times A$ and the author edge attribute function a_{E_A} given by $a_{E_A}((a_i, a_j)) = |h|$ if $\exists h \in H | ((a_i, h) \in R \wedge (a_j, h) \in R)$, and $a_{E_A}((a_i, a_j)) = 0$ otherwise.*
- *then removing all edges $e \in E_A$ from $\widetilde{G_A}$ with $a_{E_A}(e) < m_c$ (so that only edges representing at least m_c common associations are kept),*
- *then successively removing from $\widetilde{G_A}$ all nodes with fewer than m_n neighbors,*

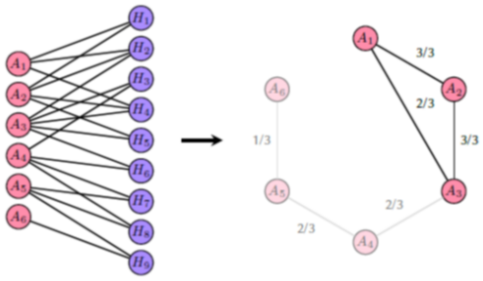


Figure 4: An illustration of projecting the bipartite social network to authors with the parameters $m_r = 2, m_c = 2, m_n = 2$. Left: the original bipartite review graph. Right: its projection to the authors where, e.g., the link between A_1 and A_3 has strength $2/3$ because there are two common associations between A_1 and A_3 in the original network (through H_2 and H_4) and the maximum number of common associations for any pair in this graph is 3. Authors A_4, A_5, and A_6 are successively removed from the graph because A_6 submitted only one review, A_5 has only one neighbor after the removal of A_6, and A_4 has only one neighbor after the removal of A_5.

- *finally normalizing the edge attributes in $\widetilde{G}$ by setting them to a'_E, with*

$$a'_E(e) = \frac{a_{E_A}(e)}{\max_{e \in E_A} a_{E_A(e)}}.$$

The projection to hotels $G_H = (A, E_H, a_A, a_{E_H})$ is constructed in an analogous way (just swapping hotels and authors in the above definition).

In the last experiment, we will aim at predicting hotel eigenvector centrality scores as they change over time. Therefore, we will work with a dynamic SN of hotels called temporal projection.

Definition 3 (Temporal Projection).
Let $G = (A, H, R, a_A, a_H, r)$. Further, let $\mathcal{T} = \{t_0, t_1, t_2, \ldots, t_n \,|\, t_0 < t_1 < t_2 < \cdots < t_n\}$ be a partition of the original time span for the reviews available in G. Let G_{t_k} be the maximum subnetwork of G, which contains only those reviews written at time $t; t_{k-1} < t \leq t_k$ and where each author and hotel have at least one neighbor.

The temporal projection of G to the authors A with the parameters $\mathcal{T}, m_r, m_c, m_n$ is the sequence of monopartite networks $\mathcal{G}_A^{\mathcal{T}} = (G_{A,t_k})_{k=1}^{n}$, where G_{A,t_k} is the projection of the network G_{t_k} to the authors A with the parameters m_r, m_c, m_n.

The temporal projection of G to the hotels H, $\mathcal{G}_H^{\mathcal{T}}$, is created analogously, just by using the projection to the hotels instead of the authors.

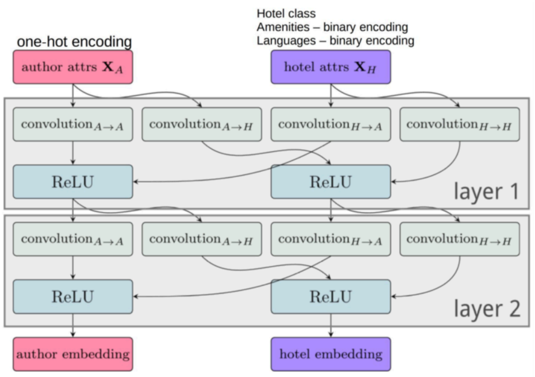


Figure 5: An illustration of a 2-layer heterogeneous GNN with convolutional GNN layers for the review rating prediction task.

