http://dx.doi.org/10.62762/TETAI.2025.740330

REVIEW ARTICLE

Exploring Graph-Based Techniques in Text Data Processing: A Comprehensive Survey of NLP Advancements

Tuğba Çelikten[1,2,*] **and Aytuğ Onan**[3]

[1] Graduate School of Natural and Applied Sciences, Izmir Katip Çelebi University, Izmir 35620, Turkey
[2] Department of Software Engineering, Faculty of Technology, Manisa Celal Bayar University, Manisa 45140, Turkey
[3] Department of Computer Engineering, Izmir Institute of Technology, Izmir 35430, Turkey

Abstract

Graph Neural Networks (GNNs) have become increasingly prominent in Natural Language Processing (NLP) due to their ability to model intricate relationships and contextual connections between texts. Unlike traditional NLP methods, which typically process text linearly, GNNs utilize graph structures to represent the complex relationships between texts more effectively. This capability has led to significant advancements in various NLP applications, such as social media interaction analysis, sentiment analysis, text classification, and information extraction. Notably, GNNs excel in scenarios with limited labeled data, often outperforming traditional approaches by providing deeper, context-aware solutions. Their versatility in handling different data types has made GNNs a popular choice in NLP research. In this study, we thoroughly explored the application of GNNs across various NLP tasks, demonstrating their advantages in understanding and representing text relationships. We also examined how GNNs address traditional NLP challenges, showcasing their potential to deliver more meaningful and accurate results. Our research underscores the value of GNNs as a potent tool in NLP and suggests future research directions to enhance their applicability and effectiveness further.

Keywords: graph neural networks (GNN), natural language processing (NLP), graph convolutional networks (GCN), heterogeneous graph neural networks, GNN-based NLP.

1 Introduction

Natural language processing (NLP), a subfield of artificial intelligence (AI), aims to provide more intuitive and human-like communication between humans and computers. NLP involves developing AI models or algorithms that can understand natural language, process it quickly, and produce similar texts. NLP enables many practical applications, such as understanding the meaning of texts, determining emotional tone, translating text written in one

Submitted: 22 November 2025
Accepted: 02 December 2025
Published: 18 February 2026

Vol. 3, **No.** 2, 2026.
10.62762/TETAI.2025.740330

***Corresponding author:**
✉ Tuğba Çelikten
tugba.celikten@cbu.edu.tr

Citation
Çelikten, T., & Onan, A. (2026). Exploring Graph-Based Techniques in Text Data Processing: A Comprehensive Survey of NLP Advancements. *ICCK Transactions on Emerging Topics in Artificial Intelligence*, 3(2), 86–127.

language into another, or answering user questions. In this respect, it has the potential to increase efficiency in the industry and create new opportunities [1]. Research in the field of NLP is undergoing rapid development with innovations and developments in AI models.

The essence of NLP studies lies in capturing the context of the text. For the context to be understood by machines, it is first necessary to understand the internal structure of the text. The relationships between the words in the sentence, the types of words used in the sentence, and the places where they are used are the factors that affect the internal structure of the sentence. Considering all these factors, analysing the internal structure of the text is one of the most difficult stages in NLP tasks. This stage is a time-consuming process that requires powerful hardware, especially in large-scale data sets. However, the structure of the language, emotional expressions created by the emphasis in the text, abbreviations, idioms, and special uses such as proverbs are other elements that affect the context of the text. According to these factors, NLP models developed with traditional machine learning (ML) algorithms may be insufficient in capturing the context of the text. The intended performance cannot be achieved due to the semantic complexity, uncertainty, and difficulty in processing unstructured data that arise especially in the analysis of long texts. Deep neural networks (DNN), on the other hand, enable the automatic learning of complex patterns in large data sets by using word vector representations. This increases accuracy and performance in various tasks. Compared to traditional methods, deep learning (DL) models automatically determine the features and hierarchies in the data and as a result, more robust models are produced [2]. However, they do not have high potential in modelling various dimensions of contextual information. For this, graph-based methods are used. Graph neural networks (GNN) are used as a powerful tool to model complex data in different domains [3].

In this study, the following contributions are presented to shed light on and guide the studies on NLP:

- Compare the work of GNNs with traditional ML and DL models on NLP tasks such as text classification, sentiment analysis, question answering, relationship extraction, text generation, and text summarization.
- Evaluate how GNNs perform in NLP tasks, unlike traditional ML and DL models, in terms of accuracy, efficiency, and contextual information capture capabilities.
- Emphasize the advantages of GNNs over traditional methods in managing complex data structures, modelling relationships, and improving performance in NLP tasks.

The rest of this paper is organized as follows: Section 2 includes the topic of NLP in detail. Section 3 addresses DL models in the field of NLP. Section 4 explains the concept of GNN and its place in the field of NLP. Section 5 includes studies in the literature using graph-based approaches in the field of NLP, and Section 6 provides the conclusion.

2 Natural Language Processing

NLP is a subfield of AI related to the computer's ability to understand, interpret, and process the natural language spoken by humans. NLP allows for the understanding and processing of text and speech in natural language. This capability enables the development of algorithms and models that can generate relevant content [4]. Similar to the development of ML models that mimic human intelligence, NLP enables the generation of text created in human language. In this way, ML models aim to emulate human intelligence. In addition, with NLP, it has become possible to develop applications at the level of human intelligence in various areas such as extracting information from text, text classification, question-answering, and more. The application areas of NLP are illustrated with examples in Figure 1.

2.1 Text Classification

Text classification is one of the most widely used application areas in NLP. Text classification is the labelling and categorization of text content using predefined classes. Text classification models are important in classifying and manipulating complex "structured" data, including social media posts, news, academic articles, and movie reviews [5]. These models are trained on labelled data to learn the relationship between text content and labels, and can then be used to classify new, unseen text data [1]. This makes text interpretation and management easier in various industries and domains. Text classification is divided into various subcategories that cater to different needs. For example; Few Shots Text Classification aims to classify text with very few labelled examples for each class, Long Text Classification is preferred in situations where context

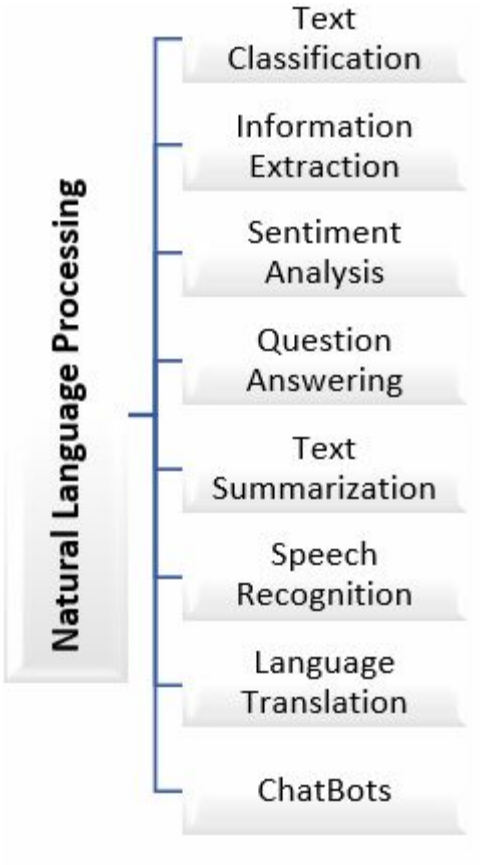

Figure 1. Application Fields of NLP.

needs to be established in long texts (e.g. articles, reports), Short Text Classification is the classification of texts such as tweets and short comments, Hierarchical Text Classification is the classification of texts under a multi-level hierarchical structure, and Multi-Label Classification is used when a text can be assigned to more than one label or category.

2.2 Sentiment Analysis

Sentiment analysis is an NLP task that aims to determine the emotion expressed in a given text. This is usually done by assigning a sentiment label such as positive, negative, or neutral based on the content of the text. Sentiment analysis has important real-life applications such as measuring customer satisfaction, managing political campaigns, and film and media reviews. In addition, there are different sentiment analysis approaches used for different purposes [6]. Aspect-based sentiment analysis (ABSA) focuses on specific aspects or features in the text and performs sentiment assessment for each aspect. Document-level sentiment analysis is often used in long documents, articles, or reviews and determines the overall sentiment for the entire document. Sentence-level sentiment analysis provides information about the emotional tone of each sentence, especially in long texts. Temporal sentiment analysis analyses how sentiment changes over time. Multi-view sentiment analysis analyses a text from multiple perspectives or features (e.g., text content, user profile) and provides a more comprehensive sentiment analysis by combining these perspectives. These methods and types allow sentiment analysis to be performed more effectively and precisely in various contexts and application areas.

2.3 Question Answering

Question Answering (QA) systems are artificial intelligence technologies designed to deliver precise and relevant answers to user queries. QA systems can automatically respond to questions expressed by users in natural language and present these answers by pulling from various data sources (text documents, databases, the internet) [1]. Since QA aims to answer questions by analysing the structure and meaning of language, it can also include other NLP tasks such as information extraction and text classification. QA systems have diversified according to user needs, data sources, and application areas, such as interactive QA systems, rule-based QA systems, time-series QA systems, and multimodal QA systems. Multimodal systems provide more comprehensive answers by extracting information from different types of data. Time-series QA systems answer questions about data or events that change over time. Rule-based systems usually provide answers using fixed rules and patterns. Interactive QA systems are systems that interact with the user and present information in a dialogue-like manner, such as chatbots or virtual assistants.

2.4 Relation Extraction

The swift way to understand the context of a document is to extract relationships within the context. Relation extraction is a subfield of NLP that involves the process of identifying meaningful concepts that reflect the essence of the text. This process allows users to understand the main context of a document without having to read it completely. Relation extraction allows more accurate and relevant key phrases to be extracted by considering the structural information of a document [7]. Relation extraction is used in various subjects such as named entity recognition (NER) [8], which identifies entities in a text, and key phrase extraction [9], which involves determining the most important expressions that cover the main topics in a document. Relation extraction can accelerate research and discovery processes by extracting relationships between biological entities

in the fields of genetics, medicine, and biology; it facilitates legal analysis by extracting relationships between entities in legal documents and legal texts; it is used to enrich knowledge bases with data collected from various sources; it can be preferred in educational materials and research documents to improve information processing and summarization processes by determining relationships between important concepts.

2.5 Text Generation

Text generation involves a model or algorithm generating new text based on a given input. Text generation is typically performed using language models, neural networks, and other ML techniques, which are trained to generate meaningful, context-aware, and usable text. Recently, data augmentation techniques have become popular for text data and NLP applications [10]. It can be used in areas such as emotional conversation generation [11], paraphrase generation [12], and syntax-guided text generation [13]. Text generation speeds up the content production process and saves content creators more time, produces human-like responses using chatbots and virtual assistants for tasks such as customer service, technical support, and general information provision, can help translation systems produce more fluent and meaningful translations, and can be a time saver for analysts and managers by automatically generating data analysis results and business reports.

2.6 Text Summarization

Text summarization is an effective method of making text analysis and similar tasks more accessible by summarizing important information from extensive text data. Text summarization can be studied in subfields such as extractive summarization and creative summarization. While extractive summarization focuses on selecting and compiling important passages directly from the main text, creative summarization produces new texts to convey main ideas in a condensed form [14]. Text summarization helps users deal with large amounts of information by quickly and effectively extracting important information from long texts or documents. It provides readers with quick information by providing a concise version of newsletters, articles, or media reports; summarizing educational materials and research reports allows students and researchers to quickly review important information; summarizing social media content makes it easier to understand trends and popular topics, etc.

2.7 Information Retrieval

Information retrieval (IR), defined as the process of obtaining sources appropriate to an information need from among large data collections, is a core function in many real-world applications [100]. In other words, it is the process of providing a user with access to the information they need in a large dataset. For example, search engines like Google use information retrieval techniques to find the most appropriate web pages for the information the user is looking for. Essentially, IR aims to find the right information, in the right order, and quickly. In Information Retrieval (IR) systems, ranking is important to make it easier for the user to access the information they are looking for. Because a query may return many results, but not all of them may meet the user's needs. Therefore, the system prioritizes the correct and most relevant information. Ranking allows the user to access the information they are looking for more quickly and efficiently.

3 Deep Neural Network

Deep Neural Network (DNN) is a type of artificial neural network and plays an important role in the DL field. DNNs are called "deep" because they contain many layers. DNNs consist of an input layer, multiple hidden layers, and an output layer.

Input Layer: This represents the input data of the model. Each neuron in the network represents a feature or input variable.

Hidden Layers: These layers allow the model to learn the data in a more abstract and meaningful way.

Output Layer: Gives the predictions or results of the model. The structure of the output layer varies depending on the problem to be solved.

DNNs have made a breakthrough in the field of short-text classification. Compared with traditional text processing algorithms, text processing algorithms based on DL can learn deeper and more complex features of text, and automatically extract text features to achieve end-to-end processing, eliminating complex manual operations [15].

3.1 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are artificial neural networks developed to process sequential data such as natural language texts or time series. These networks can effectively capture contextual and temporal relationships between data. Thanks to their ability to analyse connections between information while preserving meaning in long texts, they are effectively used in many areas such as NLP, speech

recognition, or computer vision.

The basic working principle of RNN is to process data sequentially in time steps and produce a new output at each step using the output of the previous step. This process consists of 3 stages: Forward Propagation, Backward Propagation, and Weight Update, as follows [16]:

Forward Propagation: Input data is given to the RNN in order. Each word is processed in a time step. The RNN calculates a hidden state in each step and uses this state for the next step. For example, when analysing the words in a sentence, each word is processed in order and the RNN updates the hidden states to understand the relationship between the words.

Backward Propagation Through Time: After calculating the error between predictions and true values, the error is propagated to past time steps. So, weights are updated by considering the error not only in the current step but also in the past steps. For example, if the prediction of the last word of a sentence is wrong, this error is propagated throughout sentence and it is learned how to correct this error at each step.

Weight Update: Based on the gradients (error measurements) obtained during back-propagation, the weights are updated (to reduce the error). For example, if the RNN misguessed the meaning of a word, the weights are adjusted to minimize this error.

The equations of the RNN are showed in from Eq. (1) to Eq. (5) [16].

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)} \tag{1}$$

$$h^{(t)} = \tanh\left(a^{(t)}\right) \tag{2}$$

$$o^{(t)} = c + Vh^{(t)} \tag{3}$$

$$y^{(t)} = \sigma\left(o^{(t)}\right) \tag{4}$$

$$\sum_{t=1}^{\tau} L^{(t)} = L\left(x^{(1)}, \ldots, x^{(\tau)}, y^{(1)}, \ldots, y^{(\tau)}\right) \tag{5}$$

Eq. (1) explained the hidden state operation. In the equation $a^{(t)}$ is activation at the t-th time step, b is the bias term, W identifies the weight matrix between hidden states, $h^{(t-1)}$is the output of the hidden state from the previous time step, the U is the weight matrix between the input and the hidden state, and $x^{(t)}$ is the input at the t-th time step. Eq. (2) shows the process of hidden state activation. Here, $h^{(t)}$ shows the hidden state at the t-th time step. Eq. (3) shows the process of output activation. In the equation, $o^{(t)}$ identifies output activation at the t-th time step, c is the bias term of the output layer, and V is the weight matrix from the hidden state to the output. Eq. (4) shows the output process. In this step, $y^{(t)}$ is the predicted output at the t-th time step, and σ is the Sigmoid activation function. The total loss function, as shown in Eq. (5), is necessary to evaluate the overall performance of the model and to update the weights during the backpropagation process. In the equation, τ is the total number of time steps, $L^{(t)}$ is the loss function at t-th time step, $x^{(t)}$ is the input vector at t-th time step, and $y^{(t)}$ is the target output vector at t-th time step.

3.2 Convolutional Neural Network

Convolutional Neural Networks (CNNs) are one of the DNNs which are designed to discover local patterns and features in text or visual data. Unlike classical ML algorithms, a CNN can learn complex features from the input, meaning that the feature extraction is automatically performed by the network. This reduces the need for manual feature extraction. CNNs are particularly popular for their ability to adapt to texts of different lengths using kernels of different sizes. This enables the understanding of both concise expressions and extensive contextual sequences [2].

The general CNN architecture for a text processing task includes the following layers, as shown in Figure 2, input layer, embedding layer, convolutional layer, pooling layer, fully connected layer, and output layer. Depending on the task at hand, the number of layers, filter sizes, and other hyperparameters may vary.

Input Layer: The role of the input layer is to transform the data into a proper format and provide the necessary features so that the model can train correctly.

Embedding Layer: The embedding layer helps the model understand the input text by representing words as dense, low-dimensional vectors. These vectors can capture semantic relationships between words and enable the model to learn more effectively.

Convolutional Layer: This layer uses special filters to learn and extract important features from the data. These filters move around different regions of the data to extract specific features. The results obtained after applying the filters are known as feature maps. These maps are representations of specific features in the input data.

Pooling Layer: Usually comes after convolution layers and is used to reduce the size of feature maps, reduce computational load, and preserve the most important

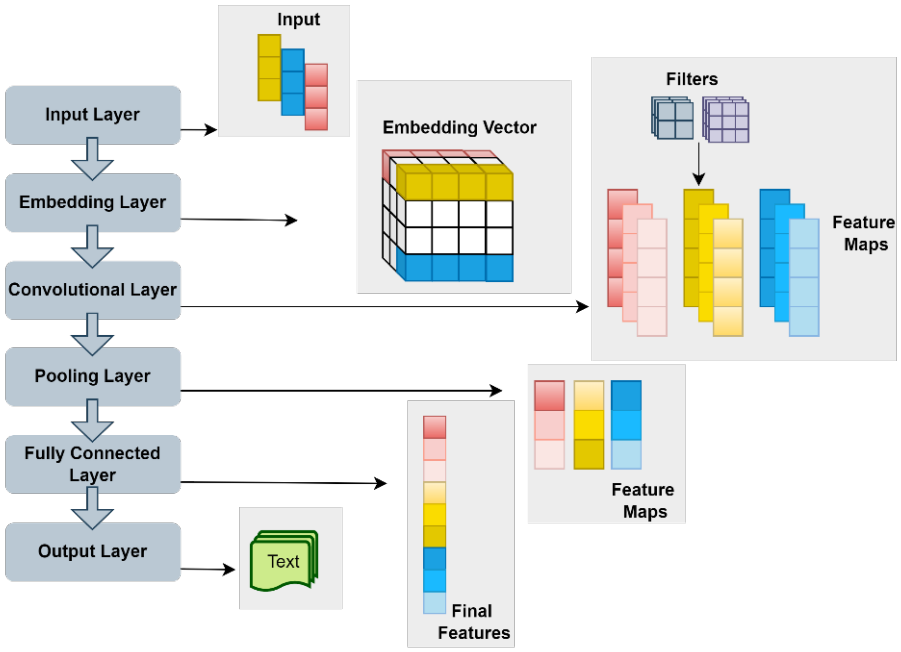

Figure 2. CNN Architecture.

features.

Fully Connected Layer: Also known as a dense layer, is a layer where each neuron is connected to all neurons in the previous layer. This layer is used to process features at a higher level and make final classification or regression predictions.

Output Layer: The layer is the layer where the model provides its final predictions. The design of the output layer varies depending on the problem which is tried to solve.

3.3 Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of RNN developed by Hochreiter and Schmidhuber. LSTMs have special structures called memory cells that allow information to be retained for long periods. While traditional RNNs use a simple activation function to control the flow of information, LSTMs provide more fine-grained control over when data is stored, when it is forgotten, and when it is converted to output. LSTMs contain three main gates: the forget gate, the input gate, and the output gate. These gates determine when information is added to the memory cell, when it is removed, and when it is updated. In a basic LSTM network architecture, as shown in Figure 3, each gate performs certain mathematical operations.

