A Survey of Automatic Text Summarization using Graph Neural Networks

Marco Ferdinand Salchner
marco.salchner@student.uibk.ac.at
Department of Computer Science,
University of Innsbruck, Austria

Adam Jatowt
adam.jatowt@uibk.ac.at
Department of Computer Science &
Digital Science Center,
University of Innsbruck, Austria

Abstract

Although automatic text summarization (ATS) has been researched for several decades, the application of graph neural networks (GNNs) to this task started relatively recently. In this survey we provide an overview on the rapidly evolving approach of using GNNs for the task of automatic text summarization. In particular we provide detailed information on the functionality of GNNs in the context of ATS, and a comprehensive overview of models utilizing this approach.

1 Introduction

The advent of the internet has led to an explosion in the amount of textual information available online. The extensive availability of textual information paired with a need to quickly understand it has led to major efforts in the field of automatic text summarization (ATS). For a comprehensive review and survey of ATS as a task we recommend a recent survey by El-Kassas et al. (2021).

The goal of ATS is to produce a concise, correct and fluent summary of a given text. Although this definition is intuitively understandable, there is no commonly agreed upon formal definition for these qualities. This is in part due to the difficulty of the task itself as producing a summary with the above properties is challenging, even for humans.

Although the field of ATS has made major steps forward, generally, we differentiate between two basic approaches to the task of ATS. The first being extractive ATS and the second being abstractive ATS. Extractive ATS involves extracting text spans from the original document such that a summary is generated. Thus the main challenge consists in identifying useful spans of text from the original document. This approach is popular as it eliminates the non-trivial task of generating factually correct and coherent sentences. Abstractive ATS on the other hand involves the challenging task of generating novel sentences for a summary. Despite impressive advancements, generating factually correct and fluent sentences is still a major challenge in ATS (Kryściński et al., 2020). In addition to the above, one differentiates between the task of single-document and multi-document summarization.

1.1 Why graph neural networks ?

Contemporary solutions to the task of ATS suffer from a number of issues, chiefly an inconsistent evaluation protocol and, somewhat, a lack of progress, as noted by Kryściński et al. (2019). In recent years GNNs have been successfully applied to a number of downstream NLP tasks such as classification (Liu et al., 2020b) (Zhang et al., 2020b) and translation (Xu et al., 2021) (Yin et al., 2020). Although GNNs may not be able to solve all problems related to the task of ATS, we believe that they can at least give a new perspective to this task. Generally GNNs bring a number of advantages to ATS which we believe to be significant enough to warrant further research, and this survey. In particular we want to highlight the following aspects of GNNs:

- **Scalability and Flexibility**. A vast number of ATS models are based on **BERT** (Devlin et al., 2019). However, the computational complexity of **BERT**-based ATS models grows quadratic with the input length; due to the self-attention operation. This fact renders them impractical for long, or even medium sized text documents. Recently some work has been done in order to circumvent this limiting factor (Ding et al., 2020) (Zhang et al., 2021). In contrast, GNNs can scale by their nature to graphs of thousands of nodes and more. This is in part due to the linear scaling of the memory cost with regards to the input size. The total memory cost of a GNN model depends on the size of the graph, the number of layers and the feature vector size of the nodes present. Formally, for L layers and an input

of N nodes with each node's feature vector being of size H the memory complexity is $O(LNH)$. But even for very large graphs on the scale of millions of nodes one can utilize GNNs. This can be achieved using methods such as neighbour sampling or distributing the graph over multiple GPUs, as done for example by Jia et al. (2020b). We recommend the paper by Li et al. (2021) for insights as to how one can train large and very deep GNNs. As the input of a GNN is a graph, the input can vary in size, therefore GNNs are also able to cope with changing text sizes and structures. Both of these aspects combined allow GNNs to produce summaries which are not restricted by hard-coded limits related to input or output size.

- **Understanding and Explainability**. It is often difficult to understand *why* a model arrived at a certain conclusion. Additionally it is often difficult to see *how* the model aggregates information. This is not the case with GNNs, as with the help of methods such as **GNN Explainer** (Ying et al., 2019) one can understand which nodes were used by the model to reach its output. This removes a layer of the *black-box magic* present in many current non-GNN models. We recommend the survey by Yuan et al. (2020) for an overview of methods for generating explanations for GNNs.

1.2 Related Surveys

As the application of GNNs to ATS is rather novel, to the best of our knowledge there is only one survey on the topic. The survey by Luo et al. (2020) gives an introduction to the topic. However, it does not provide much detail on GNNs in the context of ATS, nor does it cover all models in the space with a taxonomy.