4. Supporting Experiments

In our experiments, we worked with a bipartite SN $G = (A, H, R, a_A, a_H, r)$ and used a filtered review graph with $m_a = m_h = 12$. The resulting preprocessed dataset contained 76,692 reviews of 1,287 hotels from 4,404 authors. While the author features, $\mathbf{X}_A$, correspond to one-hot encodings of author IDs, the hotel feature matrix $\mathbf{X}_H$ used for this task contains:

- hotel class,
- existence of hotel website (binary),
- presence of 15 most popular amenities (binary),
- availability of 5 most popular languages (binary),
- optionally, other information like the eigenvector centrality or hotel-hotel links used by the projection to the hotels was also included as a weighing factor in some models.

As the filtered review graphs contain two distinct types of nodes, we will employ the *heterogeneous graph transform* [9]. See Figure 5 for an illustration of the bipartite author-hotel-review social network and its encoding, .

Definition 4 (Heterogeneous GNN). *Let*

- $\mathbf{A} \in \{0,1\}^{|A|\times|H|}$ *be the adjacency matrix of reviews written by the authors for the hotels,*
- $\mathbf{X}^{\mathbf{e}} \in \mathbb{R}^{d_R\times|R|}$ *be the author-hotel edge feature matrix,* [2] d_R *being the dimensionality of review embeddings,*
- $\mathbf{W}_H \in \mathbb{R}^{|H|\times|H|}$ *and* $\mathbf{W}_A \in \mathbb{R}^{|A|\times|A|}$ *be the weighted adjacency matrices of the projections to hotels and authors,*
- $\mathbf{H}_H^{(l)} \in \mathbb{R}^{d_H^{(l)}\times|H|}$ *and* $\mathbf{H}_A^{(l)} \in \mathbb{R}^{d_A^{(l)}\times|A|}$ *be the current hidden state matrices of the hotel and author nodes,*

[2]The review feature matrix $\mathbf{X}^{\mathbf{e}}$ is not used in the review rating prediction task, but it is used in hotel class prediction.

- $d_R, d_H^{(l)}, d_A^{(l)} \in \mathbb{N}$ *be the dimensionalities of the review, hotel and author input embeddings for layer* l*, while* $d_H^{(l+1)}, d_A^{(l+1)} \in \mathbb{N}$ *be the corresponding dimensionalities of output embeddings for layer* l*,*
- $\mathrm{F}_{A\to H} : \mathbb{R}^{d_A^{(l)}\times|A|} \times \{0,1\}^{|A|\times|H|} \to \mathbb{R}^{d_H^{(l+1)}\times|H|}$ *be a graph neural function that computes the embeddings of hotel nodes based on the current activation states of author nodes* $\mathbf{H}_A^{(l)}$ *and the adjacency matrix* $\mathbf{A}$*,*
- $\mathrm{F}_{H\to A} : \mathbb{R}^{d_H^{(l)}\times|H|} \times \{0,1\}^{|A|\times|H|} \to \mathbb{R}^{d_A^{(l+1)}\times|A|}$ *be a graph neural function that computes the embeddings of author nodes from the current activation states of hotel nodes* $\mathbf{H}_H^{(l)}$ *and the adjacency matrix* $\mathbf{A}$*,*
- $\mathrm{F}_{H\to H} : \mathbb{R}^{d_H^{(l)}\times|H|} \times \mathbb{R}^{|H|\times|H|} \to \mathbb{R}^{d_H^{(l+1)}\times|H|}$ *be a graph neural function that computes the embeddings of hotel nodes from their hidden states* $\mathbf{H}_H^{(l)}$ *and the weighted adjacency matrix* $\mathbf{W}_H$*,*
- $\mathrm{F}_{A\to A} : \mathbb{R}^{d_A^{(l)}\times|A|} \times \mathbb{R}^{|A|\times|A|} \to \mathbb{R}^{d_A^{(l+1)}\times|A|}$ *be a graph neural function that computes the embeddings of author nodes from their hidden states* $\mathbf{H}_A^{(l)}$ *and the weighted adjacency matrix* $\mathbf{W}_A$*.*
- *We can use as a graph neural function, e.g.,* SAGE, GAT, GCN, ChebNet, LSTM *or* GRU*, among others.*