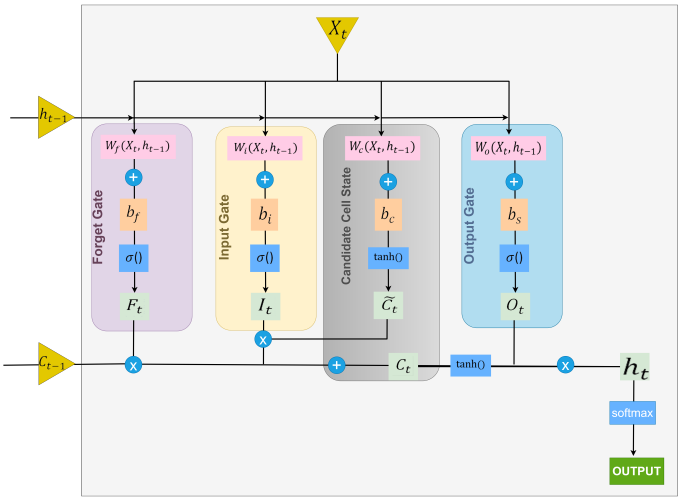

Figure 3. LSTM Network Architecture.

Forget gate: In the forget gate, the current input X_t and the previous hidden state $h_{(t-1)}$ are used to decide which information of the prior state should be stored in long-term memory and which should be forgotten. This decision is based on how important the information in the past cell states is.

Input Gate: The input gate vector I_t is calculated by considering the current input X_t and the previous hidden state $h_{(t-1)}$. This vector determines whether or how much new information should be added to the new LSTM memory cell $\tilde{C}_t$. Using the forget gate and input gate vectors, the final memory cell C_t takes its final form by preserving the previous information and adding new information.

Output Gate: Output gate regulates the transfer of information in the LSTM cell to the next time. The output gate vector O_t is multiplied by the tan hyperbolic function $\tanh$ of the cell state C_t to create

the new hidden state h_t [17].

3.4 Bidirectional Long Short-Term Memory

Bidirectional Long Short-Term Memory (Bi-LSTM) is one of the RNN approaches that is widely used in NLP problems. Bi-LSTM captures information from both directions by running two copies of the LSTM, one forward and one backward. In this way, during each epoch (training cycle), the forward LSTM network and the backward LSTM network are trained, separately. These trained networks are connected in the output layer. Consequently, the full contextual information about the past and future in the input sequence is captured [18]. This allows the model to use more context information, which helps obtain more accurate results. As expressed in Figure 4, Bi-LSTM enables the model to consider both the previous and next context at each time step. For example, in a sentence that includes n words, in the forward direction, the first word is processed first. Then the second word is processed together with the first. Then the $(n-1)^{th}$ word is analysed together with the n^{th} word.

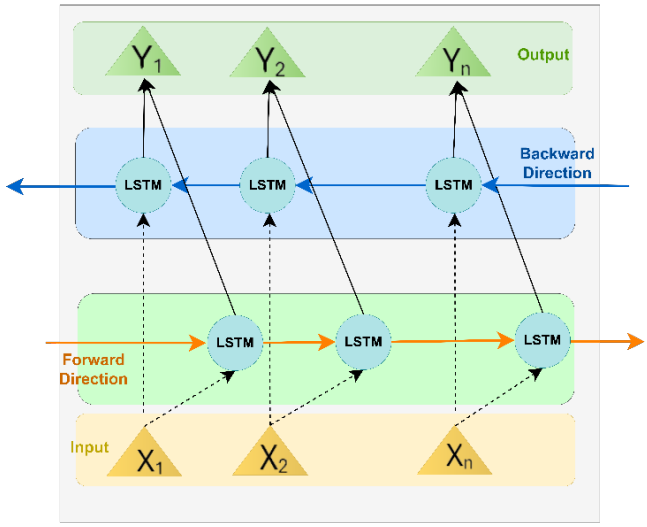

Figure 4. Bi-LSTM Network Architecture.

By combining the information obtained from these two directions, Bi-LSTM creates a richer and context-independent representation of each word. In this way, the model can better understand the context between the words and make more accurate predictions. In the backward direction, the n^{th} word is processed first, then the n^{th} word and the $(n-1)^{th}$ word are processed, and finally the first word is processed.

3.5 Gated Recurrent Unit

A Gated Recurrent Unit (GRU) is a simplified version of the LSTM network, designed to learn long-term dependencies more effectively and to reduce computational cost. GRU has two main gates: an update gate and a reset gate. In the GRU working principle, firstly r_t is calculated in the reset gate using the input vector x_t and the cell state in the previous time step h_{t-1}. This calculation process in the reset gate is done as shown in Eq. (6). In this equation, W_r is the weight matrix for the reset gate, and σ is the sigmoid activation function. Then, the update gate z_t is calculated again using the input vector x_t and the previous state h_{t-1}, as shown in Eq. (7). W_z is the weight matrix for the update gate and σ is the sigmoid activation function. In the next stage, the candidate memory state $\tilde{h}_t$ is calculated, as shown in Eq. (8). Here $W_{\tilde{h}}$ is the weight matrix for the candidate memory state and $\tanh$ is the hyperbolic tangent activation function. Finally, the final memory state, h_t, is calculated, as shown in Eq. (9). The update gate creates a mixture between the previous state and the candidate memory state and h_t is obtained [19]. In this way, GRU effectively learns and maintains long-term dependencies by updating the memory states at each time step.

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{6}$$

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{7}$$

$$\tilde{h}_t = \tanh\left(W_{\tilde{h}} \cdot [r_t * h_{t-1}, x_t]\right) \tag{8}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{9}$$

3.6 Bidirectional Gated Recurrent Unit

A Bidirectional Gated Recurrent Unit (Bi-GRU) is a neural network model consisting of two GRUs. At each time step, inputs are provided to both forward and backward GRUs simultaneously. The outputs of the Bi-GRU model are determined using the combined information from these two directions [19]. That is, the Bi-GRU contains two separate GRU cells, as shown in Eq. (10)) to Eq. (13) [20], one processes data from left to right (forward direction), and the other from right to left (backward direction). The forward GRU processes words from left to right, while the reverse GRU processes words from right to left. As a result, the outputs of both GRUs are combined, and thus the model understands the full context around each word (both before and after). This allows the model to understand both past and future information.

Thus, it provides a richer representation, especially in applications such as NLP.

$$\overrightarrow{h}_t = \text{GRU}(x_t, \overrightarrow{h}_{t-1}) \tag{10}$$

$$\overleftarrow{h}_t = \text{GRU}(x_t, \overleftarrow{h}_{t-1}) \tag{11}$$

$$h_t = \alpha \overrightarrow{h}_t + \beta \overleftarrow{h}_t + b \tag{12}$$

$$y_t = \sigma(W_b h_t) \tag{13}$$

where $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ represent the state of the hidden layer in the positive and negative directions at time t. α and β represent the weights in the positive and negative directions, and b represents the translation value. W_b is the weight matrix.

Traditional text classification methods are particularly difficult to apply to short text data. Compared with traditional ML methods such as logistic regression and support vector machines (SVMs), DL models can prioritize local and sequential features of the text and show good results in terms of classification. However, they ignore the global features of the text when modelling short texts [15]. In addition, traditional ML and DL methods are inadequate in providing solutions to NLP problems with unstructured data. The different types of data from the data source also make this difficult. In recent years, GNNs have attracted widespread attention from researchers because these networks can effectively handle text structures containing complex relationships and different types of data samples and preserve global word features. Consequently, in recent years, GNNs have been applied to NLP tasks.

4 Graph Neural Network

In recent years, models developed with DL approaches have shown high-performance results in every field, from image processing to NLP. Despite their high performance, traditional DL models can only operate on data in Euclidean space. For this reason, traditional methods are inadequate in solving problems with irregular, complex structures and variable-size data. For this purpose, GNNs that can also process non-Euclidean data have been introduced. GNNs are a DL model used to perform operations on data in the graph data structure and analyze these structures using neural networks. Graphs are data structures consisting of nodes and edges. The purpose of GNNs is to develop generalized models by using the dependencies between nodes and edges in the graph structure. In GNNs, node properties, and message passing are updated by collecting information from neighbouring nodes and reflecting this information to node properties through an update function. This process continues with each iteration, which updates the nodes to include a broader range of neighbourhood information. To understand how GNNs work, it is important to understand the message-passing framework and its basic formulation, as shown in equations Eq. (14,15) [21].

Message Aggregation:

- Each node receives messages from its neighbouring nodes.
- Messages are created using the properties of neighbouring nodes.
- Often an aggregation function is used (e.g. sum, average, maximum).

Message Update:

- The collected messages are updated along with the current properties of the node.
- New node features are calculated using an activation function and learnable parameters.

$$h_u^{(k+1)} = \text{UPDATE}^{(k)}\left(h_u^{(k)}, \text{AGGREGATE}^{(k)}\left(\{h_v^{(k)} \mid \forall v \in N(u)\}\right)\right) \tag{14}$$

Basic GNN Formulation:

$$h_u^{(k+1)} = \sigma\left(W_{\text{self}}^{(k)} h_u^{(k)} + W_{\text{neigh}}^{(k)} \sum_{v \in N(u)} h_v^{(k)} + b^{(k)}\right) \tag{15}$$

- $h_u^{(k+1)}$: The state of node u at layer $k+1$.
- σ: Activation function (e.g., tanh, ReLU).
- $W_{\text{self}}^{(k)}$ and $W_{\text{neigh}}^{(k)}$: Learnable weight matrices.

- $b^{(k)}$: Learnable bias term.
- $N(u)$: The neighbourhood set of nodes u.

In recent years, GNNs have been at the forefront of DL research, showing the most advanced performance in various applications. Thanks to GNNs, interest in processing graph-structured data that can solve complex interactions between objects has increased. It has become possible to develop generalizable and reliable models, especially with large and unstructured data such as social media interactions, advice and recommendation systems, and analysis of molecular structures in the field of medicine or chemistry. GNNs, like DL models, create numerical representations of the inputs in their structure, in other words, nodes, and edges, usually in the form of a matrix. They transmit information through the operations they perform on these matrices and detect dependencies between nodes and edges. Essentially, a GNN model is an algorithm that uses graph connectivity to learn and model relationships between nodes. Through an iterative process based on graph structure, GNN takes graph feature vectors and transforms them into output feature vectors (representing target predictions) [11, 22]. For example, if the problem under consideration is a classification problem, the output representation(s) can generally be considered a class. The output representations obtained in a simple GNN model are compared with real values through a loss function. This loss function usually uses a back-propagation algorithm. As a result of the comparison, the model's parameters are updated and the training process continues in this way. This process continues iteratively until the minimum loss function is obtained, and the training process is completed when the optimum result is reached. The performance of the trained model is evaluated with test data or a validation set. In GNNs, output representations can also be used in different subtasks. For example, a GNN model can be developed to analyse the account movements of a bank's customers. Using the model's output data, it can be examined whether a customer can pay a certain amount of loan. As another example, keyword extraction can be performed using the graph structure over text data. Different text generation tasks can be accomplished with the created keywords (output representations). These advantages of GNNs overcome the difficulties experienced in the field of NLP due to complex and unstructured data sets. Together, GNN and NLP are a powerful combination for creating high-performance and generalizable models. Thanks to this combination, the text context and internal structure are analysed effectively. In this way, it has become possible to overcome challenging tasks by integrating GNN approaches into NLP tasks. GNNs have been applied to many NLP tasks, such as sequence labelling, machine translation, and relation extraction. They also have been shown to be successful in question-answering tasks [23].

4.1 Graph Interaction and Convolution Approaches

These techniques refer to the process of information flow and node properties in graph structures. This topic covers methods that examine how interaction and convolution techniques are applied in graph structures.

Graph Convolutional Neural Network

Graph Convolutional Neural Network (GCN) is a type of GNN that uses a convolutional process to create node representations of a graph. The basic idea of GCN is to apply convolution operations to the nodes of a graph. In this process, representations of nodes are learned using the features of a node and its neighbour nodes. In GCN, each node collects the feature information of its adjacent nodes, this process is similar to performing convolution on the graph. Through this convolution, the features of neighbouring nodes are used to update the features of the current node. Through successive layers of message passing, the nodes of the last layer learn not only the direct neighbours but also the characteristics of the neighbours of these neighbours [8].

In a basic GCN model, the representation of a word in a sentence is calculated by the formula in Eq. (16) [8]. First of all, an adjacency matrix $A = (a_{i,j})_{n \times n}$ is created for the dependencies between words. In the equation, in the case of $i = j$ or if there is a dependency between the $i\text{-}th$ word and the $j\text{-}th$ word, $a_{i,j} = 1$, otherwise $a_{i,j} = 0$.

$$h_i^{(l)} = \text{ReLU}\left(\sum_{j=1}^{n} a_{ij} \cdot \left(W^{(l)} \cdot h_j^{(l-1)} + b^{(l)}\right)\right) \quad (16)$$

Then, in each GCN layer, each $I\text{-}th$ node collects the feature information of its neighbour nodes. Also, the output representation $h_i^{(l)}$ of the $i\text{-}th$ word is calculated in each $l\text{-}th$ layer. W^l is the learnable weight matrix used in the $l\text{-}th$ layer and b^l is the learnable bias vector used in the $l\text{-}th$ layer. ReLU is the activation function that enables the model to gain non-linear

properties by resetting negative values.
GCNs can be preferred when data such as text and images do not have a graph-based structure, or when data with a graph-based structure such as social networks have complex and diverse connections between edges [24].

Graph Attention Network
Graph Attention Network (GAT/GAN) enables you to perform operations on this data by determining the importance of nodes and connections in data organized in the form of a graph. GAT often uses multiple attentional mechanisms in parallel. Each attention mechanism calculates its attention scores and aggregation results. These multiple attention mechanisms capture different aspects of the relationships in the graph. The outputs of multiple attention mechanisms are combined to create a final feature vector for each node, usually through merging or averaging. GAT working principle is as follows [25]:

- Each node and edge are expressed by a feature vector.
- Each node accesses information from its neighbours. Taking this neighbourhood information into account, GAT evaluates each node's interaction with its neighbours.
- GAT uses an attention mechanism to model relationships between neighbours. This mechanism gives weight to information from each neighbour. These weights determine the importance of nodes over their neighbours.
- The attention mechanism predominantly collects feature vectors from the neighbours of the node. This aggregation process creates a new representation of the node based on information from neighbouring nodes.
- The weighted information collected is usually processed with an activation function.
- GAT includes a final output layer for specific tasks on the graph (classification, prediction, etc.). This layer performs specific tasks using the general properties of the nodes or graph.

The equations from Eq. (17) to Eq. (19) [26] explain how GAT works and how GNNs gather information by taking into account the relationships between nodes: Initial representations z_i are determined for each node. These representations represent the properties of the nodes. These representations are usually created by an embedding process and a vector is assigned to each node in the graph. Then, using the z_i, the attention coefficient a_{ij} between node i and neighbouring node j is calculated. This coefficient is obtained by combining the node representations and using an activation function such as LeakyReLU.

$$a_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[\mathbf{W}\mathbf{z}_i \parallel \mathbf{W}\mathbf{z}_j\right]\right)\right)}{\sum_{k\in N_i}\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T\left[\mathbf{W}\mathbf{z}_i \parallel \mathbf{W}\mathbf{z}_k\right]\right)\right)} \tag{17}$$

$$h_i = \sigma\left(\sum_{j\in N_i} a_{ij}\mathbf{W}_h\mathbf{z}_i\right) \tag{18}$$

$$\mathbf{W} = \mathbf{W} - \eta\frac{\partial L}{\partial \mathbf{W}} \tag{19}$$

The new representation h_i of node i is updated using the weighted sum of the attention coefficients a_{ij} and the representations of neighbouring nodes.
The W weight matrices in Eq. (17) to Eq. (19) are the parameters optimized in the learning process of neural networks which are randomly assigned and updated based on the data during the training process. This matrix is used to update the node representations after the attention coefficients a_{ij} are calculated. The node representations are multiplied by the weight matrices and propagated forward using activation functions. At this stage, new representations h_i of each node are calculated. A loss function L is used to evaluate the accuracy of the model's predictions. This function measures the difference between the predicted values and the real values. Based on the output of the loss function, gradients ∂ are calculated to update the weight matrices. The weight matrices are updated with a parameter called the learning rate η. In each iteration, the weights are updated to fit the model better with the data.

Message Passing Neural Networks (MPNNs)
Message Passing Neural Networks (MPNNs) are a type of neural network that uses multiple passing steps in addition to the basic GNN operation to process

graph data. MPNN consists of a series of messaging and updating steps to learn the relationships between nodes and edges [27]. Firstly, MPNN collects the features of neighbouring nodes at each step. And it does this with a message function as in Eq. (20) [28].

$$m_v^{t+1} = \text{Aggregate}_{w \in \mathcal{N}(v)} \left[M^t(h_v^t, h_w^t, e_{vw}) \right] \quad (20)$$

Here m_v^{t+1} is the sum of messages received by the node V from neighbours in step $(t+1)$-th. Secondly, in the message collection phase, nodes update their own properties using the messages they receive. In the update process, the node updates its new property using its current property h_t and the messages it collects (m_v^{t+1}). In the last phase, all properties are collected and a readout operation is performed as in Eq. (21) [28] to obtain the general properties of the graph.

$$\hat{y} = \text{Readout}\left(h_1^T, h_2^T, \ldots, h_n^T\right) \quad (21)$$

Here, h represents the feature of node i obtained in the T-th step, and the Readout function computes a single summary value by combining these features. As a result of the readout operation, all node features are brought together and a graph-level feature vector is obtained. MPNNs contribute to effective studies on many types of graph data such as social networks, biological networks, and chemical compounds.

Self-Attention Graph Neural Networks

Self-Attention Graph Neural Networks (SAGNNs) are a model that combines graph-based structures with self-attention mechanisms. Graph-based models focus on learning relationships between nodes on graph data. However, they have difficulty modeling dependencies or relationships in long sequences. Self-attention mechanism overcomes these limitations and can dynamically learn the importance level of each element and determine its weights. This is especially important for complex structures such as social networks [29]. Thanks to self-attention, importance levels are dynamically assigned to different inputs and how important a particular input is compared to other inputs is learned. SAGNNs combine these two powerful tools to better understand long-term dependencies between nodes in graph data. The model can dynamically determine the strength of relationships between nodes and thus make more accurate predictions.

Gated Graph Neural Networks

Gated Graph Neural Network (GGNN) is a type of artificial neural network that works similarly to the RNN structure and processes data in a graph structure. GGNN uses gate mechanisms to learn the relationships between nodes in graph-based data and to model how these relationships change over time. Gate mechanisms decide which information will update the state of the network. In other words, these gates determine how much of the past information will be kept and how the current state will be updated. The general mechanism of GGNNs is explained in the equations below [30].

$$q_{\text{input}} = \sigma(A_{\text{input}} \cdot x + B_{\text{input}} \cdot u + b_{\text{input}}) \quad (22)$$

Eq. (22) determines how much the input gate will be opened based on the input and the current state. In other words, the input gate calculates how much of the new input will be considered when updating the current state. In the equation, x is the current state vector, u is the input vector, A_{input} and B_{input} are graph filters for the input gate, b_{input} is the bias term, and σ is the logistic sigmoid function controlling the gate with output between 0 and 1.

$$\text{state} = \sigma(A_{\text{state}} \cdot x + B_{\text{state}} \cdot u + b_{\text{state}}) \quad (23)$$

In the state gate equation, Eq. (23), calculates how much of the past state will be taken into account in calculating the new state. A_{state} and B_{state} are graph filters for the state gate, weighting the state and input based on the graph structure, while b_{state} is the bias term for the state gate. Other terms work similarly to the previous equation.

$$x_{\text{new}} = \sigma_c \left(q_{\text{state}} \odot (A_{\text{state}} \cdot x) + q_{\text{input}} \odot (B_{\text{state}} \cdot u) + b_{\text{updated}} \right) \quad (24)$$

In the last Eq. (24), the new state is calculated using the information obtained from the input and state gates. Here, q_{state} is the output of the state gate, q_{input} the output of the input gate, A_{state} and B_{state} are graph filters used for the state and input gates, b_{updated} is the bias term for the updated state, σ_c is the hyperbolic tangent function compressing the output to between -1 and 1, and $\circ$ denotes the Hadamard product (element-wise multiplication).