As for GNNs themselves, there exists a large number of surveys on GNNs as a technology. In particular, we want to highlight the surveys by Wu et al. (2020b) and Zhou et al. (2020) which provide a general overview on GNNs and their applications in different fields. The survey by Abadal et al. (2021) provides more technical and theoretical details on GNNs. The survey by Wu et al. (2021a) is of particular interest as it focuses on the usage of GNNs for NLP as a domain. Additionally, we want to note here the survey by Wu et al. (2020a) on the usage of GNNs in recommender systems and the review by Malekzadeh et al. (2021) on the usage of GNNs for text classification.

For a more theoretical approach we point the reader to the analysis by Xu et al. (2018) which establishes a number of important theoretical properties for GNNs. An analysis of the Vapnik–Chervonenkis (VC) dimension of GNNs was performed by Veličković et al. (2018).

In general, the contributions of our survey are as follows:

1. We provide a detailed explanation of GNNs in the context of ATS.
2. We introduce a simple taxonomy for GNN models used for ATS.
3. We provide a comprehensive overview of innovative GNN models for ATS and discuss further directions for future research.

The rest of survey is structured as follows. First we will give a comprehensive explanation of GNNs in the context of ATS. Next, we will explore a number of interesting and innovative models. Finally, we will finish with a conclusion and an outlook on the future usage of GNNs for ATS.

2 Graph Neural Networks

Graph based methods for ATS are not a new innovation, with methods such as TextRank (Mihalcea and Tarau, 2004) being first presented in the early 2000s. In fact, the core idea of early graph-based approaches such as TextRank is similar to the one used by GNNs designed for the ATS task. The idea is to encode the structural and semantic information contained within a text document into an explicit form with the help of a graph.

Deep learning has by now become a common tool for solving tasks throughout many domains, with various end-to-end paradigms such as recurrent-neural-networks (RNNs) or convolutional-neural-networks (CNNs) emerging as versatile and powerful tools. In particular deep learning has shown extraordinary power for data which is *Euclidean* in nature. As an example, images can be represented as regular grids in Euclidean space. Using such a representation CNNs are able to extract meaningful local representations with the help of convolution. However, for many domains and applications such a representation is often either inconvenient or not even directly possible. The most obvious example being the field of

chemistry in which molecules should be modeled as graphs. The complexity of translating such complex data into existing deep learning paradigms has led, in part, to the development of GNNs which attempt to leverage the power of deep learning for *non-Euclidean* data.

2.1 Defining the Graph

The first step of developing a GNN involves defining and designing the graph structure to be used. Formally a graph is defined as $G = (V, E)$ where V is a set of nodes, and E a set of edges. Let $v_i \in V$ denote a node; then each edge $e \in E$ is defined as $e_{i,j} = (v_i, v_j)$, that is pointing from node v_i to node v_j. In addition to that for a graph with $n = |V|$ nodes and $m = |E|$ edges a feature matrix $X \in R^{n \times d}$ is defined where each node i carries a feature vector $x_i \in R^d$. Note that d denotes the dimension of the feature vector. This encodes the information for the structure represented by the node. Often it is also important to encode specific information for the edge of two nodes. For this we additionally define an edge feature matrix X^e where $X^e \in R^{m \times c}$ with each edge e between nodes i and j carrying an edge feature vector $w_{i,j} \in R^c$ where c denotes the dimension of the edge feature vector.

Using the above definitions there are two common scenarios, either the data is inherently structured or inherently unstructured. In the first case, a direct translation into the above structure is possible, although additional information may be encoded by the designer. For example in knowledge graphs or molecule simulations such a direct translation would be possible, as the data itself already forms a valid graph, only the feature vectors would have to be engineered in an appropriate way. In the second scenario the data implicitly contains a graph-like structure, as is the case for ATS. Natural language text contains structure but that structure is not directly available as a graph. Simplistically one could consider text as a linear graph, that is each sentence is a node and all sentences following each other are connected with edges. However, this would not expose all the possible information hidden, as text implicitly contains much structural and semantic information, both of which can explicitly be modeled with the help of a graph.

A common scenario for ATS is that the text is encoded into the graph using word nodes and sentence nodes. The above definition does not allow such an option as it assumes an homogeneous graph, that is each node's feature vector is equal in dimension. However, for ATS and many other tasks heterogeneous graphs are more common. This is due to the fact that heterogeneous graphs allow to include different types of structures within the same graph. However, all of this is not an issue as one can define a feature matrix X for each type of node that exists within the graph. In the case of ATS with word and sentence type nodes one would define $X_w \in R^{m \times d_w}$ as the feature matrix for all word nodes and $X_s \in R^{n \times d_s}$ as the feature matrix for all sentence nodes with d_w being the dimension of the word node feature vectors, and d_s representing the dimension of sentence node feature vectors.