The (sum-aggregating) heterogeneous graph layer HetLayer *computes the next hidden activation states of author and hotel nodes* $\left(\mathbf{H}_H^{(l+1)}, \mathbf{H}_A^{(l+1)}\right) \in \mathbb{R}^{d_H^{(l+1)}\times|H|} \times \mathbb{R}^{d_A^{(l+1)}\times|A|}$ *as:*

$$
\begin{aligned}
\mathbf{H}_H^{(l+1)} &= \mathrm{ReLU}\left(\mathrm{F}_{A\to H}\left(\mathbf{H}_A^{(l)}, \mathbf{A}, \mathbf{X}^{\mathbf{e}}\right)\right) + \\
&\quad + \mathrm{ReLU}\left(\mathrm{F}_{H\to H}\left(\mathbf{H}_H^{(l)}, \mathbf{W}_H, \mathbf{X}^{\mathbf{e}}\right)\right) \\
\mathbf{H}_A^{(l+1)} &= \mathrm{ReLU}\left(\mathrm{F}_{H\to A}\left(\mathbf{H}_H^{(l)}, \mathbf{A}\right)\right) + \\
&\quad + \mathrm{ReLU}\left(\mathrm{F}_{A\to A}\left(\mathbf{H}_A^{(l)}, \mathbf{W}_A\right)\right).
\end{aligned}
$$

Further, let $\mathbf{X}_H, \mathbf{X}_A$ *be the hotel and author feature matrices,* $L \in \mathbb{N}$ *be the number of layers, and let* $\mathrm{HetLayer}_l$ *be a heterogeneous graph layer constructed based on the given* $\mathrm{F}_{*\to *}$ *functions for every* $l \in \{1, \ldots, L\}$*.*

The heterogeneous graph neural network HetGNN *with layers* $\mathrm{HetLayer}_l$ *is then defined as:*

$$
\mathrm{HetGNN}\left(\mathbf{X}_H, \mathbf{X}_A, \mathbf{A}, \mathbf{W}_H, \mathbf{W}_A\right) = \left(\mathbf{H}_H^{(L)}, \mathbf{H}_A^{(L)}\right).
$$

$\left(\mathbf{H}_H^{(l)}, \mathbf{H}_A^{(l)}\right)$ *is set to* $(\mathbf{X}_H, \mathbf{X}_A)$ *if* $l = 0$ *and to* $\mathrm{HetLayer}_l\left(\mathbf{H}_H^{(l)}, \mathbf{H}_A^{(l)}, \mathbf{A}, \mathbf{W}_H, \mathbf{W}_A\right)$ *otherwise.*

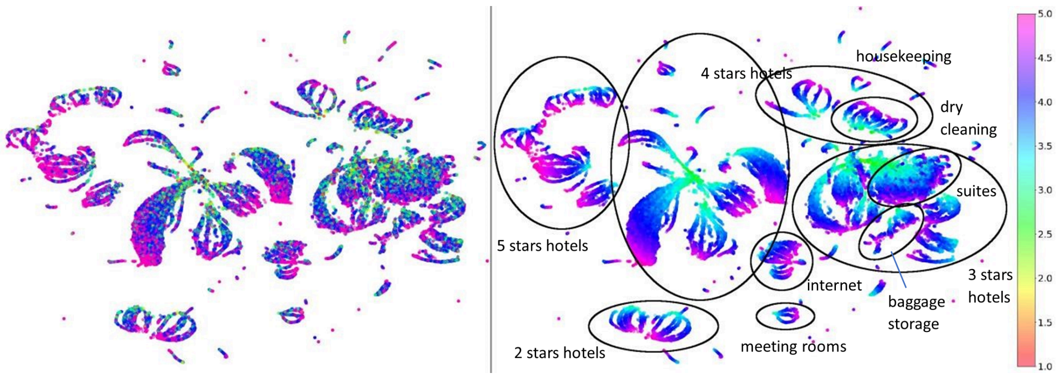


Figure 6: UMAP embeddings of the reviews from the hidden layer of the best-performing GNN model used for the review rating prediction colored by rating (left: true; right: predicted). The right-hand side also depicts the regions of various hotel attributes internally detected by the network.