As expressed in these equations, thanks to GGNNs, information flow, and state updates are controlled more flexibly.

4.2 Graph Construction and Generation

Graph construction and generation include the processes of creating graph structures and generating new graphs. Below are some key concepts that explain this category.

Heterogeneous Graph Neural Networks
Heterogeneous Graph Neural Networks (HGNNs) graphs that contain different types of nodes and/or different types of edges.

While HGNN provides the graphical structure of data relationships, it also provides high-level semantic information of the data. An HGNN defined as a heterogeneous graph (or heterogeneous information network) is defined by the set of nodes (V) and the set of links (E). V represents each node and E represents each link. Each node $(v \in V)$ and each link $(e \in E)$ is associated with their mapping functions $(\Phi(v) : V \rightarrow A$ and $\varphi(e) : E \rightarrow R)$. Here, A indicates node types, R indicates connection types, and $A + R > 2$. The network diagram of G can be viewed as a meta-template of a heterogeneous graph $G = V, E$ with node type mapping function $\Phi(v) : V \rightarrow A$ and link type mapping function $\varphi(e) : E \rightarrow R$. It is defined as (A, R). A network diagram is a graph defined over connections defined by node types from A and relationships from R [27].

HGNN is a powerful tool for understanding complex data structures and extracting valuable information from this data. For example, HGNN;

- enables the integration of information from various data sources and facilitates the analysis of these data together,
- analyse relationships between different types of entities such as users, posts, tags in social networks,
- can be used in the field of bioinformatics to discover relationships between genes, diseases and drugs,
- can be used to determine semantic similarities between texts or in knowledge-based question-answer systems.

Dynamic Graph Neural Networks

Dynamic Graphs, unlike static graphs, contain nodes, edges, and properties that change over time. Static graphs have a fixed structure that does not change over time. In other words, nodes and edges remain fixed and do not change from the moment the graph is created. In contrast, in dynamic graphs, nodes, edges, and even properties of nodes and edges (e.g. weights) can change over time, that is, there is a time dimension. Dynamic graph neural networks (DGNNs) are a type of neural network that can model dynamic, time-evolving relationships. In time-varying social networks, it is a preferred neural network framework for modeling user behavior, analysing how financial transactions and market relationships change over time, or modeling time-varying traffic flow [31].

In general, the general model paradigm of a DGNN includes message passing to collect information from neighbouring nodes and edges, and a node update step to update hidden states based on aggregated messages and node properties. The information collection and update formula for the basic DGNN is formulated in Eq. (25) [32].

$$h_{v,t} = \text{Update}_t\left(h_{v,t-1}, \text{AGG}_t\left(h_{u,t-1} \mid u \in N(v)\right), x_{v,t}\right) \tag{25}$$

In the equation, $(\text{AGG}_t\left(h_{u,t-1} \mid u \in N(v)\right))$ is used to collect the previous hidden states from the neighbouring nodes of the node v. With the $Update_t()$ function, the new hidden representation $(h_{v,t})$ of node v is calculated using the previous hidden state $(h_{v,t-1})$, the collected message $(m_{v,t})$, and the current node properties $(x_{v,t})$.

DGNNs can be considered in two ways: as Discrete-Time Dynamic Graphs (DTDGs), as shown in Figure 5, and Continuous-Time Dynamic Graphs (CTDGs), as shown in Figure 6. In DTDG, time is divided into certain steps (e.g. every hour, every day) such as t_1, t_2, etc. At each time step, there are different graphs such as G_{t_1} at time t_1, G_{t_2} at time t_2. Changes in the structure of the graph are captured only at these certain time points. In CTDG, instead of time steps such as t_1, t_2, and t_3, there is a continuous time flow and every moment is taken into account. At any moment in time (uninterruptedly), the structure of the graph is updated, so the graph changes instantly as events occur [31].

Spatio-Temporal Graph Networks
Spatio-temporal graphs are a special type of GNN that can process the connections between nodes in both

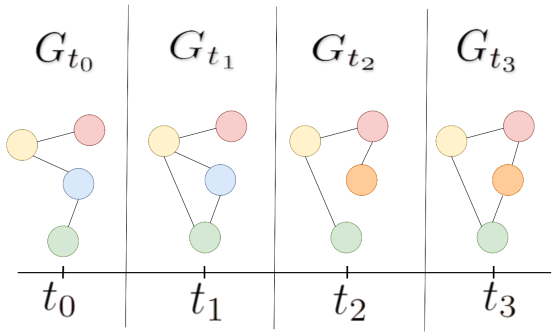

Figure 5. Discrete-Time Dynamic Graphs (DTDG).

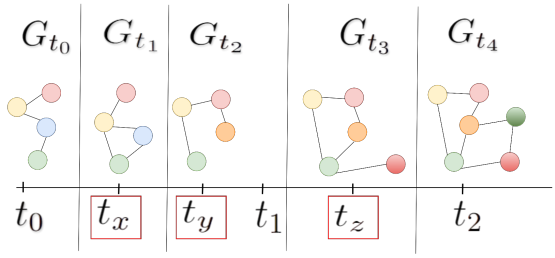

Figure 6. Continuous-Time Dynamic Graphs (CTDG).

spatial and temporal terms. These types of networks are often used to learn and model spatial and temporal dependencies in complex data structures. These graphs contain nodes, edges, and adjacency matrices; nodes represent specific locations, and edges represent connections between these nodes. The number of nodes is usually fixed, while the number of edges can change or remain fixed over time. Graphs can be directed or undirected, and weighted or unweighted. Spatio-temporal graphs can be static (time-invariant) or dynamic (time-varying).

The type of graph to be used depends on the type of data and the problem being analysed. A spatio-temporal graph is expressed as $G_t = (V, E_t, A_t)$ [33]:

- V: The set of nodes.
- E_t: The set of edges.
- A_t: The adjacency matrix at time t.

Graph Augmentation

The field of data augmentation (DA) encompasses techniques for augmenting/generating training data without the need for further data collection or labeling. Augmented data acts as a regularizer when training data-driven models and reduces the overfitting problem. Due to the irregular structure of graph data, it is difficult to apply classical data augmentation techniques directly to the graph data structure. To overcome this challenge, Graph Data Augmentation (GDA) methods are used. GDA methods offer an attractive solution to reduce noise, fill in noise, and generally align the graph structure better with the objectives of the target learning task. Moreover, these methods can be categorized as rule-based approaches and learned approaches according to whether learning occurs in the data augmentation process.

Rule-Based Approaches: In these methods, certain rules are used to modify the graph. For example, it can randomly remove some nodes of the graph or change some connections. In this process, learning is usually not done; i.e., how the graph is modified is based on predetermined rules.

Learning-Based Approaches: Learning-based approaches refer to augmentation methods that include parameters that can be learned in the process of generating augmented examples. Graph structure learning methods assume that the observed graph data is noisy, incomplete, or completely missing, so they first try to learn the "clean" graph structure and then use it during training and inference of GNNs. More effective modifications to such techniques are made using methods based on learning graph properties. [34, 35].

4.3 Graph Embedding

Since graphs are usually high-dimensional and complex structures, it is difficult for ML algorithms to understand them. Graph embedding solves this problem by transforming the nodes in the graph into low-dimensional vectors (vector representations). Various graph embedding methods are explained below to solve this problem.

Graph AutoEncoder

Graph Autoencoder (GAE) is a type of autoencoder model used to process graph data and learn latent representations of this data. Graph data consists of nodes and connections (edges) between these nodes. GAE processes such structured data to create a lower-dimensional representation of the graph that preserves the original information. A basic GAE consists of encoder and decoder components.

Encoder: It usually consists of one or more layers. The encoder takes the node properties and structure of the graph, usually using a Graph Convolutional Network (GCN), and transforms this information into a lower-dimensional latent representation. In each

layer, new representations are generated using the node properties and the adjacency matrix.

Decoder: It tries to re-estimate the structure or connections of the original graph using the latent representations. The solver part can be used to estimate the connections between node pairs or to complete missing edges.

The main goal of GAE is to estimate the connection probabilities between each node pair in the graph structure and to create the reconstructed adjacency matrix. The reconstructed adjacency matrix is calculated by the following formulas. In the first layer, Eq. (26), the node feature matrix X is multiplied by a weight matrix W_1, and then an activation function σ is applied. The adjacency matrix $\tilde{A}$ and the normalized Laplacian matrix $\tilde{L}$ are involved in this process.

$$Z^{(1)} = \sigma(\tilde{A}XW_1) \tag{26}$$

$$Z^{(2)} = \tilde{A}Z^{(1)}W_2 \tag{27}$$

In the second layer, Eq. (27), $Z^{(1)}$ is the hidden representations obtained from the first layer. W_2 is the learnable weight matrix. Consequently, the hidden representations $Z^{(2)}$ of the second layer are computed [36].

Variational Graph Auto Encoder

Variational Autoencoders (VAEs) are an improved version of a regular autoencoder by adding a Bayesian component that learns the parameters representing the probability distribution of the data. Therefore, VAE can be said to be a generative model [37]. Variational Graph Autoencoder (VGAE) is a framework used to learn data in the graph structure based on VAE. As in the traditional GAE architecture, it consists of encoder and decoder components. In the encoding part, VGAE takes an adjacency matrix and a feature matrix and obtains a latent variable in the output. In the decoding part, VGAE obtains a recreated adjacency matrix according to the latent variable.

For this, first, the input data of the model is represented in a lower-dimensional latent space with the mean value (μ) of the representation of each node in the latent space and the standard deviation $\log \sigma$ of the representation of each node in the latent space. For this, an adjacency matrix obtained from the node features and the relationships between the nodes is processed in the GCN network. Then, a latent variable (z) is calculated for each node according to Equation 28. The expression ϵ in Eq. (28) is a random noise vector drawn from a normal distribution with mean 0 and variance 1.

$$z = \mu + \sigma \cdot \epsilon \tag{28}$$

Using the latent representation (z) obtained by the encoder, the adjacency matrix A_m of the graph is estimated in the decoder stage. This estimate is made by the inner product of the latent vectors. The Sigmoid function (σ) is applied to the inner product of the latent vectors as formulated in Eq. (29). The result gives the estimated adjacency matrix, i.e. the probability that each pair of nodes is connected to each other [38].

$$\hat{A}^m = S(zz^T) \tag{29}$$

DeepWalk

DeepWalk [39] is one of the vector point-based graph embedding methods [40]. DeepWalk aims to preserve the structural information in the graph by creating mathematical vectors of nodes in the graph and to represent the nodes in a more meaningful way. Thus, the structural relationships of the nodes in the original graph are preserved and these nodes become more manageable in mathematics and computation. DeepWalk uses random walks to represent the nodes in the graph as low-dimensional vectors. This algorithm performs random walks of a certain length from each node in the graph and aims to preserve the similarities in the graph with the node vectors obtained as a result of these walks.

DeepWalk performs random walks of a certain number (t) from each node and creates vector representations of the nodes with the examples obtained from these walks. The vector representation of each node represents the original structure as well as possible by preserving similar connections in the graph. To preserve the indirect relationships between two nodes, the dot products between the vectors are calculated and it is aimed for this similarity score to be close to 1 [41].

Node2Vec

Node2vec [42] is an algorithm that aims to model the relationships between nodes in a graph in both local and global contexts. Unlike the DeepWalk method, which attempts to model the relationships between nodes using random walks, Node2vec uses a biased random walk approach. This method combines

both BFS (breadth-first search) and DFS (depth-first search) strategies and uses various parameters for these strategies. This allows for a more flexible exploration of node neighbourhoods. One of these parameters, p (return parameter), controls the probability of returning from the node. Higher p values reduce the probability of returning, allowing for broader exploration. The other parameter, q (in-out parameter), controls the direction in which node neighbourhoods are explored. $q > 1$ encourages local exploration, while $q < 1$ encourages broader exploration. In DeepWalk, transition probabilities are determined by considering node neighbourhoods in random walks, while in Node2vec, they are calculated together with a search bias α_{pq} that determines the transition probabilities from node to neighbour. In this way, it is possible to create a more flexible and comprehensive node representation.

In Node2vec, the transition probability from one node to another is calculated with the formula in Eq. (30).

$$\pi_{vx} = \alpha_{pq}(t, x) \cdot k_{vx} \tag{30}$$

According to this equation, π is the transition probability from node v to node x. k_{vx} is defined as the edge weight and α_{pq} is defined as the search bias depending on the parameters p and q [40].

Large-Scale Information Network Embeddings

Large-Scale Information Network Embeddings (LINE) [43] is a method used to model the relationships between nodes in graph-structured data. Unlike random walk-based methods such as DeepWalk and Node2vec, LINE aims to model first-degree and second-degree similarity information between nodes.

First-Order Similarity: It expresses the similarity between two nodes that are directly connected in the graph. This similarity is calculated with a formula similar to the Sigmoid function. The first-order similarity between nodes n_i and n_j can be calculated by replacing x in the Sigmoid function, shown in Eq. (31, 32), with the vector representations of these nodes (s_i and s_j). The Sigmoid function normalizes the values between 0 and 1, which represents the probability similarity between n_i and n_j.

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{31}$$

$$p(n_i, n_j) = \sigma(\mathbf{s}_i^\top \mathbf{s}_j) = \frac{1}{1 + \exp(-\mathbf{s}_i^\top \mathbf{s}_j)} \tag{32}$$

Second-Order Similarity: It is the modeling of relationships based on the common neighbours of nodes. In this way, the similarity between two nodes is measured by the similarity of the local neighbourhood structures of these nodes. In Eq. (33, the second-order similarity between nodes n_i and n_j is formulated. In the equation, the $(\exp(\text{similarity}(n_j, n_i)))$ expresses how well the node n_j is in the neighbourhood of n_i. The $(\sum_{k \in N} \exp(\text{similarity}(n_k, n_i)))$ shows the suitability of all nodes in the graph with node n_i [44].

$$p(n_j \mid n_i) = \frac{\exp(\text{similarity}(n_j, n_i))}{\sum_{k \in N} \exp(\text{similarity}(n_k, n_i))} \tag{33}$$

As a result, graph representations with LINE can be used to better understand the relationships and structures in the graph and are suitable for use in various ML tasks.

4.4 Graph Analysis and Learning

This section covers techniques for understanding the structural and relational properties of graphs and using this knowledge with various learning methods.

Knowledge Graphs

A knowledge graph is a directed graph that shows multiple relationships between entities (E) and relations (R) between these entities. Each knowledge graph consists of nodes and edges. Nodes represent entities and edges represent relationships between entities. Based on this, a knowledge graph is a multi-related, directed graph consisting of triple expressions of the form (entity-1, relation, entity-2) [45], as shown in Figure 7. A knowledge graph is created through the stages of knowledge acquisition, knowledge improvement, and knowledge evolution, respectively. Knowledge retrieval is the process of gathering information about entities and relationships from structured data to create a knowledge graph. Since the extracted triples may be missing, the next information curation step corrects these shortcomings with additional data. Finally, the evolution of real-world knowledge over time may not be reflected in the generated static knowledge graphs, so the graphs are dynamically updated with the knowledge evolution step [46].

Co-occurrence Graphs

The concept of co-occurrence in NLP refers to the

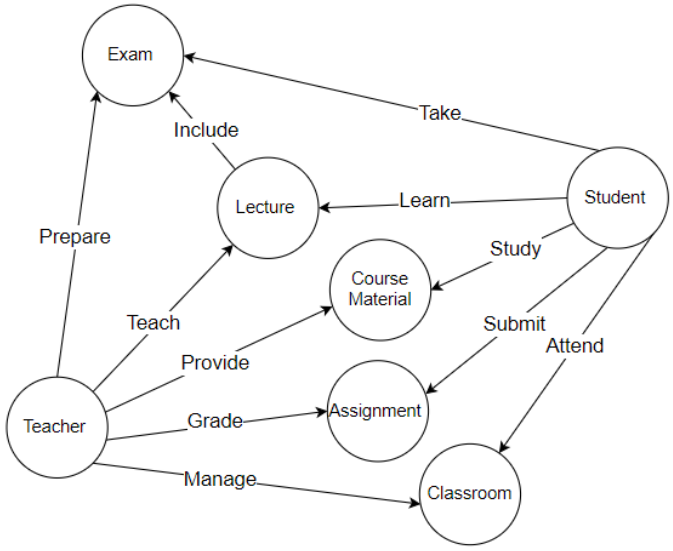

Figure 7. Example of Knowledge Graph.

situation of using at least two words together. In more detail, it refers to the frequency with which two or more words are used together in the same context within a text. For example, if the words 'student' and 'school' frequently occur together in a text, these two words have a high "co-occurrence" value. Such co-occurrences play an important role in understanding the semantic relationships between words and determining the closeness of words to each other. They are frequently used in NLP tasks, such as information extraction, text mining, text summarization, and text generation.

Since graphs have complex data structures, the representations to be created as a result of the network must be obtained with an accurate analysis result. For this reason, the concept of "co-occurrence" in GNNs can be used to analyze the relationships between nodes or edges. The edge weights that define the relationship of nodes to each other can be determined by the "co-occurrence" values. An edge with a high co-occurrence value also has a high weight, which indicates the existence of a strong relationship between the nodes to which this edge is connected. The step of creating a co-occurrence graph is expressed in Algorithm 1 [47].

Semantic Graphs

Semantic graphs are used to model concepts in a language and their relationships in a more systematic way. In this way, complex relationships and meanings in a language can be modeled more effectively. For example, in text classification, a semantic graph can be used to represent the context of words, allowing the model to consider not only words but also their relationships with other words. Semantic graphs can be examined under two headings as Descriptive Semantic Graphs and Executable Semantic Graphs [48].

Descriptive Semantic Graphs are generally used to express the meaning and content of a particular topic or piece of text in detail. They can generally be used in NLP tasks such as analyzing the meaning of texts, extracting information, and generating text.

Executable Semantic Graphs are task-specific, dynamic, and functional graphs. In other words, they can take various data inputs and produce outputs based on this data.

In general, when these two graph types are considered together, Descriptive Semantic Graphs can be preferred to create a semantic representation of data, while Executable Semantic Graphs can be used to process and analyze this semantic representation for a specific task.

Graph Isomorphism Network (GIN)

Graph Isomorphism Network (GIN) was developed using the Weisfeiler-Lehman test algorithm to create a GNN architecture that will satisfy even the strongest (discriminative) GNN conditions.