For ATS it is also sometimes useful to utilize the design space of *directed vs. undirected* edges. The choice of *directed vs. undirected* edges allows to explicitly encode structural information. For instance, one can explicitly encode the order of words or sentences with directed edges.

2.2 Defining the Neural Network

There are three ways in which GNNs differ from more conventional neural networks, the *input*, the *output* and the *way information is aggregated* within the network. The input is the above described graph.

The output and associated loss function are in the case of GNNs bound to the structure of the data i.e. the graph. Generally there are three levels of output possible, namely, on the *node level*, on the *edge level* and on the *graph level*. Both node and edge level output involve predictions or classifications on their respective components of the graph; additionally, in the case of edge level output, one can pose the task of predicting the edge itself. On the graph level one can pose the task of predicting subgraphs or perform graph segmentation. In the context of ATS, the most direct output and loss formulation consist in predicting binary inclusion labels on nodes representing sentences, or phrases, using a cross-entropy loss.

The way information is aggregated is usually done through either spatial convolution or spectral convolution. The basic idea of spatial convolution involves extending the well-known convolution operator to graph structures, whereas spectral convolution is based on graph signal processing. In recent years spatial convolution has become a

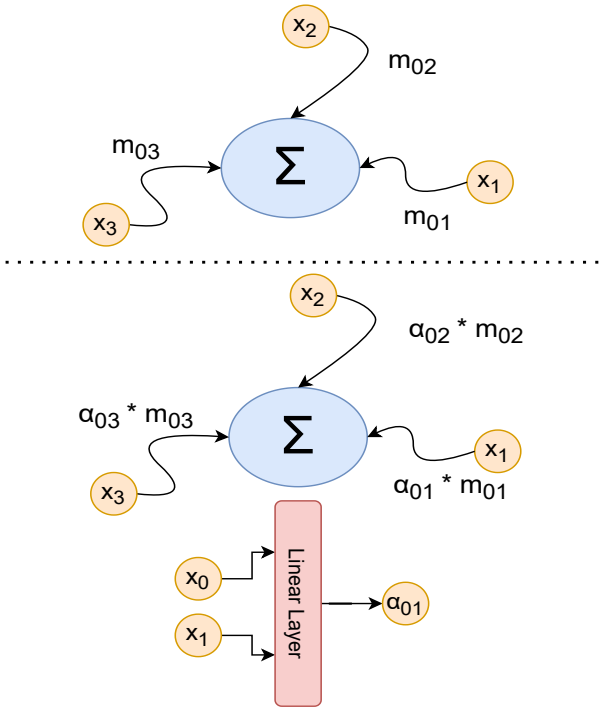

Figure 1: Comparison of spatial convolution (above) and GAT (below) over the node zero with feature vector x_0. In particular, the attention scores α in the GAT attenuate the messages received and are obtained by a learnable linear layer of the two nodes involved in the message exchange.

popular approach and is the preferred approach for GNNs for ATS due to the flexibility and efficiency it offers. More specifically, in terms of efficiency, spectral convolution involves either computations over the entire graph or eigenvector computations. None of which are necessary for spatial convolution; additionally, spatial convolution does not assume a fixed graph structure, allowing better generalization. In addition to this the locality of the spatial convolution operation also allows it to be performed in batches of nodes, instead of the entire graph. This is especially relevant for large graphs, or, in the context of this paper, large input texts.

The setup described thus far implicitly assumes that the GNN performs extractive summarization. This is in fact the case and for what we will later define as *standalone GNNs* i.e. GNNs that are not part of a larger system. No purely abstractive approaches have so far been developed, to the best of our knowledge.

2.3 Spatial Convolution and Message Passing

One can view spatial convolution as used in GNNs as a generalization of the convolution used in neural networks such as CNNs. As an example, in the case of images, one can imagine 2D convolution as being applied to a regular grid of nodes where each node represents a pixel in the image. The resulting 2D convolution applied to one target node is then the weighted average of node (pixel) values of the neighbours of the target node. Generalizing this idea to a non-regular grids leads to spatial convolution. However, different to images and regular grids, in graphs, the neighbours of each target node are unordered and can vary in number and their feature vector representation. The major challenge with this extension consists therefore in dealing with the unordered and inconsistent neighbourhood sizes inherent to homogeneous and heterogeneous graphs, with an additional challenge being posed by the differing feature vector representations in heterogeneous graphs.