We programmed the SW[3] for the experiments mainly in Python 3 [20]. The implemented command-line interface (CLI) utilities benefit from the PyTorch Geometric [6], PyTorch Geometric Temporal [16] and NetworkX [7] libraries.

We applied the 10-fold cross-validation for training and testing.[4] The Adam method [10] helped optimize the models, with the parameters $\beta_1 = 0.9, \beta_2 = 0.999$ and with learning rates 0.1% for 800 epochs. The mean squared error (MSE) between the true $\mathbf{x}$ and predicted $\mathbf{y}$ ratings $\mathrm{MSE}(\mathbf{x}, \mathbf{y}) = \textsc{mean}\left(\sum_i \left((\mathbf{x})_i - (\mathbf{y})_i\right)^2\right)$ represented the loss function. We report on the results referring to the root mean squared error $\mathrm{RMSE}(\mathbf{x}, \mathbf{y}) = \sqrt{\mathrm{MSE}(\mathbf{x}, \mathbf{y})}$.

4.1. Review Rating Prediction

Each review incorporates several ratings from 1 to 5 that the evaluation author awarded to the reviewed hotel. Missing rating values for the author-hotel pairs naturally raise the question of a possible review rating prediction. Formally, we will work with a bipartite SN $G = (A, H, R, a_A, a_H, r)$, and our objective will be to extend the original domain of the edge attribute function r to the entire set of possible edges $A \times H$, by leveraging the information given by the attribute functions r, a_A, and a_H.

In the context of GNNs, review rating prediction epitomizes a *link label prediction problem* (i.e., predict the overall hotel rating r; the per-item ratings remain ignored) and belongs to the area of recommender systems. The generic definition for the review rating prediction models is thus:

Definition 5 (Review Rating Prediction Model). *Let* HetGNN *be a heterogeneous graph neural network constructed for* $\mathbf{X}_H, \mathbf{X}_A, \mathbf{A}, \mathbf{W}_H$, *and* $\mathbf{W}_A$. *Then, the review rating prediction model is a graph autoencoder network model with the encoder* ENC *defined as:* $\textsc{enc}(\mathbf{X}_H, \mathbf{X}_A, \mathbf{A}, \mathbf{W}_H, \mathbf{W}_A) = \mathrm{HetGNN}(\mathbf{X}_H, \mathbf{X}_A, \mathbf{A}, \mathbf{W}_H, \mathbf{W}_A) = (\mathbf{H}_H, \mathbf{H}_A)$; $(\mathbf{H}_H, \mathbf{H}_A) \in \mathbb{R}^{d_H \times |H|} \times \mathbb{R}^{d_A \times |A|}$. *The decoder* DEC *is defined for hotel* $h \in H$ *embedding* $\mathbf{h}_h = (\mathbf{H}_H)_h \in \mathbb{R}^{d_H}$ *and author* $a \in A$ *embedding* $\mathbf{h}_a = (\mathbf{H}_A)_a \in \mathbb{R}^{d_A}$, *as:*

$$\textsc{dec}(\mathbf{h}_h, \mathbf{h}_a) = \mathbf{\Theta}_2 \,\mathrm{ReLU}\left(\mathbf{\Theta_1}\begin{pmatrix}\mathbf{h}_h \\ \mathbf{h}_a\end{pmatrix} + \mathbf{b}_1\right) + \mathbf{b}_2,$$

where $\mathbf{\Theta_1} \in \mathbb{R}^{d_1 \times (d_H + d_A)}, \mathbf{b}_1 \in \mathbb{R}^{d_1}$ *and* $\mathbf{\Theta}_2 \in \mathbb{R}^{d_2 \times d_1}, \mathbf{b}_2 \in \mathbb{R}^{d_2}$ *represent two fully connected linear layers with* $d_1, d_2 \in \mathbb{N}$ *being the dimensionalities of the edge embeddings produced by these layers.*

Based on the pair of hotel and author embeddings, the decoder defined in this way produces a review rating prediction of dimensionality d_2.