In addition, GIN can distinguish the similarities and differences between graphs at the maximum level. GIN generalizes this test and can make the maximum distinction between graphs. Also, with GIN, it is possible to learn the relationships between nodes on graphs in the best way and to distinguish graphs using these relationships. The process of a GIN network is shown in Eq. (34, 35) [49].

$$\text{Aggregate}_v^{(k)} = \sum_{u \in N(v)} h_u^{(k-1)} \tag{34}$$

$$h_v^{(k)} = \text{MLP}^{(k)} \left((1 + \epsilon^{(k)}) \cdot h_v^{(k-1)} + \text{Aggregate}_v^{(k)} \right) \tag{35}$$

In the equations, first, information is collected from neighbouring nodes, that is, the representations of the node v's neighbours are collected. In the second step, the update is performed with a multi-layer perceptron (MLP). In other words, the collected neighbouring node representations and the previous representation of the current node are combined and updated. In the update process, $h_v^{(k)}$ is the representation of node v at

Algorithm 1: Co-occurrence Graph Construction

Data: A set of m documents
Result: A co-occurrence graph $G(t)$
Initialization:
$G(t=0) = (W(0), E(0), g(E,t))$ where $W(0) = \emptyset$ and $E(0) = \emptyset$;
Process Documents:
for $j = 1$ **to** m **do**
  For each document D_j;
  for $l = 1$ **to** k **do**
    For each sentence $S_{j,l}$;
    for $i = 1$ **to** h **do**
      For each word $w_{j,l,i}$;
      Add Words to Graph:
      if $w_{j,l,i} \notin W(t)$ **then**
        $W(t+1) = W(t) \cup \{w_{j,l,i}\}$;
      end
      Connect Words:
      if $i > 1$ ***and*** $(w_{j,l,i}, w_{j,l,i-1})$ *are in the same sentence* **then**
        if $(w_{j,l,i}, w_{j,l,i-1}) \notin E(t)$ **then**
          $E(t+1) = E(t) \cup \{(w_{j,l,i}, w_{j,l,i-1})\}$;
          $g((w_{j,l,i}, w_{j,l,i-1}), t+1) = f(w_{j,l,i}, w_{j,l,i-1})$;
        end
      end
      Add Document Information:
      Calculate a document centroid Centroid$_{D_j}$;
      Link document ID (e.g., URL) to words:;
      Link$(w_{j,l,i})$ = URL$_{D_j}$;
      Increment Indices:
      $i \to i+1$;
    end
    $l \to l+1$;
  end
  $j \to j+1$;
end
Process All Text:
if $i \leq h$ **then**
  Process the next word
end
if $l \leq k$ **then**
  Process the next sentence
end
if $j \leq m$ **then**
  Process the next document
end
Stop

the end of the k-th iteration, $MLP^{(k)}$ is the MLP for the k-th iteration, $\epsilon^{(k)}$ is a fixed and learnable parameter for the k-th iteration, and $h_v^{(k-1)}$ is the representation of node v in previous iterations.

Graph Kernels
Graph kernels are used to calculate and compare similarities in graph-structured data. In other words, graph kernels try to capture the semantic structures that graphs have. The field of NLP has a wide application area in solving real-world problems in graph-structured data [50]. There are various approaches used for graph kernel architectures, such as Neighborhood Aggregation, Kernels for graphs with continuous labels, Random walk kernels, etc. [51]. Each approach analyzes the graph differently.

Assignment and Matching-Based Approaches:
The aim is to find the most appropriate component match in two graph structures. For this, the Optimal Assignment (OA) Kernel is used. In OA kernel:

- A label or representation is determined for each node.
- The obtained representations are compared using a basic kernel.
- The obtained similarity expresses the overall similarity between the two graphs.

However, this match usually does not form a positive semi-definite kernel, which makes it difficult to use some methods.

Subgraph Patterns:
Subgraph patterns include various techniques to make the comparison and analysis of graphs more efficient.

Bag-of-Words: With this method, graphs can be treated as a collection of nodes and edges, similar to traditional NLP tasks.

Graphlet Kernel: Counts small subgraph patterns within the graph. These patterns express a certain structure and compare the graph over these patterns.

Neighbourhood subgraph pairwise distance kernel: Compares the subgraph around each node with a string representing it at a certain depth.

Walks and Paths:
A more general analysis can be done by comparing the order of node or edge properties in graphs with Walks and paths.

Shortest-path kernels: Compares the lengths and labels of shortcut paths between nodes in two graphs. However, this method can have high computational complexity.

Random Walk Kernel: Compares the label orders of random walks between two graphs. This method can be complex because it requires direct product graphs and matrix operations.

Kernels for Graphs with Continuous Labels: Some kernel methods used in comparing graphs with continuous labels are as follows:

KGraphHopper and GraphInvariant Kernels: Weisfeiler-Lehman label renewal is used to measure the structural similarity of node labels with the GraphInvariant kernel and a matrix is created with the GraphHopper kernel that measures the positions of nodes in the shortest paths. In this way, the labels of nodes and the relationships of these nodes in the graph structure are taken into account.

KSubgraph Matching Kernel: Evaluates matches between all subgraphs in the graph and uses specialized kernels to compare node labels.

KDescriptor Matching Kernel: Provides information propagation among node neighbors and uses pyramid matching kernels to compare node labels.

5 Recent Advances in NLP with GNNs

Challenges in NLP The general problems encountered in the field of NLP are quite complex due to the diversity and complexity of texts. Traditional text processing methods face challenges such as word independence and lack of context, while DL models have to deal with issues such as data volume and model complexity. Short texts may pose challenges for supervised learning models due to the lack of labelled data [52].

Since it is difficult to capture the meaning in short texts, sufficiently effective analysis may not be possible on topics such as information extraction, sentiment analysis, or topic modelling. However, processing a large amount of unlabelled data with limited labelled data may not yield accurate results [53]. This makes it difficult to capture long-distance dependencies in modelling inter-sentence relationships, which hinders the full understanding of the context. On the other hand, the rapid evolution of the Internet can make it difficult to continuously update sentiment dictionaries, which may limit the effectiveness of sentiment analysis. DL methods can be effective in solving these problems, but training complex network structures can pose challenges [55]. In addition, establishing the connections between users and questions in QA platforms can be complex, because questions can belong to multiple topics and users can follow many topics. Existing methods may struggle to establish these relationships effectively [56]. Lack of user inference can degrade the performance of question suggestion algorithms. Furthermore, current emotional chat systems are often limited to users' emotional inputs and may ignore other important emotional information.

Proposed Solutions to Problems: HGNNs are a proposed approach to capture long-distance dependencies and inter-sentence relationships more effectively. This method can improve document-level relationship extraction [54]. However, extracting relationships only at the sentence level may not capture relationships across multiple sentences effectively enough. To overcome these limitations, dynamic graphs are structured with multi-step connections for each document, allowing for better modelling of implicit relationships. This approach provides more comprehensive relationship extraction by managing document-level complexities [26]. HGNNs can also be effectively used in sentiment analysis and chat systems. Existing sentiment chat systems are often limited to users' sentiment input only and may ignore other important information. HGNNs can be used to more comprehensively address all emotional components of conversation—such as facial expressions, voice, and personality. This approach can improve the quality of emotional responses [57]. These emotional analysis and interaction advances may set new standards, especially in multimodality processing.

Limitations of the Proposed Solutions: The basic structure of GNN architectures is quite complex. Despite the effectiveness of these structures in capturing long-distance dependencies and modelling inter-sentence relationships, this approach has certain limitations. First of all, the complex structure of HGNN-based methods and the need to work with large data sets increase computational costs. This situation can be problematic, especially in small-scale projects or applications with limited resources. In addition, although the process of establishing multi-step connections in HGNN-based methods provides significant advantages, these structures can sometimes include unnecessary or irrelevant relationships in the model. Because each connection may have different weights in the learning process of

the model and incorrect associations may take place in the network. On the other hand, since graph-based methods require large data sets, the labelling process in this data becomes costly, or working with unlabelled data may limit the efficiency of these models. The limited number of labelled data may negatively affect the learning process of the model, because in this case, the model may not be able to fully learn the diversity in the real world. As a solution to this problem, the use of data augmentation techniques may be preferred. However, incorrect or low-quality augmented data may reduce the performance of the model. Additionally, data augmentation techniques need to be carefully selected and implemented to be effective, as incorrect data augmentation can further narrow the generalization ability of the model.

Future Research Graph-based techniques, especially GNNs, provide a powerful framework for handling complex relationships in NLP tasks. However, many important areas remain unexplored to fully realize their potential. Future research could focus on the following directions: Scalability and Efficiency Improvements: While GNNs are effective at capturing relationships between entities in graphs, computational complexity remains a challenge when working with large-scale data. Future work could focus on optimizing the scalability of GNNs, developing more efficient architectures, or using approximate methods to maintain performance while reducing computational costs. Handling Dynamic Graphs: Many real-world applications, especially in NLP, deal with dynamic or evolving graphs (e.g., social media or web data). Research on dynamic GNN (DGNNs) could enable the development of models that can adapt to changing structures and relationships in data over time. This could lead to major improvements in time-varying tasks such as trend analysis, sentiment evolution, and real-time information retrieval. Graph-Based Models for Cross-Language NLP Tasks: Another promising direction for graph-based techniques is to model the relationships between words or concepts in different languages through graphs. This can improve machine translation, multilingual knowledge retrieval, and cross-lingual sentiment analysis, better capturing semantic similarities and differences across languages. Integration of Knowledge Graphs: Interest in knowledge graphs, which represent entities and their relationships in a structured manner, has increased in recent years. Future work can focus on integrating knowledge graphs with GNNs, which can improve the understanding of domain-specific knowledge, leading to better results in tasks such as question answering, knowledge retrieval, and recommendation systems.

Studies on Explainability: As GNNs become increasingly prevalent in critical NLP applications such as healthcare and finance, the explainability of these models is of utmost importance. Future research can focus on increasing the transparency of graph-based models and making model decisions more understandable and trustworthy to users. This is a critical factor, especially in areas where model decisions must be justified.

Application to Multimodal NLP Tasks: Graph-based techniques can also be applied to multimodal NLP tasks where different types of data, such as text, images, and audio, are processed together. Future work can explore how GNNs can integrate information from multiple modalities to provide more comprehensive understanding and analysis in tasks such as multimodal sentiment analysis, visual annotation, and audiovisual question answering.

Personalized Graph-Based Models for User Interactions: Personalized models that consider user preferences and interactions are important in areas such as social media and recommender systems. Future research can focus on developing personalized GNNs that learn from user-specific data, enabling more accurate recommendations, content filtering, and emotion detection in user interactions.

5.1 Text Classification

Hua et al. [15] proposed a heterogeneous GCN-based (SHGCN) method for text classification. The model first constructs the graph network of texts and extracts entity and word nodes in this network. Then, the relationship of graph nodes is determined through the mutual information between words, the relationship between documents and words, and the trust between words and entities. Then, the word features are represented by BERT embedding and augmented by BiLSTM. Finally, the augmented word features are combined with the document graph representation features to predict document categories. Experiments are conducted on public datasets such as AGNews, R52, and MR to verify the performance of the model. The classification accuracy of SHGCN is 88.38%, 93.87%, and 82.87%, respectively. A summary of key studies in text classification using graph-based NLP is provided in Table 1.

Wang et al. [52] worked on the topic of text

Table 1. Summary of Text Classification Studies on Graph-Based NLP.

Author	Methods	Dataset	Proposed Method	Evaluation Metrics
Hua et al. [15]	BERT, Bi-LSTM, HGNN	AGNews, R52, and MR	SHGCN	Accuracy
Wang et al.[52]	GNN, GRU	MR, Ohsumed, R8, R52, 20NG	TextFCG	Accuracy
Cui et al. [53]	GCN	Ohsumed, AGNews, StackOverflow, THUCNew, Snippets, MR	ST-Text-GCN	Accuracy, Macro-F1
Bhadra et al. [58]	GNN, GCNN	—	—	Accuracy
Yang et al. [59]	GCN, HGNN	Biomedical, Dblp, MR, SST1, SST2, TREC and Twitter	WC-HGCN	Accuracy
Xu et al. [60]	Multi-perspective aggregation based GNN	Amazon, HuffPost, FewRel	Frog-GNN	Accuracy
Hua et al. [61]	GAT, GNN	MR, Ohsumed, R8, R52, 20NG	HieGNN	Accuracy
Yang et al. [62]	BERT, GGNN, Co-Attention and Aggregation	MR, SST-2, R8, Ohsumed	BEGNN	Accuracy, Macro-F1
Zeng et al. [63]	GCN	AAPD, RMSC-V2, Reuters-21578	S-GCN	Micro-F1, Precision, Recall
Onan [114]	Hierarchical Graph, Knowledge Graph, Attention-based Graph Learning	20NG, Airline Twitter dataset, App dataset, MR, Ohsumed, R52, R8, Sarcasm dataset	—	Accuracy, Precision, Recall, F1-score, ANOVA test
Ai et al. [64]	GAN, HIG	TagMyNews, Snippets, Ohsumed, MR, and Twitter	EMGAN	Accuracy, F1

classification by combining DL methods and graph-based methods. They developed the GNN and GRU-based learning model TextFCG. The model aimed to combine the contextual relationships in the text on a graph. In addition, the transductive learning methods used in existing text classification tasks showed a decrease in their performance when they encountered texts containing new words, and the TextFCG was proposed as a solution to this problem. In this approach, the tasks of emotion classification, classification of medical summaries into specific diseases, and classification of news content are addressed. In the model, a separate graph was created for each text in the dataset. The graphs represented the words in the texts and the contextual relationships between them. As the second stage, it was aimed to learn the contextual relationships between the texts in more detail by using GNN on the created graphs. Using the relationships enriched by GNN, the text was divided into classes with the TextFCG. In summary, the strategy of analysing the relationships of the words in each text within the text and then the contextual relationships with other texts were applied in the model. Consequently, it is emphasized that the experiments TextFCG achieved successful performance in the text classification task.

Cui et al. [53] worked on a study to overcome the problems of insufficient labelled data, the manifold structure, which is a mathematical concept used to represent the internal properties of the data, preferred instead of the sample text creation technique applied in traditional methods. In the study, Ohsumed, AGNews, Stack Overflow, THUCNews, Snippets, and MR datasets were used. Within the scope of the study, the GCN-based ST-Text-GCN (Self-Training Method Based on GCN) method, which is a word-based recursive method, was introduced. This method can train itself when labelled data is insufficient to improve classification performance. Instead of adding additional information, labelled data is passed on to other samples along the manifold structure. In this way, the model identifies words with high word reliability as pseudo-labels and increases the classification performance by using these words. It is claimed that the ST-Text-GCN method has high

robustness compared to state-of-the-art methods in the literature, without using pre-trained word embedding methods or additional information.

Another study on short text classification was conducted by Bhadra et al. [58]. They carried out a study which was a GNN-based text classification model. The study was developed to identify potential influencers and micro-influencers, which is an important challenge in the social media environment. Within the scope of the study, the data set was manually labelled, and simulated using bootstrapping, and a connection matrix was created showing the relationship of the nodes to each other. First, GNN was used to consider node and edge features. In this model, first of all, the interaction numbers of people are created, and if the number of interactions of people exceeds a certain threshold, the person is labelled as an 'influencer', otherwise the person is labelled as 'not an influencer'. After the application of the GNN, GCN, which additionally contains a convolution layer at the node level, was included in the model. As a result of the study, an accuracy rate of 91.63% was obtained with the GNN model and 49.4% with the GCN model. Since simulated the data was used on the developed models, the performance of the study on real-world data is unknown.

The other study based on GCN, which was conducted by Yang et al. [59] called WC-HGCN for short text classification. To enhance learning performance and understand the context, the words in the text and the interactive concepts of these words were taken into consideration. For this aim, it was propounded that the proposed model can retrieval the interaction between words and their associated concepts. In the study, firstly, words and concepts were obtained for each text, and using these a heterogeneous graph was created. The graph contains word and concept nodes and word-word edge, word-concept and concept-word directional edges that are updated with the GCN. The experiments were conducted with six data sets frequently used in short text classification (Biomedical, Dblp, MR, SST1, SST2, TREC, and Twitter) and both WC-HGCN model and models described as successful in the field of text classification in the literature. Experimental results show that WC-HGCN achieves better performance compared to other baseline methods. However, the proposed WC-HGCN model has the problem of deteriorating performance as the layer increases in GCNs.

The study that approaches the topic of text classification in a different way was conducted by Xu et al. [60]. They developed the Frog-GNN model using a few-shot approach. It is emphasized that the Frog-GNN model achieved better results in classification compared to the Prototypical networks approach, which is inadequate in capturing class differences in complex data. Within the scope of the study, two methods are used; it is a multi-perspective aggregation based GNN that creates pre-trained pair representations. In this way, Frog-GNN gains the ability to learn new classes with a small number of samples. Three different datasets including the categorization of Amazon customer reviews, the classification of HuffPost headlines into news categories, and relationship classification tasks with FewRel were used. The results showed that the Frog-GNN model performed better than existing BERT-based models.

In addition to other studies in the literature, a text classification model that also evaluates hierarchical relationships has been addressed by Hua et al. [61]. They developed a model called Hierarchical GNN (HieGNN). HieGNN has been claimed to produce more effective results by incorporating the hierarchical structure of text. With HieGNN, it aimed to better understand the context by extracting the hierarchical relationship between words, sentences, and documents and, accordingly, to perform the text classification task more efficiently. In the study, graph structures were created using graph attention networks (GATs) at three different levels: word, sentence, and document level, using R8, R4.2252, 20NG, Ohsumed, and MR datasets. Also, GNN was used on the created graphs. The results indicate that the proposed model was successful in extracting hierarchical information in text classification tasks and achieved high-performance results in the text classification task.

Yang et al. [62] integrated the Bert into GNN and used it in the text classification task. In the developed BEGNN (Bert-Enhanced text GNN) method, four different text classification data sets (MR, SST-2, R8, and Ohsumed) created in the fields of sentiment analysis, news classification, and medical text classification were used. In the study, the graph structure of each text in the data set was created by taking into account the relationships between words. In this way, the structural features of the entire text are determined. Additionally, the texts were converted into a specific format with Bert Tokenizer. Texts obtained with Bert Tokenizer were used as input data

in the Bert language model. At this stage, it is aimed to extract the semantic structure of the texts in the data set with the Bert language model. As the next step, a GNN model was built to define the context of the text and the semantic relationship between words. Finally, with the co-attention module, which was created to ensure the integrity between the representation data at hand, the features extracted from the GNN, semantic features were combined with word, sentence, and document-level operations, and their weights were determined. Ultimately, it is stated that the results obtained with the BEGNN model provide an increase in the overall classification performance.

Compared to other methods in the literature, Zeng et al. [63] developed a GCN-based text classification model Semantic-sensitive GCN (S-GCN) by considering both semantic and word-general relationships simultaneously. A global graph consisting of texts, words, and tags is constructed and helps to extract the relationships between similar documents. Then, an encoder is designed to initialize the text nodes in the graph and pre-trained, thus obtaining the semantic features of the documents. Then, a GCN is used to classify the text nodes and can successfully combine the node information. Finally, the adjacency matrix is normalized and the hidden layer representations of the word nodes are stored, which can address the problem of predicting unseen texts during training. It shows that the multi-label text classification model implemented with AAPD, RMSC-V2, and Reuters-21578 datasets shows superior performance compared to other models.

The study by Onan [114] proposes a hierarchical graph-based text classification framework that leverages contextual node embeddings combined with a BERT-based dynamic fusion mechanism. This approach captures both semantic relationships and contextual information within the text, resulting in enhanced classification accuracy. The framework effectively integrates graph structures and transformer-based language models to improve performance on various text classification tasks.

Ai et al. [64] addressed the issue of short text classification with the Edge-Enhanced Minimum-Margin Graph Attention Network (EMGAN) model they proposed. Previously, a Heterogeneous Information Graph (HIG) was created, which represents the relationships between three features such as entity, topic, and keywords. Then, the connections of HIG were improved with the X-Shaped structure edge enhancement method. Finally, feature information was collected with the Minimum Margin Graph Attention Network (MMGAN) and the classification performance was increased. According to the experiments conducted with TagMyNews, Snippets, Ohsumed, MR, and Twitter datasets, it was stated that the EMGAN method performed better than other methods.