According to the above generic definition, we have built review rating prediction models with the parameters of the following type and range:

- the rating prediction dimensionality d_2 is always 1 (we predict the overall rating only),
- the number of layers L ranged through values 1, 2, and 3 (higher values led to worse performance),
- the number of hidden channels (the dimensionality of node embeddings) ranged through the values 4, 8, 12, and 16 (higher values yielded worse performance), and remained the same for all layers ($d = d^{(l)}$ for all $l \in \{1, \ldots, L\}$),

[3]https://github.com/elkablo/gnn-social-tripadvisor.
[4]Computational resources were provided by MetaCentrum NGI (https://www.metacentrum.cz/en) under the e-INFRA CZ project (ID:90254) supported by the Ministry of Education, Youth, and Sports of the Czech Republic.

- the same heterogeneous graph neural function $\mathrm{F}_{\text{hetero}}$ was used for both $\mathrm{F}_{A\to H}$ and $\mathrm{F}_{H\to A}$; its possible variants ranged through the SAGE, GAT, and GNN* models,
- the same homogeneous graph neural function F_{homo} was used for both $\mathrm{F}_{A\to A}, \mathrm{F}_{H\to H}$. Possible variants ranged through the same functions as for the heterogeneous case and comprised also GCN and $\mathrm{ChebNet}_{K=2}$. Models without homogeneous edges were, however, tested, too.

In the tests, the graph autoencoder with 12 hidden channels for one hidden layer with SAGE used for $\mathrm{F}_{\text{hetero}}$ and GAT for F_{homo} provided the best results ($\mathrm{RMSE} = 0.8538$). Table 2 compares the performance of the best five models. Overall, training was relatively rapid (about 30 s), with the actual ratings shifted on average by 0.8 from the predicted ones. More hidden channels arranged in fewer layers seem to provide better results for the review rating prediction task. While all the reported models use the SAGE layer for heterogeneous author-review message passing, the GAT layer leads to slightly better results when used for the homogeneous links.

Figure 6 shows the UMAP visualization of the processed data representations developed in the hidden layer of the best-performing model. The colors in the picture indicate the obtained ratings. The visualization confirms an adequate knowledge extraction capability of the trained network.

4.2. Hotel Class Prediction

In the hotel class prediction task, our objective is to estimate the hotel class attribute (i.e., the number of stars as one of 9 possible values $1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5$) by considering the information from the other attributes. In the context of GNNs, we call this task a *node classification problem.* In Figure 8, the hotel class attribute value we want to predict is depicted by the red "?". Potential use cases for a model predicting the hotel class include, e.g., the detection of fake or misleading information about hotels submitted by their owners and the assessment of hotel class where we lack this information. Alternatively, the model issues another score for users considering the hotel for accommodation.

The methodology for the hotel class prediction experiment remains the same as for review rating prediction. Formally, we will work with the same bipartite SN $G = (A, H, R, a_A, a_H, r)$ and preprocess the data accordingly, except for the hotel feature matrix $\mathbf{X}_H$ that does not contain the hotel class now since this will be the target attribute. We set $m_a = m_h = 12$ for review graph filtering and augment the bipartite network with author-author and hotel-hotel edges. The models for hotel class prediction have the following form:

Definition 6 (Hotel Class Prediction Model). *Let* HetGNN *be a heterogeneous graph neural network constructed for* $\mathbf{X}_A, \mathbf{A}, \mathbf{W}_H, \mathbf{W}_A$. *Further, let* $\mathbf{X}_H$ *be the hotel feature matrix without hotel class, let* $\mathbf{X}^{\mathbf{e}}$ *be the author-hotel edge feature matrix. The hotel class prediction model is then defined as:* $\textsc{HotelClassPred}\left(\mathbf{X}_H, \mathbf{X}_A, \mathbf{X}^{\mathbf{e}}, \mathbf{A}, \mathbf{W}_H, \mathbf{W}_A\right) = \mathbf{\Theta}\mathbf{H}_H^{(L)} + \mathbf{b}$. $\mathbf{H}_H^{(L)}$ *can be obtained from* $\left(\mathbf{H}_H^{(L)}, \mathbf{H}_A^{(L)}\right) = \mathrm{HetGNN}\left(\mathbf{X}_H, \mathbf{X}_A, \mathbf{X}^{\mathbf{e}}, \mathbf{A}, \mathbf{W}_H, \mathbf{W}_A\right)$. $\mathbf{\Theta} \in \mathbb{R}^{d\times d_H}$ *and* $\mathbf{b} \in \mathbb{R}^d$ *represent one fully connected linear layer with* $d = 1$ *for the resulting node embedding.*

The other parameters (the number of hidden layers, hidden channels, and the homogeneous and heterogeneous graph neural functions) range over the same values as in the previous experiment. Again, the training of most models was fast (under 30 s).

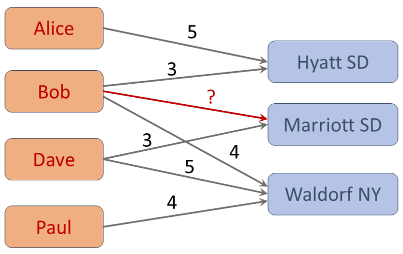


Figure 7: An illustrative example for the review rating prediction task: Alice, Bob, Dave, and Paul provided six ratings for three hotels. The rating Bob would assign to Marriott SD is unknown (depicted by the red "?" sign) and has to be predicted based on the other known attribute values.

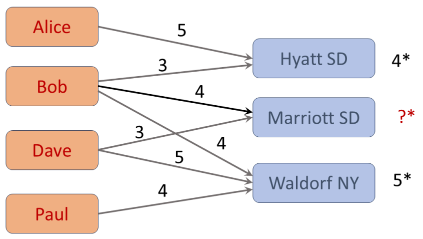


Figure 8: An illustrative example for the hotel class prediction task: Alice, Bob, Dave, and Paul provided seven ratings for three hotels. We know the classes for the hotels Hyatt SD and Waldorf NY. The hotel class for Marriott SD is unknown (depicted by the red "?" sign in the figure) and has to be predicted based on the other known attribute values.

Table 2
Comparison of the five best-performing models trained to solve the respective task.

$\mathrm{F}_{\mathrm{hetero}}$ *[GCRN type]*	$\mathrm{F}_{\mathrm{homo}}$	nr. of layers *[the order K]*	nr. of hidden channels	nr. of trainable parameters	mean train. time	RMSE with the 99% confidence interval
Graph autoencoder-based architectures for the review rating prediction task:[1]						
SAGE	GAT	1	12	106,861	27.32 s	0.8538 $\pm$ 0.0132
SAGE	GAT	1	16	142,609	27.87 s	0.8543 $\pm$ 0.0067
SAGE	SAGE	1	12	107,101	26.12 s	0.8577 $\pm$ 0.0103
SAGE	SAGE	1	16	142,929	26.29 s	0.8585 $\pm$ 0.0098
SAGE	SAGE	2	16	144,513	31.05 s	0.8585 $\pm$ 0.0139
Heterogeneous GNN-based architectures for the hotel class prediction task:[2]						
SAGE	ChebNet	2	12	107,653	24.47 s	0.4154 $\pm$ 0.0514
SAGE	GAT	2	12	107,305	24.27 s	0.4163 $\pm$ 0.0347
SAGE	GCN	2	12	107,257	20.85 s	0.4178 $\pm$ 0.0364
SAGE	GCN	2	8	71,345	20.81 s	0.4184 $\pm$ 0.0218
SAGE	SAGE	1	16	142,337	15.82 s	0.4198 $\pm$ 0.0185
GCRN-based architectures for the temporal hotel score prediction task:[3]						
[LSTM]		*[2]*	16	5,121	92.85 min	0.0020 $\pm$ 0.0009
[LSTM]		*[2]*	8	2,049	91.93 min	0.0027 $\pm$ 0.0014
[LSTM]		*[3]*	12	5,089	97.92 min	0.0028 $\pm$ 0.0016
[GRU]		*[3]*	16	5,585	55.23 min	0.0033 $\pm$ 0.0017
[GRU]		*[3]*	4	965	49.56 min	0.0018 $\pm$ 0.0054

[1] The times were obtained for training on NVIDIA GeForce RTX 2080 Ti GPU.
[2] The times were obtained for training on NVIDIA A100-SXM4-40GB GPU.
[3] The times are not comparable as 3 different GPUs were used for training.