5.2 Sentiment Analysis

Sentiment analysis using graph-based techniques has seen significant advancements in recent years. Researchers have developed various approaches that leverage graph neural networks to better capture contextual relationships and dependencies in textual data. A comprehensive summary of key studies in this area is provided in Table 2.

For sentiment analysis tasks, Wang et al. [6] approached the subject from a different angle by also using images. He proposed a model called cross-instance GNN (CIGNN) that detects sentiment in text-image pairs. In the next step, a co-occurrence matrix is created using the co-occurrence relationships between the attributes and generates an Attribute_GCN. Then, point-wise mutual information and message-passing mechanisms (Text_GNN) are utilized to update the representations of the edges and nodes generated from the textual dataset. To ensure sufficient interaction between image and text, the original text and image representations interact with Attribute_GCN and Text_GNN through the use of multihead attention, respectively. Finally, connect the resulting fusion features and input them into the softmax layer to predict the final sentiment.

Li et al. [55] combined a GNN and BERT to analyse the emotional content of texts and classify them with specific sentiment labels. In the study, a model based on cross-document learning called Multistream BERT GCN (MS-BertGCN) was developed. It has been suggested that the developed model will be a solution to the problem that is frequently encountered in graph-based models and that the use of original texts as a dataset causes the loss of some effective information. In cross-document learning, there is a general framework learning based on the similarities and interactions between text groups. In the MS-BertGCN model, first, the texts in the training set are grouped according to class similarities. Then, heterogeneous graphs are created for each class with the BertGCN model. The MS-BertGCN model is developed with the obtained heterogeneous graphs.

Table 2. Summary of sentiment analysis studies on Graph-based NLP.

Author	Methods	Dataset	Proposed Method	Evaluation Metrics
Wang et al. [6]	Point-wise Mutual Information, Message-passing Mechanisms, Multi-Head Attention, GCN, GNN	ResNet101, COCO, ImageNet, MVSA-Single, MVSA-Multiple, TumEmo	Attribute-GCN, Text-GNN, CIGNN	Accuracy, F1
Li et al. [55]	GCN, BERT, HGNN	MR, SST-2	MSBertGCN	Accuracy
Wang et al. [65]	BERT, GGNN, Attention-based Readout Function	Provo, GECO, Mishra, ZuCo, MR, SST-2, STS-Gold	SEMGraph	Accuracy
Gu et al. [66]	GCN	Restaurant14, Laptop14, Tweets, Restaurants16	EK-GCN, WSIN	Accuracy, F1
Lu et al. [67]	GAT, GRU	Twitter, Rest14, Rest15, Rest16, Lap14	Hete_GNNs	Accuracy, F1
Wu et al. [68]	BERT, Attention network, GCN	Twitter, Rest14, Rest15, Rest16, Lap14	LDEGCN	Accuracy, Macro-Avg. F1
Sunny et al. [69]	Stanford CoreNLP, Stanza, Berkeley neural parser, context trees, RGAT	SemEval14 and Twitter14	SynthFusion	Accuracy, Macro-F1
Jin et al. [70]	Attention mechanisms, BERT, HGNN	MR, IMDB, SST-5	BUGE	Accuracy, Precision, Recall, F1
An et al. [71]	Attention Mechanism, HGNN, Information Propagation	Rest2014, RestLarge, Rest2014-hard, RestLarge-hard, MAMS-Small	HAGNN	Accuracy, F1
Zeng et al. [72]	BERT, HGNN	MOSI and MOSEI	HIS-MSA	MAE, Correlation Coefficient, Accuracy, F1

Wang et al. [65] addressed the task of sentiment analysis using GNN from a different perspective than traditional methods. It created a graph architecture by combining emotion information and eye movement data. In the study, SEMGraph, Provo, GECO, Mishra, and Zuco datasets containing eye movement information, and MR, SST-2, and STS-Gold datasets containing emotion concepts were used. First, node representations were created from the texts in the dataset for the graph architecture with a BERT-based model. As a second step, the Linguistic Probing Eye Movement Paradigm is proposed to understand how grammatical features are related to eye movements and to extract eye movement features using this relationship. After understanding the relationship between grammatical features and human reading processes with eye movements, it was focused on the emotion analysis task using the information obtained. For this purpose, with a weighting strategy developed, emotion concepts and eye movement features in the data set are combined and emotion-eye movement weights are determined. A graph representing the relationship between emotion and eye movements was created using the determined weights. GGNN (Gated GNN) models the relationships and information interaction between nodes in the graph. Afterward, to better understand the interaction between word nodes on the graph, FFD (First Fixation Duration) weight, which expresses the duration of the first point the eye focuses on in the text, and Total Reading Time

(TRT) attention mechanisms were integrated into the model. Experiments were first performed without eye movement data, and then eye movement data were also included. With the addition of this data, it was observed that the SEMGraph model showed higher performance in the sentiment analysis task.

The topic of aspect-based sentiment analysis (ABSA) using GNN was discussed by Gu et al. [66]. A GCN-based EK-GCN model was developed using sentiment-lexicon and part-of-speech-information components. A sentiment score was assigned to each word with the sentiment-lexicon component. In addition, the grammatical type of the word was determined using part-of-speech and its effect on the sentiment analysis task was taken into consideration. For the EK-GCN model, open-source Restaurant14, Laptop14, Twitter, and Restaurant16 datasets containing Positive, Neutral, and Negative tags were used. Considering the results obtained, it has been shown that the proposed model performs better in sentiment analysis tasks compared to models using attention mechanisms, semantic-based models, other GCN-based models, and traditional embedding vectors. In addition, in the proposed model, due to the complexity of text structures and the presence of more than one subject-based word in a sentence, the Word-Sentence Interaction Network (WSIN) network was developed to perform sentence analysis more effectively. It has been suggested that it is possible to better learn and interpret the meaning of the sentence in topic-based sentiment analysis tasks by using the WSIN network.

A model called Hete-GNNs has been proposed using heterogeneous graphs on ABSA. In this model proposed by Lu et al. [67], the grammatical structures of words, the connections between words, and the sentence structure were analysed comprehensively by combining sentiment words. Experiments were conducted on Twitter, Lap14, Rest14, Rest15, and Rest16 datasets with the HGNN model created with these different data types. Additionally, GRU was used for pre-trained embedding representations. According to the results obtained, it was concluded that the Hete_GNNs model, developed for sentiment analysis by combining the syntax trees of sentences, word relations, and sentiment dictionary information, can capture context information better than existing methods.

A GCN-based study was also carried out by Wu et al. [68], considering the effects of emotional words on ABSA. A model consisting of two stages called LDEGCN (Local Dependency-Enhanced GCN) was proposed, which aims to increase local dependency. First, the Semantic Feature Extraction Module, which includes an attention network, was built. This aimed to focus word representations (obtained using BERT) on aspects within the context and to extract semantic features effectively. In the next stage, a dependency graph called the Dependency-Enhanced Module was created and emotional relationships were determined by weighting it with SenticNet. To focus on local dependencies, the local dependency weighting (LCW) method was applied to the graph. More comprehensive hierarchical features were obtained by combining these two modules. Studies on Twitter, Rest14, Rest15, Rest16, and Lap14 datasets have been reported to show significant improvements compared to state-of-the-art models.

A different approach was developed by Sunny et al. [69] to solve the problems encountered while performing grammatical analysis in sentiment analysis tasks. This method, called SynthFusion, uses different grammatical analysis tools such as Stanford CoreNLP, Stanza, and Berkeley neural parser to analyse natural language texts grammatically and create syntax trees containing the relationships between words. The created context trees were combined with SynthFusion and an ensemble graph was created. In the next stage, the RGAT (Relational Graph Attention Network) was added to the model, and the connections between the trees in the ensemble graph were strengthened, that is, their weights were updated. Finally, experiments were conducted on the model with SemEval14 and Twitter14 datasets. The results obtained with the SynthFusion model can produce reliable results and a better performance is achieved with the strategy of combining context trees obtained using different grammatical analysis tools.

Jin et al. [70] proposes a novel Bert-based unlinked graph embedding (BUGE) model for sentiment analysis using BERT and attention mechanism. For the BUGE model, firstly, text representations are obtained using BERT-based embedding techniques, and a comprehensive text sentiment heterogeneous graph is created to capture global co-occurrence information between words more effectively. As a second step, Linkless Subgraph Decomposition is created that preserves both global and local information. The BUGE model performs the representation learning process to update node representations and ensure information transfer and attention mechanisms are

Table 3. Summary of question answering studies on Graph-based NLP.

Author	Methods	Dataset	Proposed Method	Evaluation Metrics
Li et al. [23]	BERT, BM25, GAT	National Unified Legal Professional Qualification Examination Counselling Book, JEC-QA	GESAN	Accuracy
Thambi et al. [45]	GNN, BERT-based GNN	MetaQA, WebQSP	—	Hits@k
Wu et al. [56]	GCN, K-mean	Stack, Zhihu, DBLP, Yelp	TCGNN	Entropy, Purity, Precision, Recall
Liu et al. [73]	GCN	ConvQuestions, ConvCSQA	—	Accuracy, F1
Gao et al. [74]	GNN, GAT, MRC	QANGAROO MEDHOP	MedKGQAs	Accuracy
Cai et al. [75]	BERT, Bi-LSTM, GCN, Graph Embedding Generator	CommonsenseQA, OpenbookQA	DSSAGN	Accuracy, Interpretability
Li et al. [76]	GCN, Adaptive Message Passing	TimeQA, TempReason, SituatedQA	CoAG	Exact Matching (EM), F1
Gao et al. [77]	T-GNN, LLM	CronQuestions	GenTKGQA	Hits@1, Hits@10
Su et al. [78]	GAT, GNN, KG	CommonsenseQA, OpenBookQA	PipeNet	Accuracy
De Cao et al. [79]	R-GCN, ELMo	WIKIHOP	Entity-GCN	Accuracy

used at this stage. The extensive experimental results conducted on MR, IMDB, and SST-5 datasets with the BUGE model created in this way confirm the model's effectiveness.

Different from traditional methods, An et al. [71] developed a sentiment analysis model called HAGNN. HAGNN integrates three types of nodes (word nodes, topic nodes, and sentence nodes) into a heterogeneous graph framework. This architecture enables the exchange of structural and semantic information between sentences and topics, enhancing the model's ability to learn from diverse expressions and shared structural patterns across sentences. Experimental validations on five public datasets show that HAGNN achieves 1.25% improvement in average accuracy and 1.16% improvement in average macro F1 score compared to existing models.

Focusing on multimodal sentiment analysis, Zeng et al. [72] developed a model called HIS-MSA. First, it used BERT and in-domain corpus strategies to make BERT more suitable for tasks such as multimodal sentiment analysis. Then, it combined multimodal data into heterogeneous graphs. Finally, HIS-MSA uses a single-modal label generation module to balance multimodal tasks and single-modal tasks. Experiments have shown that the HIS-MSA method achieves significant performance improvement beyond the current state.

5.3 Question Answering

Graph-based techniques have been increasingly applied to question answering tasks, as summarized in Table 3. These approaches leverage graph structures to better model relationships and dependencies in QA problems.

Li et al. [23] developed a model that answers questions using graphs in the legal field. They addressed the model they proposed, called GESAN, in stages. First, they classified the questions according to their legal subjects with the BERT model. Then, they found the most relevant evidence from the legal knowledge base with the BM25 method and converted the question-option pairs and evidence texts into representation vectors with the BERT model. They created a graph using these vectors. Then, they tried to determine which evidence was more important using GAT. Finally, they combined important evidence into a single representation and tried to predict the answers. According to their results, they said that the model's

performance was increased with relevant evidence and that the model was effective.

Thambi et al. [45] proposed a different approach to improve the performance of knowledge-based Q&A systems. The method aims to enable KGQA systems to provide more accurate answers to both simple and complex questions by combining GNN and BERT-based approaches. In the first step, a subgraph is created that includes all nodes and edges related to the subject entity of the question. After the subgraph is created, the relevance score is calculated, which determines how relevant the nodes and edges are to the question. This process is called pruning and only the most meaningful subgraph part is studied. The pruned subgraph is an embedding node using BERT-based GNN. The embedding representations for the nodes are compared with the embedding representation of the question and the nodes with the highest similarity are considered as possible answers. Extensive tests on MetaQA and WebQSP datasets have shown that the proposed model is more effective compared to existing KGQA systems.

Wu et al. [56] in their work, the community presented a method for answering questions. Addressed the problems of question diversity and the lack of a topic tag for the question. For this purpose, a new topic clustering algorithm called TCGNN is proposed in the study. First, a topic association matrix was created to represent the relationships of manually tagged topics. For unlabelled topics, key phrases were extracted from the questions, and topic relationships were represented. The created topic association matrix was converted into a graph representation using GNN. GNN represents the relationships between topics by creating embedding vectors of topics on this graph. The resulting embedding vectors were divided into clusters using the K-means clustering method to identify groups of clusters containing similar represented topics. In experiments conducted on different data sets, the topic clustering performance of TCGNN was evaluated and it was stated that CGNN demonstrated superior performance compared to other methods.

Another study focusing on conversational knowledge-based question-answer systems is discussed by Liu et al. [73]. A study was conducted to improve topic tracking in Conversational KBQA (Knowledge-Based Question Answering) systems and to increase the accuracy of the answers. The study consists of 3 modules; a Textual Reader that predicts topic entities, a Transition Graph that creates the relationships between these entities and topic changes, and a question-answer module that evaluates candidate answers to find answers to questions. These tools calculate the scores of topic entities using graph-based neural networks and thus improve the performance of question-answer systems. Experiments with ConvQuestions and ConvCSQA datasets confirmed that the proposed methods perform well in topic tracking and outperform the baseline methods.

For graph-based Q&A, Gao et al. [74] conducted a study in the field of drug-drug interactions. This model, called Medical Knowledge Graph Question Answering (MedKGQA), uses machine reading comprehension from closed-domain literature and creates a knowledge graph containing "drug-protein" triplets from open-domain documents. The model makes DDI predictions using the resulting knowledge graph. As a result, the model answers the question of how the interaction between two drugs will be.

Graph-based studies have also been carried out to improve the operating performance of multi-hop question-answering systems. One of these works belongs to Cai et al. [75]. They proposed a model for multi-hop question-answering called DSSAGN (Dependent Syntactic Semantic Augmented Graph Network), which can process complex queries over knowledge graphs using dependent syntactic analysis and semantic relations. In the model, GCN processes the information between nodes and edges in knowledge graphs. It also used an embedding generator to represent entities and relationships within knowledge graphs more meaningfully and accurately. Additionally, the Answer scoring module has been integrated into the model to identify and rank correct answers. Experiments with CommonsenseQA and OpenbookQA datasets reveal that they can carefully evaluate the syntactic and semantic contexts of questions and model multi-step relationships effectively.

Li et al. [76] have taken the issue of question-answering from a different perspective. The model developed in their study, called CoAG (Context-Enhanced Adaptive Graph Network), can capture and use complex time dependencies in long texts to provide accurate and effective answers to time-sensitive questions. First, they developed the Hybrid Text Encoder module. With this module, a text encoder supported with global information has been

Table 4. Summary of relation extraction studies on Graph-based NLP.

Author	Methods	Dataset	Proposed Method	Evaluation Metrics
Liu et al. [8]	GCN, Corenlp tool, UMLS, Attention Mechanism	BioRelEx, ADE and DDI	KESAAGCN	Micro-F1
Martínez-Cruz et al. [9]	GCN, Co-occurrence Graph	SemEval-2010, LDKP3K, NUS, DUC-2001, Inspec	—	F1@K, AUC-ROC
Tian et al. [19]	RoBERTa, Bi-GRU, RGCN, Axial Attention, Biaffine Layer	Geological reports from the Geological National Archives in China	—	Precision, Recall, F1
Sun et al. [26]	BERT, RoBERTa, GAT	DocRED	TDGAT	Precision, Recall, F1, IgnF1
Shi et al. [54]	HGNN, TCN, GTN	Based on biometric dataset	DocRE-HGNN	Intra/Inter (Precision, Recall, F1)
Zou et al. [80]	BERT, HGNN, GAT	Chinese medicine books	—	Precision, Recall, F1
Khanfir et al. [81]	GCN, MHA	IAM, Esposalles	SGTE, Cross-GCN	CER, F1
Liu et al. [82]	EA-GCN, GNN, Attention mechanism	Twitter15, Twitter16, Pheme, Weibo	DCEP	Accuracy, F1
Hu et al. [83]	GNN, HGNN, LADIES algorithm	Open Academic Graph (OAG) and Amazon Recommendation	GPT-GNN	MRR, Micro F1
Pi et al. [84]	BERT, GCN	DocRED, RE-DocRED	ECRG	F1, AUC

developed. Secondly, time stamps in the documents were extracted with the Timeaware Episode Extraction module, and sections with timestamps close to the question were identified. Finally, the Adaptive Message Passing module is designed to capture differences between sentences with timestamps and transfer dependencies between remote sentences. The comprehensive and comparative experimental results conducted within the scope of the study emphasize that the proposed model shows a significant increase in performance, especially in solving long-distance time-sensitive questions.

Gao et al. [77], developed a framework called GenTKGQA (Generative Temporal Knowledge Graph Question Answering) using the temporal knowledge graph for question answering. This framework allows large language models (LLMs) to answer time-related questions in two stages. 1) ICL-based subgraph retrieval phase: LLMs are used to determine the important information and time intervals in the question. 2) IT-based answer generation phase: The system generates the correct answer using the subgraph. Large language models (LLMs) and graph signals are used together to improve the answering of complex questions. It has been declared that the GenTKGQA model outperforms the current best methods.

Su et al. [78] developed the grounding-pruning-reasoning framework, PipeNet, to provide a more efficient and effective reasoning process by using knowledge graphs in question-answering tasks. In the grounding phase, they match the question and candidate answers with the knowledge graph. Then, they determine and remove the less important nodes among the matched nodes with pruning. Finally, they try to find the correct answer by reasoning on the remaining nodes with reasoning. Experimental results with CommonsenseQA and OpenBookQA datasets indicate that PipeNet both optimizes memory usage and increases the performance of the reasoning module by effectively pruning the noisy nodes of the knowledge graphs.

De Cao et al. [79] developed a model that can answer questions by reading multiple documents and enabling information integration between documents. In the

study, first of all, context-based word representations were obtained with ELMo. These representations were used to better analyze the meanings and relationships of entities. Afterwards, an entity graph was created. In this graph, entities in documents are represented as nodes, and edges are created between entities in the same document. Relationships between nodes are modelled with Relational-GCN and at each step, the representations of nodes are updated. More in-depth information was extracted with Multi-step Reasoning. Finally, the updated node representations are used to calculate the probability of each candidate's answer. The model selects the answer with the highest probability. The results show that the model is successful in the tasks of reading multiple documents and answering questions and performs better than existing methods.

5.4 Relation Extraction

Relation extraction has benefited significantly from graph-based techniques, as evidenced by the studies summarized in Table 4. These approaches leverage graph structures to capture complex relationships between entities in text.

A study was undertaken to deal with domain-specific challenges such as special terms, abbreviations, and long sentence structures in biomedical documents. In this study by Liu et al. [8], the Knowledge-Enhanced Model with Syntactic-Aware Attentive GCN (KESAAGCN) model, consisting of 3 modules, was proposed. Firstly, a graph representing the initial understanding was created by taking raw text input with a syntactic-aware initial span graph construction module developed. In this process, a syntactically aware attentional GCN based on dependency trees of sentences was used. In the second stage, useful information about the domain was included in the model using another GCN, based on an external knowledge base with a domain knowledge graph construction module. Finally, with the final span graph construction module, the initial spacing graph and the domain information graph are combined with the attention mechanism to obtain a more robust graph for the final prediction. It is underlined that the model, which aims to increase performance by combining both external knowledge base and syntactic information, is successful in detecting relationships between distant entities in long and complex sentences.

Martínez-Cruz et al. [9] conducted a study to eliminate the limiting factors in existing methods and to perform keyword extraction from long texts more effectively and accurately. In the study, firstly, a co-occurrence graph was created with long text documents. In this graph, nodes represent words and edges represent relationships between words used together in the text. Graph representations were obtained by applying a self-supervised convolutional graph network (GCN) model trained with the connection prediction task on the co-occurrence graph. It has been suggested that the approach used, unlike supervised methods, does not require labelling, which is a time-consuming and challenging task to create for training, and instead uses natural co-occurrence patterns in the text to learn meaningful representations of words. Experiments were conducted with the developed model for two different fields consisting of scientific and news documents. Experiments tested whether graph representations could improve the quality of pre-trained language models by providing additional contextual information. Various pre-trained language models have been used, including BERT7, SciBERT 8, DistilBERT9, KBIR10, and Longformer11. Top of Form Graph embedding integration has been proven to improve the performance of Keyword Extraction models for long documents by providing a more comprehensive understanding of the context of a word within the entire text. However, an increase in processing time ranging from 20% to 50% was observed, depending on the length of the document.

Tian et al. [19] addressed the task of information extraction using geological reports. For this purpose, he developed the RoBERTa-RGCN-BiGRU-Axial Attention-Biaffine model structure. In the first step, the texts in the geological reports are given to the RoBERTa model and hidden layer representations are obtained for the tokens. In the second step, the token representations from RoBERTa are transferred to the BiGRU layer and richer representations are obtained by using the Axial Attention mechanism. In the third step, the RGCN (Relation GCN) layer is used to learn the relationships between entities and represent them in the graph structure. In the last step, entity classification and relationship detection are performed in the Biaffine layer. The information from the RGCN layer is used to label object entities under certain relationships. The performance of the model is evaluated by comparing it with various models. The results emphasize that the proposed model provides a significant performance increase compared to other models.

Sun et al. [26] focused on the task of extracting

information from text data in their work. He proposed a new approach based on a dynamic GAT to extract relations at the document level. In the studies presented in the literature on the subject, it is generally claimed that solutions are offered by creating static graphs with heuristic rules and that the models developed in this way ignore the originality of the documents. In the proposed TDGAT approach, document-level relationships are handled using dynamic graph-based methods. The study took place in two phases. In the first phase, GAT was used and entity relationship pairs were determined. Then, the existence of entity pair relationships was questioned with the binary classification model. In the second phase, document-level relationships are extracted in more detail using the entity pair connection information obtained in the first phase. In other words, a dynamic graph is created using the information from the previous stage and finer relationship types are determined through this graph. The performance of the model obtained using the GNN was evaluated on the dataset called DocRED, which includes documents obtained from Wikipedia and Wikidata sources. Experimental results have shown that the presented TDGAT method is more effective than other models in extracting document-level relationships.

Shi et al. [54] proposed the model called HGNN for Document Level Relationship Extraction (DocRE-HGNN) using HGNN for the task of information extraction from text. Stating that the methods presented in the literature on information extraction generally analyse at the sentence level, the study aims to handle the task of extracting information from the text by considering all the sentences in the text, instead of doing it on a sentence basis. First of all, context representations (embedding vectors) were obtained by determining the relationships between the words in the data set using the TCN (Temporal Convolutional Networks) architecture. Creating embedding vectors with bi-directional TCN is aimed at storing historical information and preserving long dependencies. A heterogeneous graph was created containing the relationships between the resulting embedding vectors (words, sentences, and other entities). Semantic relationships on the graph are created with the GTN (Graph Transformer Networks) model. In the last stage, numerical representations of the relationships on the graph that can be processed by AI models were obtained with the Representation Acquisition approach. It is argued that with the developed model, intra-sentence and inter-sentence relationships are extracted effectively. Experiments were conducted with a text-based biometric dataset containing chemistry-disease relationships (CDR), and genetic-disease relationships (GDA). It has been observed that the experimental results successfully handle intra-sentence and inter-sentence relationships.

Zou et al. [80] developed a model using GNN to extract information from traditional Chinese medicine texts. The dataset was created with texts obtained from traditional Chinese medicine books. Representations of the texts in the dataset were obtained by using a fine-tuned version of the Bert language model. Afterward, a HGNN was constructed with different types of structures in the text. Graph attention mechanisms have also been added to heterogeneous neural network layers because traditional models cannot show sufficient performance in processing the semantic relationships of data in heterogeneous structures. In this way, it is aimed to prevent poor performance that may arise from the complex relationships of the texts in the data set and to contribute to making the model more sensitive and effective. After the heterogeneous graph was created, the entities in Chinese medicine texts were identified with the "Two-Stage Entity Identification" method in the final analysis stage. In the first stage, the starting and ending positions of Chinese medicine texts are determined by using a two-class classifier. In the second stage, entity-relationship pairs were determined by determining Chinese medicine entities and the relationships between them using the starting and ending positions. Experimental results using HGNN and the BERT indicate that the proposed model achieves better performance compared to other models.

GCN was used to perform handwritten text recognition and named entity recognition (NER) operations together in a different way. In this study by Khanfir et al. [81], an end-to-end encoder-decoder model combining transformers and GCN was proposed. The proposed model uses the advantages of self-attention mechanism and GNNs for representation learning and relationship extraction at the same time. The model consists of two stages. First, the Sparse Graph Transformer Encoder (SGTE) model they developed was used to represent text and visual features and control information propagation by using the Multi-Head Attention (MHA) mechanism. Second, the Cross-GCN model was created by combining the outputs of SGTE with the outputs

of the MHA block in the decoder. Cross-GCN was developed to provide better alignment of visual features to characters and NE labels and to help make representations more accurate. In the experimental results conducted with Esposalles and IAM datasets, it was emphasized that state-of-the-art models were achieved.

Liu et al. [82] used feature extraction to detect fake news on social media. In his work, he presents a new model called DCEP (detection method based on content, emotion, and propagation) to better understand the emotional aspects of news content. Emotional features are extracted from four main components: emotion lexicons, emotion intensity, emotion score, and symbolic features. In the next stage, emotional interactions during the spreading process of news are simulated using the attention mechanism with emotional interaction modelling. These interactions are used as edge features in the graph structure. Then, a news graph enriched with emotional information is created with edge-aware GCN (EA-GCN). In the last stage, an advanced graph representation is modelled by combining the EA-GCN module output with global emotional information. According to the experimental results conducted with Twitter15, Twitter16, Pheme, Weibo datasets, it is stated that the proposed model is superior to many state-of-the-art approaches. real-world datasets have shown that the proposed DCEP model achieves successful results.

Hu et al. [83] developed a pre-trained GNN model that can capture data distribution. By creating a pre-trained GNN, it is aimed to extract rich semantic information from unlabelled data. The proposed GPT-GNN model is designed to train GNNs on large-scale and heterogeneous graph data. The working process of GPT-GNN includes the steps of creating the initial graph, determining the node attributes, building the GPT-GNN model, estimating the subgraphs created by permutations, and training the model. In addition, sampling is performed using the LADIES (Layer-Wise Distributed Subgraph Sampling) algorithm, especially for heterogeneous graphs. It has been stated that the GPT-GNN model shows high performance in various tasks compared to pre-trained GNN models.

Another study in the RE field proposes to extract evidence sentences in advance and to construct graphs using these evidence sentences to estimate the relationships between entities. Pi et al. [84] proposed the model called Enhancing Cross-evidence Reasoning Graph (ECRG) to design a center-sentence-based evidence extraction rule and obtain higher quality evidence according to this rule. First, the important structures in the document were analysed with the evidence extraction module. Then, the vector representations of the words in the document were extracted with BERT and turned into a graph structure with the GCN network. In the last stage, the information from the previous stages is run by a ML model to understand the entity relationships in the document. It was emphasized that the ECRG model can extract deeper semantic information by preventing unnecessary and repetitive sentences from negatively affecting the model performance in the experiments on the DocRED and RE-DocRED datasets.

5.5 Text Generation

Graph-based approaches have shown promising results in text generation tasks, as summarized in Table 5. These methods leverage graph structures to model complex relationships and generate coherent text.

Ahmed et al. [10], aimed to produce different texts by using the Clonal Selection Algorithm (CLONALG) and Abstract Meaning Representation (AMR) graphics, preserving the original context, meaning, and grammatical structure of the text. The methodology of the study consists of three main stages: 1) Abstract Meaning Representation (AMR) graphics were created to graphically represent the meaning of the sentence. 2) By making controlled mutations on AMR graphs with the Clonal Selection Algorithm (CLONALG), domain-specific keywords were preserved and the contexts of the texts were preserved. 3) New texts were produced with the pre-trained T5 algorithm using modified AMR graphics. Classification models were trained using the augmented data set. It is emphasized that the proposed method manages to overcome the limitations of existing methods for text augmentation.

Guo et al. [13], on the other hand, presented a GNN-based approach to creating more consistent, grammatical, and meaningful sentences in text production. First, syntactic dependency trees of the texts were created with Stanford Parser. In the second step, a GNN model that will operate on the graphs obtained from the dependency trees is created. In addition, message-passing and node updates were made between nodes in the graphs using Message Passing Neural Network (MPNN). The model created in this way was trained with Maximum

Table 5. Summary of text generation studies on Graph-based NLP.

Author	Methods	Dataset	Proposed Method	Evaluation Metrics
Ahmed et al. [10]	AMR, CLONALG, T5	COVID Twitter, PolitiFact, Gossip Cop	—	Accuracy Precision Recall F1, BLEU
Guo et al. [13]	Stanford Parser, GNN, MPNN, MLE, LeakGAN, SeqGAN	IMDB	—	BLEU-n, Edit Distance, NLL (Negative Log-Likelihood)
Liang et al. [57]	HGNN-Based Encoder, Emotion-Personality-Aware Decoder, Attention Mechanism	MELD, DailyDialog	—	PPL, BLEU, Dist-n, W-avg
Du et al. [85]	GCN, AST, GRU	Java codes collected on GitHub	G-DCS	BLEU-n
Lai et al. [86]	Dual-Process Theory, KG, GCN, GPT-2	NT-DESC, Person and animal from KG Wikidata	CogNLG	BLEU, METEOR, TER, ROUGE, PARENT
Wang et al. [87]	DRT, GNN, Local Graph Encoders, Deep Traversal Encoders, Levi Graph Method	Parallel Meaning Bank	—	BLEU, METEOR, NLG, ROSE
Chi et al. [12]	seq2seq, GCN, Attention Mechanism	Quora, ParaNMT	—	BLEU, METEOR, ROUGE-n
Lou et al. [88]	HGNN, GAT	DBLP, Aminer, IMDB	GAAD	Macro/Micro-F1
Chen et al. [89]	GCN, Graph Encoder, Sequence Encoder	KGTEXT	KGPT	BLEU, METEOR, ROUGE-L
Onan [113]	GNN, Genetic Algorithm, GAT	Electronics Dataset from Amazon, Emotions Dataset, Semeval2018, SST1, SST2, The Kindle Dataset, TREC Dataset	GTR-GA	BLEU, Cosine Similarity, Accuracy
Tang et al. [90]	GNN, GPT	OGB-arxiv, PubMed, Cora	GraphGPT	Accuracy, Macro-F1, AUC

Likelihood Estimation (MLE). Meanwhile, the text generation process has been improved with LeakGAN and SeqGAN feedback mechanisms. The performance and results of the developed model were evaluated on both synthetic and real text data. As a result, it has been stated that the model can produce well-formed and consistent structural texts by exhibiting successful performance in both synthetic and real texts.

Liang et al. [57] conducted a study on the inability of emotional speech production systems to adequately perceive emotions and express them appropriately. In the study, a heterogeneous graph-based model is proposed for emotional speech generation. First, a Heterogeneous Graph-Based Encoder was designed and the speech content (dialog history, emotional flow, facial expressions, voice, and speakers' personalities) was represented by a HGNN. Then, using an Emotion-Personality-Aware Analyzer, it produced a response with emotions appropriate to the context of the conversation, taking as input the coded graph representations, the emotions predicted by the encoder, and the current speaker's personality. By combining information from different sources with a heterogeneous graph-based model, emotions are perceived more effectively. The results of the study revealed that the developed heterogeneous

graph-based model performed better than other models by using various information sources. While automatic evaluations show that the use of additional information improves the model's answer quality, human evaluations confirm the model's ability to produce consistent and meaningful answers.

Du et al. [85], developed a model called G-DCS (GCN-Based Deep Code Summary Generation) using graphs to automatically generate code descriptions/summaries from Java codes. GCN was used in the study to model the structural relationships of the code. GCN extracts the structural information of the code by processing the data taken from the abstract syntax tree (AST) of the code and includes this information in the code summarization process. In addition, GRU neural network was used to process the time series information of the code in the study. The experiments were carried out with Java codes collected from GitHub. The obtained results indicated that the G-DCS model provides an effective method to automatically generate the summary of the Java code and that this method creates more accurate summaries by using the structural information of the code.

Another study on text generation was addressed by combining dual processing theory in cognitive science and graph architecture. Lai et al. [86] proposed a text generation framework called CogNLG using the perceptual and analytical systems in Dual Process Theory. In the CogNLG framework, S1 functions as a GPT -2-based text generation model. S2, on the other hand, functions as a GCN-based information extraction system. Experimental results performed on large-scale KG-to-text datasets indicate that the dual system structure makes the text generation process explainable and scalable. Additionally, CogNLG has been highlighted as outperforming the best existing studies on a variety of metrics.

A graph-based method has been developed in order to create more accurate and desired structured sentences in natural language by controlling the subject and focus point during the transition from meaning to text. In this study handled by Wang et al. [87], it was aimed to provide Topic-Focus Articulation (TFA) control in the transition process from meaning to text. This includes the ability to transform between active and passive voice sentences, especially in sentences with transitive verbs. Discourse Representation Theory (DRT) was used in the study to represent meaning logically and graphically. These representations were then transformed into a graph with the Levi Graph Method. In the next step, the adding TFA process was performed. The created GNN architecture was designed with Local Graph Encoders and Deep Traversal Encoders. The results show that TFA types have significantly different effects on the performance of models on graph-structured data.

NLP models can effectively use syntactic information when generating synonyms of sentences. In this direction, a study was conducted to obtain meaningful and diverse sentences by coding syntactic dependencies in paraphrase production. In the study discussed by Chi etal. [12], the Seq2seq model was trained by combining it with GCNs. Dependency trees were encoded via GCNs and this information was integrated into the seq2seq model. Additionally, by using the attention mechanism, the model was enabled to focus on certain parts of the input sentence and make more meaningful translations. The results of the study showed that syntactic information plays an important role in paraphrase production and that this information can be effectively encoded using GCNs.

Lou et al. [88] had discussed a study called GAAD (Graph-data Augmentation and Adaptive Denoising) to avoid being affected by graph structure differences and noisy nodes in the graph structure for data augmentation. First of all, by using the edge relations and node properties learned from the original graph structure, the graph structure was created with edges related to high probability. Afterward, embedded representations of the nodes were obtained with GAT. Finally, the effect of noise nodes was redacted due to the increased edge relationships in the data augmentation step. Detailed performance evaluation of the model shows its superiority compared to other current methods.

Chen et al. [89], presented a solution to the data-to-text generation problem by using graph-based and sequence-based methods together. For this purpose, he developed the KGPT (Knowledge-Grounded Pre-trained Transformer) model. Two types of encoders, sequence-based and graph-based, are used in the KGPT model. Graph encoder was used to represent the knowledge graph and process structured information. Sequence encoder processes text data sequentially. KGPT also used a copying mechanism that allows the model to accurately copy certain elements from the input as output. It has been said that the pre-trained KGPT model gives very good results in tests with the full dataset. Additionally, it was

Table 6. Summary of text summarization studies on Graph-based NLP.

Author	Methods	Dataset	Proposed Method	Evaluation Metrics
Zaffar et al. [14]	NER, GNN	CNN Daily Mail	—	Rouge-n, F1, Recall
Huang et al. [91]	Bi-LSTM, CNN, GRU, GAU, GNN	CNN/Daily Mail, MR	GA-GNN	Rouge-n
Onan et al. [112]	Hypergraph, Hypergraph Contextual Attention, Extractive Read-out Strategy, Attention Mechanism	XSum, CNN/DailyMail, PubMed	MCHES	ROUGE, BERTScore, MoverScore
Nourbakhsh et al. [92]	HGNN, GATe, Bi-LSTM	CNN/DailyMail	ConHGNN-SUM	Rouge-n
Yu et al. [93]	NER, PGN, KG, GCN	MEC devices, Jiangsu Provincial Meteorological Bureau, Wikipedia, Baidu Baike, major news websites	KGCPN	Accuracy, Precision, Recall, F1, Rouge-n
Etaiwi et al. [94]	DL (LSTM), BahdanauAttention	AlJazeera.net	SemG-TS	Rouge-n
Jiang et al. [95]	BERT, Neural Topic Model, GAT, HGNN	CNN/DailyMail, XSum	GATSum	Rouge-n
Chen et al. [96]	R-HGNN, KG	CNN/DailyMail, NYT50	RHGNNSumExt, RHGNNSumAbs	Rouge-n
Ghadimi et al. [97]	GCN, DSN	DUC 2004, CNN/DailyMail	GCSumm	Rouge-n
Zhao et al. [98]	HGNN, GAT, NTM	CNN/DM, NYT	—	Rouge-n
Onan et al. [111]	TG-GNN, GNN, GCN, Maximal Marginal Relevance	XSum, CNN/DailyMail, PubMed	KETGS	ROUGE, BERTScore, MoverScore, METEOR, BLEU
Frisoni et al. [99]	Event Graphs, AMR Graphs, Attention Mechanism	CDSR	Cogito Ergo Summ	Rouge-n

emphasized that it can achieve reasonable results for few-shot learning or zero-shot learning.

In this study, Onan [113] propose GTR-GA, a novel approach that combines graph-based neural networks (GNNs) with genetic algorithms (GA) for text augmentation. The method aims to enrich textual data to improve the generalization ability of models. Experimental results demonstrate that GTR-GA provides more effective and diverse text augmentation compared to traditional techniques. This work offers an innovative solution for enhancing model performance, especially in scenarios with limited data.

Tang et al. [90], developed a model called GraphGPT to achieve stronger generalization ability in graph learning tasks by integrating graph structural information into language models. In this model, the Text-Graph Grounding structure is used to connect textual and graphic structures. Additionally, Dual-Stage Instruction Tuning was performed to enable the model to understand graphical data and adapt across various tasks. Moreover, graph matching tasks are addressed to enable the model to understand the inherent structural properties of graph data by providing alignment between graph tokens and language tokens. GraphGPT has implemented Step-by-Step Reasoning to understand graph data by reasoning step by step. It has also been achieved by extracting valuable information from large language models (e.g. ChatGPT) through CoT distillation and transferring it to GraphGPT. GraphGPT has demonstrated superior overall validity in both supervised and zero-shot learning tasks.

5.6 Text Summarization

Graph-based techniques have been widely applied to text summarization tasks, as detailed in Table 6. These approaches utilize graph structures to identify important information and generate concise summaries.

Zaffar et al. [14] conducted a study to test whether the integration of GNN and NER is also effective

in practice in the text summarization task. For this purpose, he proposed a summarization system that includes the stages of text pre-processing, entity recognition with NER, and graph analysis with GNN. With NER, important entities in the text were identified and labelled. Afterward, a graph structure was created that extracts important information by modelling complex relationships in the texts. The summarization performances of GNN and NER-based approaches and large language models such as GPT were compared. It has been stated that the proposed approach is effective in terms of content coverage of summaries, coherence, conciseness, and focus on main themes.