In this experiment, the GNN with 12 hidden channels in two hidden layers with SAGE used for $\mathrm{F}_{\mathrm{hetero}}$ and $\mathrm{ChebNet}$ for $\mathrm{F}_{\mathrm{homo}}$ achieved the best results ($\mathrm{RMSE} = 0.4154$). In most cases, the class predicted by the model thus differed by at most one hotel class (considering its granularity of 0.5). Table 2 shows the performance of the best five models. Regarding accuracy, the SAGE graph neural function again outperformed other functions for the author-hotel heterogeneous links.

4.3. Hotel Score Prediction

The final experiment evaluates the chance of predicting how a hotel's score changes over time, where the score corresponds to the eigenvector centrality of the hotel in the projection to hotels. To solve the task, we create a temporal projection of the bipartite SN to hotels $\mathcal{G}_H^{\mathcal{T}}$ and then aim at predicting the eigenvector centrality based on the static hotel attribute function a_H and the dynamic edge weights in the temporal projection.

For this task, we divided the time interval with the scraped reviews (2003–2022) into monthly partitioning $\mathcal{T}$ and then created a temporal projection to the hotels with parameters $\mathcal{T}, m_r = 20, m_c = 3, m_n = 3$. From the resulting dynamic network, we removed the initial snapshots that contained less than 10,000 edges. In the last several years, the centralities stabilized. Therefore, we removed the last six years' data from the projected temporal network, too. The remaining 109 monthly snapshots were split into 80%–20% training-testing sequences (comprising 87 and 22 snapshots, resp.).

We initialized the network models with random weights using the same hotel feature matrix $\mathbf{X} = \mathbf{X}_H$ and Adam parameters like in review rating prediction. Concerning the temporal data, we ran the experiments five times for evaluation instead of using k-fold cross-validation. We trained each model for 3000 epochs with the MSE loss. For this task, we applied the GCRN-LSTM and GCRN-GRU models. In addition to the recurrent layer, the ReLU activation function and a linear transformation were used to generate score prediction.

Definition 7 (Hotel Score Prediction Model). *Let* GRCN *be either the* $\mathrm{GRCN}_{\mathrm{LSTM}}$ *or the* $\mathrm{GRCN}_{\mathrm{GRU}}$ *model,* $\mathbf{X}$ *be the hotel feature matrix and* $\mathbf{W}^{(t)}$ *be the weighted adjacency matrices of the temporal projection to hotels for* $t \in \{1, \ldots, T\}$*. The hotel score prediction model computes the* t*-th score prediction as*

$$\mathbf{\Theta}\,\mathrm{ReLU}\left(\mathbf{X}, \mathbf{W}^{(t)}, \mathbf{X}, \mathbf{H}^{(t)}\right) + \mathbf{b},$$

where $\mathbf{\Theta} \in \mathbb{R}^{d \times d_H}$ *and* $\mathbf{b} \in \mathbb{R}^d$ *form a fully connected linear layer with* $d \in \mathbb{N}$ *hidden states.* $\mathbf{H}^{(t)}$ *is initialized with* 0 *and adjusted by* $\mathbf{H}^{(t)} = \mathrm{GRCN}\left(\mathbf{X}, \mathbf{W}^{(t)}, \mathbf{H}^{(t-1)}\right)$.

We tested networks with the order of ChebNet's Chebyshev polynomial K set to 2 and 3 and the number of hidden channels ranging through 4, 8, 12, and 16. An LSTM-based GCRN model with 16 hidden channels yielded the most accurate predictions ($\mathrm{RMSE} = 0.0020$). Table 2 shows the results for the five best-performing models. Overall, LSTM-based models with higher values of K (facilitating information flow from further distances) achieved better performance. Unfortunately, considerable time costs accompany high accuracy (training often takes longer than 1 hour).