Huang et al. [91] presented research on text summarization, an information compression technology to extract important information from long texts. The study proposes a gated attention-focused GNN model, GA-GNN, which aims to effectively improve the accuracy and readability of the text summarization process. Firstly, the input text is converted into a sentence encoding vector and a document vector containing global and local semantic information using a hierarchical encoder (Bi-LSTM and CNN). Secondly, the sentence vectors are converted into independent graph data structures and fed into GA-GNN, thereby updating the feature information of nodes and reducing the accumulation of local irrelevant information in the iteration process. Then, feature extraction is performed by gating the attention unit (GAU), which is a key component of the GA-GNN model and is formed by adding an attention gate based on GRU. Finally, decoding is performed using the LSTM-ATT network with an attention mechanism, and the text summary is generated. Experimental validation is conducted on CNN/Daily Mail and MR datasets, and the results show that the model in this paper outperforms existing methods.

In their study Onan et al. [112] introduce the Multi-Element Contextual Hypergraph Extractive Summarizer (MCHES), a novel approach to extractive text summarization. MCHES leverages contextual hypergraph networks to model complex relationships between sentences, entities, and discourse elements within a document. By incorporating multi-element interactions, MCHES enhances the capture of semantic and contextual information, leading to more coherent and informative summaries. The experimental results demonstrate that MCHES outperforms traditional summarization models, particularly in handling documents with intricate structures and diverse content

Nourbakhsh et al. [92] examined the text summarization method used to extract important details from large amounts of text data. In the study, a method called Contextualized HGNN for Extractive Text Summarization (ConHGNN-SUM) was proposed. The model constructs a heterogeneous graph with two types of nodes: basic semantic nodes (e.g., words) and supernodes (e.g., sentences). Each supernode is connected to the underlying base nodes, allowing higher-level discourse nodes to connect via intermediate underlying nodes. Nodes are structured using GAN Edge Features (GATe). The heterogeneous graph layer updates the representations of nodes by facilitating message passing between sentence and word nodes using GAT. Then, all sentences in the graph are sorted and subjected to a Bi-LSTM model to identify the most suitable sentences based on their features. Finally, the representations of sentence nodes are extracted and predictions are made for summary labels.

Yu et al. [93] proposed a scheme called KGCPN based on a Knowledge Graph-Based Multi-Document Text Summarization Auto-Generation Scheme for text summarization in a mobile edge computing (MEC) environment. In the proposed method, text data is processed using NER techniques to create a knowledge graph. The knowledge graph was strengthened using GCN and semantic representations of entities and relationships in the text were learned. Semantic representations from GCN were then fed into the PGN model and multi-document summarization was performed. Experimental results show that the model can effectively facilitate intelligent preprocessing and integration of MEC data.

Etaiwi et al. [94] presented a graph-based approach to summarizing Arabic text. In this approach called SemG-TS (Semantic Graph-Based Text Summarization), a dataset consisting of news articles collected from the AlJazeera.net website was used. First, they represented the original text as a semantic graph and then obtained structural information using the graph embedding method. In this way, they produced text summaries using DNNs. The results obtained show that the SemG-TS model performs well in terms of readability and overall satisfaction.

Jiang et al. [95] proposed a framework called Graph-Based Topic Aware Abstract Text Summarization (GTASum) that can capture global semantic information. It was emphasized that the proposed model summarizes by preserving global

semantic information, unlike existing models. In the study, sentence representations were obtained with BERT. In addition, potential topics were modelled with the Neural Topic Model (NTM). A heterogeneous graph was created using sentence representations and topics obtained with NTM. The aim here is to update local and global meanings together and to reduce the semantic fragmentation problem. Finally, this modelling was done with a Graph Attention NEtwork (GAT) and text summaries were obtained with a transformer-based decoder. GTASum, CNN/DailyMail, and XSum datasets were used to evaluate the model result. It was claimed that the model performance evaluated with the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metric outperformed many existing summarization models.

Chen et al. [96] propose a new entity-guided method named RHGNNSumAbs to produce more accurate and meaningful summaries by using both inter-sentence relationships and non-textual information. In the proposed model, the knowledge problem and the inter-sentence relationship problem are considered together. First, the R-HGNN model is used to model sentence-entity relationships and the sentence-entity graph is obtained. The obtained entities are connected with the knowledge graph (KG). The summarization process is carried out in two stages. First, the important sentences and important entities are selected simultaneously with the multi-task model. Second, the selected sentences and entities are compressed. The experimental results with CNN/DM and NYT50 datasets show that it outperforms the inferential and summarizing basic models such as HGNN-based HGNNSum and entity-oriented SENECA.

Ghadimi et al. [97] proposed an extractive multi-document summarization method called SGCSumm. SGCSUMM guarantees a minimum bound on performance when summarizing extractive text and provides a method that supports sentence embedding and graph structure-aware feature learning. The method first uses Deep submodular network (DSN) [46] to maximize the utility of submodules to optimize the evaluation of summarized content. Then, modern sentence embedding techniques are used to learn deep semantic representations of sentences in documents. Then, a graph is constructed to model the relationships between documents. Important features and relationships between documents are analysed using GCN. SGCSUMM guarantees a minimum bound on performance and exhibits comparable performance to current summarization methods in experimental evaluations.

Zhao et al. [98], developed a GNN-based extractive summarization model that aims to identify sentences that contain the most essential information. In this study, a heterogeneous graph structure is designed to represent the relationships between sentences, words, and topics. The initial representation of topic nodes is obtained using the neural topic model (NTM). The initial representation of words and sentences is obtained from the text vectorization layer. In addition, GAT is used to collect the word and latent topic information in the graph structure, so that the sentence feature representation contains richer and more accurate semantic information. Finally, the sentence feature representation is input to a sentence classifier to extract summary sentences.

Onan et al. [111] proposed KETGS, a knowledge-enhanced graph-based summarization model that integrates entity and discourse relations into a unified graph structure. The model combines Transformer encodings with a Graph Neural Network, enabling richer contextual and structural representation. Experimental results showed that KETGS outperforms several baseline methods on standard datasets.

Frisoni et al. [99], developed a new method by extending the pre-trained BART-based architecture to produce more meaningful and accurate summaries in the process of automatic summarization of biomedical publications. The model called COGITOERGOSUMM represents the information in the document using two types of graphs: Event Graphs and Abstract Meaning Representation (AMR) Graphs. Event graphs represent the events in biomedical texts and the interactions in these events. AMR Graphs show the meaning of the sentence at a conceptual level. These graphs provide a broader meaning inference when used together with event graphs. COGITOERGOSUMM creates a broader information structure by combining both graphs. In addition, important information is obtained with Attention Mechanism in the model. Afterward, Reinforcement Learning is used to support the model in deciding which information to preserve in the summaries. The model was evaluated with extensive tests on the CDSR dataset and it was stated that it exhibited competitive performance despite using fewer parameters.

5.7 Graph-based Information Retrieval

Graph-based approaches have shown significant promise in information retrieval tasks, as detailed in Table 7. These methods leverage graph structures to model complex relationships and improve retrieval performance.

Hu et al. [101] developed the Neural Graph Personalized Ranking (NGPR) model using graph-based methods. This model is proposed for Top-N recommendation systems. NGPR models user and item interactions over graph structures and directly incorporate this interaction information into the embedding learning process. The relationships between users and items are more effectively represented using an interaction graph, which increases the personalized ranking performance. In addition, the graph data is processed more deeply with complex structures such as Multilayer Perceptron (MLP) during the interaction modelling stage. The study proves that the NGPR model outperforms other methods by using the power of graph-based methods in Top-N ranking tasks with experiments conducted on three different datasets. This model significantly improves the ranking accuracy by enabling more effective use of user and item interactions through graph-based approaches.

In the study, [102] a GNN framework is proposed to generate personalized recommendation rankings by considering social networks and user-item interactions. Using GCN, the authors learn more robust representations by integrating interactions between users and items with social networks. This approach combines social connections and interaction information, resulting in more accurate ranking results. The study presents experimental results showing that the model outperforms traditional recommendation systems in personalized ranking tasks. Thus, by combining social networks and interaction data, more meaningful and accurate recommendations are provided to users.

In their study, Keyhanipour [103], propose to use deep learning methods to solve the problem of graph-based rank aggregation. The aim of the study is to combine ranking information from different sources and transform them into a more accurate overall ranking. This problem is especially important when combining ranking results produced by different algorithms or systems. The authors propose a new method developed with deep learning techniques to provide better integration of these rankings by using graph structures for ranking aggregation. This approach aims to obtain more consistent and accurate rankings by modelling the relationships between ranking results. The study highlights the effectiveness of graph-based deep learning methods in the field of ranking aggregation.

In their study, Meng et al. [104] aim to improve the ranking of objects by proposing a preference learning method based on GNN. This method uses graph structures to learn information about the relationships and interactions between objects. Basically, each object is represented as a graph node, and the connections between objects (e.g., users' preferences or similarities between objects) are modelled as edges. By working on these graph structures, GNN learns the complex relationships between objects and ensures that preferences are reflected accurately. In the study, the preference learning process focuses on understanding which objects users prefer more and which objects are close to each other. GNN is especially strong in terms of modelling connectivity and correlations in such data. In addition, GNNs learn more accurate representation vectors for each object using deep learning techniques and integrate these representations into the ranking model. As a result, this method aims to increase the accuracy of ranking objects and ranking based on user preferences.

Liu et al. [105] developed a self-supervised graph convolutional model to rank nodes in complex networks. This model combines GNN and self-supervised learning techniques to learn the relationships between nodes without labelled data. The model takes into account the connections and attributes of nodes using GCN, thus improving the ranking accuracy. Self-supervised learning allows the model to rank nodes by taking advantage of the hidden features and connections in the network structure without using its labels. This method aims to optimize the node ranking process in complex networks.

6 Conclusion

This study comprehensively examines the use of GNNs in NLP tasks. GNNs have stood out with their ability to decipher and make sense of complex contexts between texts. Recent studies have shown that GNNs provide high performance, especially with small amounts of labelled data, and can be successfully used in a wide range of NLP tasks, from social media interaction

Table 7. Summary of information retrieval studies on Graph-based NLP.

Author	Methods	Dataset	Proposed Method	Evaluation Metrics
Hu et al. [101]	GNN, Multilayer Perceptron, Interaction Modelling, Non-linear Transformation	AMusic, citeulike-a, Yelp	NGPR	HR, NDCG
Fan et al. [102]	GNN, GCN, Personalized Ranking, End-to-End Learning	Ciao and Epinions, Flixster	GraphRec+	MAE, RMSE
Keyhanipour [103]	GNN, Graph-based Rank Aggregation, DL, RankNet	MQ2007-agg and MQ2008-agg	—	Precision, Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain
Meng et al. [104]	GNN, GCN, Preference Learning, Ranking-based Loss Functions, DL	MQ2007, MQ2008, OHSUMED	—	Accuracy, Precision, Recall, F1-score, MAP, NDCG
Liu et al. [105]	GNN, GCN, Self-Supervised Learning, Node Ranking	Co-auth network, Cora network, Wiki network, Metro network	SS-GCN	MAP, NDCG, Spearman correlation coefficient
Bougouin et al. [106]	Graph-Based Ranking, PageRank	Inspec, SemEval, WikiNews, DEFT	TopicRank	Precision, Recall, F-score
Kazemi et al. [107]	TextRank, Word Weighting, Bias Embedding, Sentence Embeddings	News reportage, LIAR-PLUS	BiasedTextRank	ROUGE
Lee et al. [108]	GCN, Positional Encoding, Reading Order Contextualization	Payment, FUNSD	ROPE	F1-score
Yang et al. [109]	GCN, Attention Mechanism, Ranking Loss Function, Multi-layer propagation	Mafengwo, Weeplaces, Steam	CFAG	Recall, NDCG
He et al. [110]	Continual Learning, Transfer Learning, Joint Neural Network, Entity Recognition, Attention Mechanism, Multi-task Learning	ADE, DDI, CPR	JCBIE	Micro-F1

analysis to sentiment analysis. It has been revealed that graph-based models have a distinct advantage over traditional methods in overcoming the challenges in NLP tasks by understanding the relationships between texts in more depth. In particular, the use of GNNs in social media interaction analysis provides a better understanding of the relationships and interactions between users. This allows for more effective monitoring and analysis of the flow of information and sentiment spread on social media platforms. The success of GNNs in the field of sentiment analysis is also remarkable, as these models can more accurately detect emotional tones in texts, which offers significant advantages in many areas, from marketing strategies to customer service. These studies emphasize that GNNs can be considered an important tool in the NLP field and have the potential to be further explored and developed in the future. The impressive performance of GNNs in NLP tasks has led to further intensification and expansion of research in this area. Future research should focus on studies that evaluate the performance of GNNs on larger and more diverse datasets. This can increase the generalization capacity of models, and studies examining their adaptability to different language structures and cultures can enable GNNs to be used more effectively across various contexts. In addition, a more in-depth evaluation of the developed models in terms of reliability and explainability can provide a solid foundation for their use in industrial applications and real-world scenarios. The explainability of GNNs helps users and developers to better understand and trust model decisions. This is a critical factor, especially in high-risk applications, such as medical diagnosis

or financial forecasting. Efforts to increase efficiency in terms of energy and computational resources are also important. The complex structures and high computational requirements of GNNs can pose a significant challenge in large-scale data processing tasks. Therefore, research on energy efficiency and reducing computational costs can support the wider adoption of GNNs. In this context, the development of new algorithms and optimization techniques will contribute to making GNNs more sustainable and economical.

As a result, the potential offered by GNNs in the field of NLP provides a significant opportunity to overcome various challenges. Future research can focus on the above-mentioned proposals to further develop this potential and increase the effectiveness of GNNs in NLP tasks. In this process, an in-depth study of the theoretical and applied aspects of GNNs will play a critical role in the future evolution of NLP.

Data Availability Statement

Not applicable.

Funding

This work was supported without any funding.

Conflicts of Interest

The authors declare no conflicts of interest.

AI Use Statement

The authors declare that no generative AI was used in the preparation of this manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Patwardhan, N., Marrone, S., & Sansone, C. (2023). Transformers in the real world: A survey on nlp applications. *Information, 14*(4), 242. [CrossRef]

[2] Jia, J., Liang, W., & Liang, Y. (2023). A review of hybrid and ensemble in deep learning for natural language processing. *arXiv preprint arXiv:2312.05589*.

[3] Benslimane, S., Azé, J., Bringay, S., Servajean, M., & Mollevi, C. (2023). A text and GNN based controversy detection method on social media. *World Wide Web, 26*(2), 799–825. [CrossRef]

[4] Dara, S., Srinivasulu, C. H., Babu, C. M., Ravuri, A., Paruchuri, T., Kilak, A. S., & Vidyarthi, A. (2023). Context-Aware auto-encoded graph neural model for dynamic question generation using NLP. *ACM transactions on asian and low-resource language information processing*. [CrossRef]

[5] Abdalla, H. I., Amer, A. A., & Ravana, S. D. (2023). BoW-based neural networks vs. cutting-edge models for single-label text classification. *Neural Computing and Applications, 35*(27), 20103-20116. [CrossRef]

[6] Wang, H., Ren, C., & Yu, Z. (2024). Multimodal sentiment analysis based on cross-instance graph neural networks. *Applied Intelligence, 54*(4), 3403-3416. [CrossRef]

[7] Mehta, S., Karwa, R., Chavan, R., Khatavkar, V., & Joshi, A. (2024). Keyphrase extraction using graph-based statistical approach with NLP patterns. *Sādhanā, 49*(2), 170. [CrossRef]

[8] Liu, X., Qin, X., Xu, C., & Li, H. (2025). A knowledge-enhanced model with syntactic-aware attentive graph convolutional network for biomedical entity and relation extraction. *International Journal of Machine Learning and Cybernetics, 16*(1), 583-598. [CrossRef]

[9] Martínez-Cruz, R., Mahata, D., López-López, A. J., & Portela, J. (2025). Enhancing keyphrase extraction from long scientific documents using graph embeddings. *Applied Intelligence, 55*(7), 1-18. [CrossRef]

[10] Ahmed, H., Traore, I., Mamun, M., & Saad, S. (2023). Text augmentation using a graph-based approach and clonal selection algorithm. *Machine Learning with Applications, 11*, 100452. [CrossRef]

[11] Abadal, S., Jain, A., Guirado, R., López-Alonso, J., & Alarcón, E. (2021). Computing graph neural networks: A survey from algorithms to accelerators. *ACM Computing Surveys (CSUR), 54*(9), 1-38. [CrossRef]

[12] Chi, X., & Xiang, Y. (2021). Augmenting paraphrase generation with syntax information using graph convolutional networks. *Entropy, 23*(5), 566. [CrossRef]

[13] Guo, Q., Qiu, X., Xue, X., & Zhang, Z. (2021). Syntax-guided text generation via graph neural network. *Science China Information Sciences, 64*(5), 152102. [CrossRef]

[14] Khan, I. Z., Sheikh, A. A., & Sinha, U. (2024). Graph neural network and ner-based text summarization. *arXiv preprint arXiv:2402.05126*.

[15] Hua, J., Sun, D., Hu, Y., Wang, J., Feng, S., & Wang, Z. (2024). Heterogeneous graph-convolution-network-based short-text classification. *Applied Sciences, 14*(6), 2279. [CrossRef]

[16] Das, S., Tariq, A., Santos, T., Kantareddy, S. S., & Banerjee, I. (2023). Recurrent neural networks

(RNNs): architectures, training tricks, and introduction to influential research. *Machine learning for Brain disorders*, 117-138. [CrossRef]

[17] Mishra, B. K., & Jain, S. (2025). Word sense disambiguation for Indic language using Bi-LSTM. *Multimedia Tools and Applications, 84*(16), 16631-16656. [CrossRef]

[18] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems, 32*(1), 4-24. [CrossRef]

[19] Tian, M., Ma, K., Wu, Q., Qiu, Q., Tao, L., & Xie, Z. (2024). Joint extraction of entity relations from geological reports based on a novel relation graph convolutional network. *Computers & Geosciences, 187*, 105571. [CrossRef]

[20] He, H., Wang, H., Ma, H., Liu, X., Jia, Y., & Gong, G. (2020, October). Research on short-term power load forecasting based on Bi-GRU. In *Journal of Physics: Conference Series* (Vol. 1639, No. 1, p. 012017). IOP Publishing. [CrossRef]

[21] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., ... & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI open, 1*, 57-81. [CrossRef]

[22] Waikhom, L., & Patgiri, R. (2021). Graph neural networks: Methods, applications, and opportunities. *arXiv preprint arXiv:2108.10733*.

[23] Li, Y., Wu, J., & Luo, X. (2024). BERT-CNN based evidence retrieval and aggregation for Chinese legal multi-choice question answering. Neural Computing and Applications, 36(11), 5909–5925.[CrossRef]

[24] Phan, H. T., Nguyen, N. T., & Hwang, D. (2023). Aspect-level sentiment analysis: A survey of graph convolutional network methods. *Information Fusion, 91*, 149-172. [CrossRef]

[25] Khemani, B., Patil, S., Kotecha, K., & Tanwar, S. (2024). A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. *Journal of Big Data, 11*(1), 18. [CrossRef]

[26] Sun, Q., Zhang, K., Huang, K., Xu, T., Li, X., & Liu, Y. (2023). Document-level relation extraction with two-stage dynamic graph attention networks. *Knowledge-Based Systems, 267*, 110428. [CrossRef]

[27] Maskey, S., Levie, R., Lee, Y., & Kutyniok, G. (2022). Generalization analysis of message passing neural networks on large random graphs. *Advances in neural information processing systems, 35*, 4805-4817.