5. Conclusions

In this paper, we have explored the applicability of GNN models to the analysis of scraped Tripadvisor data. For our investigations, we scraped 3,125,631 Tripadvisor reviews for 3,260 hotels from 2,296,247 unique authors. The analyzed problems involved prediction of review ratings, assessment of the actual hotel (star) classes, and prediction of dynamic / temporal centrality scores of the hotels.

The performed experiments yield reliable results for recommender systems even without the information on textual or image-related parts of the reviews. The paper thus presents a proof of concept for boosting the performance of recommender systems with advanced AI techniques. Overall, a reasonably low number of wider hidden layers led to a better performance in achieving accuracy. Temporal models consumed, however, significantly more computational resources. The involved GNN models were able to extract adequate knowledge that requires non-trivial methods, e.g., UMAP, to be visualized in an easy-to-understand way.

To boost their accuracy, future models might embrace attributes extended, e.g., by word embeddings of the actual review texts (like in [4]) or by the information on close attractions or the quality of nearby restaurants. The provided SW could benefit from integrating a Neo4j graph database [14] as a core tool for dealing with SNs. Efficient training of deeper and temporal GNN models for dynamic SN data shall also represent a welcome addition.

References

[1] C.C. Aggarwal, "Data mining: the textbook", Springer, 2015.

[2] A.-L. Barabási and M. Pósfai, "Network Science", Cambridge University Press, 2016.

[3] J. Bruna, W. Zaremba, A. Szlam and Y. LeCun, "Spectral Networks and Locally Connected Networks on Graphs", ICLR, 2014, 14 p.

[4] D. Chitiz and A. Perlmuter, "Hotel Rating Prediction", url: https://github.com/doviec/TripAdvisor-Rating-Prediction. Accessed: 2023-07-12.

[5] M. Defferrard, X. Bresson and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering", NIPS, 2016, 9 p.

[6] M. Fey and J.E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric", ICLR, 2019, 9 p.

[7] A. Hagberg, P. Swart and D.S. Chult, "Exploring network structure, dynamics, and function using NetworkX", SciPy, 2008, 5 p.

[8] W.L. Hamilton, R. Ying and J. Leskovec, "Inductive Representation Learning on Large Graphs", NIPS, 2017, 11 p.

[9] Z. Hu, Y. Dong, K. Wang and Y. Sun, "Heterogeneous Graph Transformer", WWW, 2020, pp. 2704–2710.

[10] D.P. Kingma and J.L. Ba, "Adam: A Method for Stochastic Optimization", ICLR, 2015, 13 p.

[11] T.N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks", ICLR, 2017, 14 p.

[12] T.N. Kipf and M. Welling, "Variational Graph Auto-Encoders", 2016, doi: 10.48550/ARXIV.1611.07308.

[13] L. McInnes, J. Healy and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction", 2018, doi: 10.48550/ARXIV.1802.03426.

[14] Neo4j, https://neo4j.com, Accessed: 2023-08-19.

[15] F. Ricci, L. Rokach and B. Shapira (eds.), "Recommender Systems Handbook (3 ed.), Springer, 2022.

[16] B. Rozemberczki et al., "PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models", CIKM, 2021, pp. 4564–4573.

[17] F. Scarselli, M. Gori, A.Ch. Tsoi, M. Hagenbucher and G. Monfardini, "The Graph Neural Network Model", IEEE Transactions on Neural Networks, vol. 20, no. 1, 2009, pp. 61–80.

[18] Y. Seo, M. Defferrard, P. Vandergheynst and X. Bresson, "Structured Sequence Modeling with Graph Convolutional Recurrent Networks", ICONIP, 2018, pp. 362-373.

[19] Tripadvisor, https://www.tripadvisor.com, Accessed: 2022-12-19.

[20] G. Van Rossum and F. L. Drake, "Python 3 Reference Manual", Scotts Valley, CA: CreateSpace, 2009.

[21] P. Veličković et al., "Graph Attention Networks", ICLR, 2018, 12 p.

[22] Z. Wu et al., "A Comprehensive Survey on Graph Neural Networks", IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 1, 2021, pp. 4-24.

[23] J. Zhou et al., "Graph neural networks: A review of methods and applications", AI Open, vol. 1, 2020, pp. 57-81.