[28] Tang, M., Li, B., & Chen, H. (2023). Application of message passing neural networks for molecular property prediction. *Current Opinion in Structural Biology, 81*, 102616. [CrossRef]

[29] Liu, Y., Xing, X., Jia, Z., & Li, Y. (2023, December). Enhancing Social Recommendation with Global Dependency Modeling base on Self-Attention Graph Neural Network. In *2023 4th International Conference on Computer, Big Data and Artificial Intelligence (ICCBD+ AI)* (pp. 350-354). IEEE. [CrossRef]

[30] Marino, A., Pacchierotti, C., & Giordano, P. R. (2024). Input state stability of gated graph neural networks. *IEEE Transactions on Control of Network Systems, 11*(4), 2052-2063.[CrossRef]

[31] Zheng, Y., Yi, L., & Wei, Z. (2024). A survey of dynamic graph neural networks. *arXiv preprint arXiv:2404.18211*.

[32] Zhang, Y. (2024). A General Benchmark Framework is Dynamic Graph Neural Network Need. *arXiv preprint arXiv:2401.06559*.

[33] Jin, G., Liang, Y., Fang, Y., Shao, Z., Huang, J., Zhang, J., & Zheng, Y. (2023). Spatio-temporal graph neural networks for predictive learning in urban computing: A survey. *IEEE transactions on knowledge and data engineering, 36*(10), 5388-5408. [CrossRef]

[34] Ju, W., Wang, Y., Qin, Y., Mao, Z., Xiao, Z., Luo, J., Yang, J., Gu, Y., Wang, D., Long, Q., & others. (2024). Towards Graph Contrastive Learning: A Survey and Beyond. *arXiv preprint arXiv:2405.11868*.

[35] Zhao, T., Jin, W., Liu, Y., Wang, Y., Liu, G., Günnemann, S., Shah, N., & Jiang, M. (2022). Graph data augmentation for graph machine learning: A survey. *arXiv preprint arXiv:2202.08871*.

[36] Ghojogh, B., & Ghodsi, A. (2024). Graph Neural Network, ChebNet, Graph Convolutional Network, and Graph Autoencoder: Tutorial and Survey.

[37] Behrouzi, T., & Hatzinakos, D. (2022). Graph variational auto-encoder for deriving EEG-based graph embedding. *Pattern Recognition, 121*, 108202. [CrossRef]

[38] Ding, Y., Tian, L. P., Lei, X., Liao, B., & Wu, F. X. (2021). Variational graph auto-encoders for miRNA-disease association prediction. *Methods, 192*, 25-34. [CrossRef]

[39] Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, August). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 701-710). [CrossRef]

[40] Xu, M. (2021). Understanding graph embedding methods and their applications. *SIAM Review, 63*(4), 825–853. [CrossRef]

[41] Jeyaraj, R., Balasubramaniam, T., Balasubramaniam, A., & Paul, A. (2024). DeepWalk with Reinforcement Learning (DWRL) for node embedding. *Expert Systems with Applications, 243*, 122819. [CrossRef]

[42] Grover, A., & Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 855-864). [CrossRef]

[43] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, 1067–1077. [CrossRef]

[44] Ju, W., Fang, Z., Gu, Y., Liu, Z., Long, Q., Qiao, Z., ... & Zhang, M. (2024). A comprehensive survey on deep graph representation learning. *Neural Networks, 173*, 106207. [CrossRef]

[45] Thambi, S. V., & Reghu Raj, P. C. (2024). A novel technique using graph neural networks and relevance scoring to improve the performance of knowledge graph-based question answering systems. *Journal of Intelligent Information Systems, 62*(3), 809-832. [CrossRef]

[46] Kau, A., He, X., Nambissan, A., Astudillo, A., Yin, H., & Aryani, A. (2024). Combining Knowledge Graphs and Large Language Models. *arXiv preprint arXiv:2407.06564*.

[47] Simcharoen, S. (2022). *Cooperatively Managing and Exploiting Distributed Co-occurrence Graphs* (Doctoral dissertation, Dissertation, FernUniversität in Hagen, 2022).

[48] Zhou, J. (2023). *Generating Semantic Graphs for Natural Language* (Doctoral dissertation, Harvard University).

[49] Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

[50] Nikolentzos, G., Siglidis, G., & Vazirgiannis, M. (2021). Graph kernels: A survey. *Journal of Artificial Intelligence Research, 72*, 943–1027. [CrossRef]

[51] Kriege, N. M., Johansson, F. D., & Morris, C. (2020). A survey on graph kernels. *Applied Network Science, 5*(1), 6. [CrossRef]

[52] Wang, Y., Wang, C., Zhan, J., Ma, W., & Jiang, Y. (2023). Text FCG: Fusing contextual information via graph learning for text classification. *Expert Systems with Applications, 219*, 119658. [CrossRef]

[53] Cui, H., Wang, G., Li, Y., & Welsch, R. E. (2022). Self-training method based on GCN for semi-supervised short text classification. *Information Sciences, 611*, 18-29. [CrossRef]

[54] Shi, Y., Xiao, Y., Quan, P., Lei, M., & Niu, L. (2021). Document-level relation extraction via graph transformer networks and temporal convolutional networks. *Pattern Recognition Letters, 149*, 150-156. [CrossRef]

[55] Li, M., Xie, Y., Yang, W., & Chen, S. (2023). Multistream BertGCN for Sentiment Classification Based on Cross-Document Learning. *Quantum Engineering, 2023*(1), 3668960. [CrossRef]

[56] Wu, Y., Wang, X., Zhao, W., & Lv, X. (2023). A novel topic clustering algorithm based on graph neural network for question topic diversity. *Information Sciences, 629*, 685-702. [CrossRef]

[57] Liang, Y., Meng, F., Zhang, Y., Chen, Y., Xu, J., & Zhou, J. (2022). Emotional conversation generation with heterogeneous graph neural network. *Artificial Intelligence, 308*, 103714. [CrossRef]

[58] Bhadra, J., Khanna, A. S., & Beuno, A. (2023, April). A Graph Neural Network Approach for Identification of Influencers and Micro-Influencers in a Social Network:* Classifying influencers from non-influencers using GNN and GCN. In *2023 International Conference on Advances in Electronics, Communication, Computing and Intelligent Information Systems (ICAECIS)* (pp. 66-71). IEEE. [CrossRef]

[59] Yang, S., Liu, Y., Zhang, Y., & Zhu, J. (2023). A word-concept heterogeneous graph convolutional network for short text classification. *Neural Processing Letters, 55*(1), 735-750. [CrossRef]

[60] Xu, S., & Xiang, Y. (2021). Frog-GNN: Multi-perspective aggregation based graph neural network for few-shot text classification. *Expert Systems with Applications, 176*, 114795. [CrossRef]

[61] Hua, S., Li, X., Jing, Y., & Liu, Q. (2022). A semantic hierarchical graph neural network for text classification. *arXiv preprint arXiv:2209.07031*.

[62] Yang, Y., & Cui, X. (2021). Bert-enhanced text graph neural network for classification. *Entropy, 23*(11), 1536. [CrossRef]

[63] Zeng, D., Zha, E., Kuang, J., & Shen, Y. (2024). Multi-label text classification based on semantic-sensitive graph convolutional network. *Knowledge-Based Systems, 284*, 111303. [CrossRef]

[64] Ai, W., Wei, Y., Shao, H., Shou, Y., Meng, T., & Li, K. (2024). Edge-enhanced minimum-margin graph attention network for short text classification. *Expert Systems with Applications, 251*, 124069. [CrossRef]

[65] Wang, B., Liang, B., Du, J., Yang, M., & Xu, R. (2022, December). SEMGraph: Incorporating sentiment knowledge and eye movement into graph model for sentiment analysis. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (pp. 7521-7531). [CrossRef]

[66] Gu, T., Zhao, H., He, Z., Li, M., & Ying, D. (2023). Integrating external knowledge into aspect-based sentiment analysis using graph neural network. *Knowledge-based systems, 259*, 110025. [CrossRef]

[67] Lu, G., Li, J., & Wei, J. (2022). Aspect sentiment analysis with heterogeneous graph neural networks. *Information Processing & Management, 59*(4), 102953. [CrossRef]

[68] Wu, F., & Li, X. (2023). Local dependency-enhanced graph convolutional network for aspect-based sentiment analysis. *Applied Sciences, 13*(17), 9669. [CrossRef]

[69] Sunny, J., Padraig, T., Terry, R., & Ali, W. (2023). Syntactic Fusion: Enhancing Aspect-Level Sentiment Analysis Through Multi-Tree Graph Integration. *arXiv preprint arXiv:2312.03738*.

[70] Jin, Y., & Zhao, A. (2024). Bert-based graph unlinked embedding for sentiment analysis. *Complex & Intelligent Systems, 10*(2), 2627-2638. [CrossRef]

[71] An, W., Tian, F., Chen, P., & Zheng, Q. (2022). Aspect-based sentiment analysis with heterogeneous graph neural network. *IEEE Transactions on Computational Social Systems, 10*(1), 403-412. [CrossRef]

[72] Zeng, Y., Li, Z., Tang, Z., Chen, Z., & Ma, H. (2023). Heterogeneous graph convolution based on in-domain self-supervision for multimodal sentiment analysis. *Expert Systems with Applications, 213*, 119240. [CrossRef]

[73] Liu, Z., He, J., Gong, T., Weng, H., Wang, F. L., Liu, H., & Hao, T. (2024). Improving topic tracing with a textual reader for conversational knowledge based question answering. *IEEE Transactions on Emerging Topics in Computational Intelligence, 8*(3), 2640-2653. [CrossRef]

[74] Gao, P., Gao, F., Ni, J. C., Wang, Y., Wang, F., & Zhang, Q. (2024). Medical knowledge graph question answering for drug-drug interaction prediction based on multi-hop machine reading comprehension. *CAAI Transactions on Intelligence Technology, 9*(5), 1217-1228. [CrossRef]

[75] Cai, S., Ma, Q., Hou, Y., & Zeng, G. (2024). Knowledge graph multi-hop question answering based on dependent syntactic semantic augmented graph networks. *Electronics, 13*(8), 1436. [CrossRef]

[76] Li, J., Wu, S., Zhang, X., & Feng, Z. (2024). A Context-enhanced Adaptive Graph Network for Time-sensitive Question Answering. *ACM Transactions on Asian and Low-Resource Language Information Processing*. [CrossRef]

[77] Gao, Y., Qiao, L., Kan, Z., Wen, Z., He, Y., & Li, D. (2024). Two-stage Generative Question Answering on Temporal Knowledge Graph Using Large Language Models. *arXiv preprint arXiv:2402.16568*.

[78] Su, Y., Zhang, J., Song, Y., & Zhang, T. (2024). PipeNet: Question Answering with Semantic Pruning over Knowledge Graphs. *arXiv preprint arXiv:2401.17536*.

[79] De Cao, N., Aziz, W., & Titov, I. (2019, June). Question answering by reasoning across documents with graph convolutional networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human language technologies, Volume 1* (long and short papers) (pp. 2306-2317). [CrossRef]

[80] Zou, S., Liu, Z., Wang, K., Cao, J., Liu, S., Xiong, W., & Li, S. (2024). A study on pharmaceutical text relationship extraction based on heterogeneous graph neural networks. *Mathematical Biosciences and Engineering, 21*(1), 1489–1507. [CrossRef]

[81] Khanfir, Y., Dhiaf, M., Ghodhbani, E., Rouhou, A. C., & Kessentini, Y. (2024, January). Graph Neural Networks for End-to-End Information Extraction from Handwritten Documents. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (pp. 493-501). IEEE. [CrossRef]

[82] Liu, F., Zhang, X., & Liu, Q. (2023). An emotion-aware approach for fake news detection. *IEEE Transactions on Computational Social Systems, 11*(3), 3516-3524. [CrossRef]

[83] Hu, Z., Dong, Y., Wang, K., Chang, K. W., & Sun, Y. (2020, August). Gpt-gnn: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1857-1867). [CrossRef]

[84] Pi, Q., Lu, J., Zhu, T., Sun, Y., Li, S., & Guo, J. (2024). Enhancing cross-evidence reasoning graph for document-level relation extraction. *PeerJ Computer Science, 10*, e2123. [CrossRef]

[85] Du, C., Li, Y., & Wen, M. (2023). G-DCS: GCN-Based Deep Code Summary Generation Model. *Journal of Internet Technology, 24*(4), 965–973.

[86] Lai, P., Ye, F., Fu, Y., Chen, Z., Wu, Y., Wang, Y., & Chang, V. (2024). CogNLG: Cognitive graph for KG-to-text generation. *Expert Systems, 41*(1), e13461. [CrossRef]

[87] Wang, C., van Noord, R., & Bos, J. (2023). Controlling Topic-Focus Articulation in Meaning-to-Text Generation using Graph Neural Networks. *arXiv preprint arXiv:2310.02053*.

[88] Lou, X., Liu, G., & Li, J. (2024). Heterogeneous graph neural network with graph-data augmentation and adaptive denoising. Applied Intelligence, 54(5), 4411–4424. Springer. [CrossRef]

[89] Chen, W., Su, Y., Yan, X., & Wang, W. Y. (2020). KGPT: Knowledge-grounded pre-training for data-to-text generation. *arXiv preprint arXiv:2010.02307*.

[90] Tang, J., Yang, Y., Wei, W., Shi, L., Su, L., Cheng, S., Yin, D., & Huang, C. (2023). Graphgpt: Graph instruction tuning for large language models. *arXiv preprint arXiv:2310.13023*.

[91] Huang, J., Wu, W., Li, J., & Wang, S. (2023). Text summarization method based on gated attention graph neural network. Sensors, 23(3), 1654. MDPI. [CrossRef]

[92] Nourbakhsh, S. E., & Kashani, H. B. (2024). ConHGNN-SUM: A Contextualized Heterogeneous Graph Neural Network for Extractive Text Summarization. In *2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP)*, 1–6. IEEE. [CrossRef]

[93] Yu, Z., Wu, S., Jiang, J., & Liu, D. (2024). A knowledge-graph based text summarization scheme for mobile edge computing. *Journal of Cloud Computing, 13*(1), 9. [CrossRef]

[94] Etaiwi, W., & Awajan, A. (2022). SemG-TS: Abstractive arabic text summarization using

semantic graph embedding. *Mathematics, 10*(18), 3225. MDPI. [CrossRef]

[95] Jiang, M., Zou, Y., Xu, J., & Zhang, M. (2022). Gatsum: graph-based topic-aware abstract text summarization. *Information Technology and Control, 51*(2), 345–355. [CrossRef]

[96] Chen, J. (2024). An entity-guided text summarization framework with relational heterogeneous graph neural network. *Neural Computing and Applications, 36*(7), 3613–3630. Springer. [CrossRef]

[97] Ghadimi, A., & Beigy, H. (2023). SGCSumm: An extractive multi-document summarization method based on pre-trained language model, submodularity, and graph convolutional neural networks. *Expert Systems with Applications, 215*, 119308. [CrossRef]

[98] Zhao, H., Zhang, W., Huang, M., Feng, S., & Wu, Y. (2023). A multi-granularity heterogeneous graph for extractive text summarization. *Electronics, 12*(10), 2184. MDPI. [CrossRef]

[99] Frisoni, G., Italiani, P., Salvatori, S., & Moro, G. (2023, June). Cogito ergo summ: abstractive summarization of biomedical papers via semantic parsing graphs and consistency rewards. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 37, No. 11, pp. 12781-12789). [CrossRef]

[100] Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., Wu, C., Croft, W. B., & Cheng, X. (2020). A deep look into neural ranking models for information retrieval. *Information Processing & Management, 57*(6), 102067. Elsevier. [CrossRef]

[101] Hu, Z., Wang, J., Yan, Y., Zhao, P., Chen, J., & Huang, J. (2021). Neural graph personalized ranking for Top-N Recommendation. *Knowledge-Based Systems, 213*, 106426. Elsevier. [CrossRef]

[102] Fan, W., Ma, Y., Li, Q., Wang, J., Cai, G., Tang, J., & Yin, D. (2020). A graph neural network framework for social recommendations. *IEEE Transactions on Knowledge and Data Engineering, 34*(5), 2033–2047. IEEE. [CrossRef]

[103] Keyhanipour, A. H. (2025). Graph-based rank aggregation: a deep-learning approach. *International Journal of Web Information Systems, 21*(1), 54-76. [CrossRef]

[104] Meng, Z., Lin, R., & Wu, B. (2024). Graph neural networks-based preference learning method for object ranking. *International Journal of Approximate Reasoning, 167*, 109131. Elsevier. [CrossRef]

[105] Liu, C., Cao, T., & Zhou, L. (2022). Learning to rank complex network node based on the self-supervised graph convolution model. *Knowledge-Based Systems, 251*, 109220. Elsevier. [CrossRef]

[106] Bougouin, A., Boudin, F., & Daille, B. (2013). Topicrank: Graph-based topic ranking for keyphrase extraction. In *International joint conference on natural language processing* (*IJCNLP*), 543–551.

[107] Kazemi, A., Pérez-Rosas, V., & Mihalcea, R. (2020). Biased TextRank: Unsupervised graph-based content extraction. *arXiv preprint arXiv:2011.01026*.

[108] Lee, C. Y., Li, C. L., Wang, C., Wang, R., Fujii, Y., Qin, S., ... & Pfister, T. (2021). Rope: reading order equivariant positional encoding for graph-based document information extraction. *arXiv preprint arXiv:2106.10786*.

[109] Yang, M., Liu, Z., Yang, L., Liu, X., Wang, C., Peng, H., & Yu, P. S. (2023). Ranking-based group identification via factorized attention on social tripartite graph. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 769–777. [CrossRef]

[110] He, K., Mao, R., Gong, T., Cambria, E., & Li, C. (2022). JCBIE: a joint continual learning neural network for biomedical information extraction. *BMC bioinformatics, 23*(1), 549. [CrossRef]

[111] Onan, A., & Alhumyani, H. (2024). Knowledge-Enhanced Transformer Graph Summarization (KETGS): Integrating Entity and Discourse Relations for Advanced Extractive Text Summarization. *Mathematics, 12*(23), 3638. [CrossRef]

[112] Onan, A., & Alhumyani, H. (2024). Contextual hypergraph networks for enhanced extractive summarization: Introducing multi-element contextual hypergraph extractive summarizer (MCHES). *Applied Sciences, 14*(11), 4671. [CrossRef]

[113] Onan, A. (2023). GTR-GA: Harnessing the power of graph-based neural networks and genetic algorithms for text augmentation. *Expert systems with applications, 232*, 120908. [CrossRef]

[114] Onan, A. (2023). Hierarchical graph-based text classification framework with contextual node embedding and BERT-based dynamic fusion. *Journal of king saud university-computer and information sciences, 35*(7), 101610. [CrossRef]

Tuğba ÇELİKTEN received the M.Sc. degree in software engineering from Manisa Celal Bayar University, Manisa, Turkey, in 2023. She is currently pursuing her Ph.D. degree in computer engineering at İzmir Katip Çelebi University, İzmir, Turkey. Her research interests include artificial intelligence, natural language processing, text mining and language models. She has contributed to national research projects and serves as a reviewer for scientific conferences in the field of artificial intelligence. (Email: tugba.celikten@cbu.edu.tr)

Aytuğ ONAN (Editor-in-Chief, ICCK) received the Ph.D. degree in computer engineering from Ege University, İzmir, Turkey, in 2016. Since 2024, he has been serving as a professor at the Department of Computer Engineering, İzmir Institute of Technology, Turkey. His research interests include artificial intelligence, data science, natural language processing, text mining and large language models. He has published numerous articles in international journals and conferences and actively participates in national and international research projects. (Email: aytugonan@iyte.edu.tr)