Directly translating the above description of convolution into a mathematical formulation leads to a valid information propagation scheme. However, such a description suffers from scalability issues due to it directly operating over the entire graph. As such modern GNNs use, what is commonly referred to as, *message passing*. In practice, this means that nodes within the graph exchange messages (perform convolutions) with their neighbours for a number of iterations. Thereby the network is able to diffuse information throughout the graph. Consequently, the more iterations, the further outwards information is propagated throughout the graph. In the terminology of CNNs one would say that the more message passing iterations, the larger the receptive field of the convolution. Formally, one can define message passing (Grattarola and Alippi, 2021) for each time step t as two equations:

$$m_{i,j}^{t+1} = \phi(x_i^t, x_j^t, w_{i,j}^t), e_{i,j} \in E \qquad (1)$$

This first equation describes how messages are generated. A differentiable function ϕ generates messages m for each edge which connects nodes using the node features and edge feature present.

$$x_i^{t+1} = \psi(x_i^t, \rho(\{m_{i,j}^{t+1} : e_{i,j} \in E\})) \qquad (2)$$

The above equation is the core of the message passing framework and describes how each node feature is updated. The first part consists in the application of a permutation-invariant reduction function ρ. This function aggregates all incoming messages to a node. Then another differen-

tiable function ψ combines the reduced messages received with the previous state. Using these two equations one can utilize message passing for learnable layers.

The convolution layer for a GNN is then defined with a learnable weight W such that the message per edge is $m_{i,j}^{t+1} = x_j^t$ and the aggregation is the normalized sum of messages, i.e. $x_i^{t+1} = \sigma(b^t + \sum_{m_{i,j} \in M(i)} \frac{1}{c_{j,i}} m_{i,j}^{t+1} W^t)$ where $M(i)$ represents the set of messages received by node i, σ is the activation function, b is the bias, and $c_{j,i}$ is an appropriate scaling factor, e.g., the square root of the node degree. Note how it is important for the reduction function, in this case a *sum* function, to be permutation-invariant as otherwise GNNs could not handle the unordered nature of graphs.

The above presented convolution layer does not allow the model to *filter* unimportant neighbours. Inspired by the attention mechanism popularized by transformer networks (Vaswani et al., 2017), graph attention networks (GAT) (Veličković et al., 2018) assign attention scores to each neighbour. A schematic depiction of the two variants of spatial convolution can be seen in Figure 1 with GAT depicted on the lower part of the figure. The introduction of attention scores to the spatial convolution allows the model to explicitly assign importance to certain nodes and their messages. Just as in transformers GAT is formulated with multi-head attention. The modification to the previously presented convolution layer follows closely the common attention formulation. Formally, $x_i^{t+1} = \|_K \sigma(b^t + \sum_{m_{i,j} \in M(i)} \alpha_{i,j} m_{i,j}^{t+1} W^t)$ where $\alpha_{i,j}$ is the attention score between node i and node j and K denotes the number of concatenated heads. The attention scores are computed with $r_{i,j} = LeakyReLU(a^T [W x_i^t \parallel W x_j^t])$. This score is then normalized to obtain the attention score per edge $\alpha_{i,j} = softmax_i(r_{i,j})$. We want to highlight here a recent development which Brody et al. (2021) simply denote as GATv2. Their main improvement aims at the fact that in the above calculation both learnable parameters a and W effectively fold into a single linear layer, thus the expressive power of the layer is less than what it could be. The *fix* introduced by GATv2 pulls the two parameters apart, thus achieving more expressive power while not increasing computational complexity. Taking the above description the attention score for GATv2 is modified as follows $r_{i,j} = a^T LeakyReLU([W x_i \parallel W x_j^t])$. In both synthetic and real datasets this modification shows superior performance, which is supported by a theoretical analysis of the authors.

There are numerous modifications and extensions to the basic convolution presented here. However, for ATS models, GAT layers are dominating as the workhorse for most models. The reasoning for their dominance can be explained by the similar success that attention transformers have had in conventional neural networks for AST. We expect that GATv2 will continue this trend as it is an attractive and simple improvement for the currently dominating GAT. Although the authors of GATv2 note that it is not yet entirely clear which tasks would benefit the most from the usage of GATv2 over GAT, which will require more research and models to use GATv2.

In ATS the graphs used are in nearly all cases not homogeneous. However, the equations presented here do not work for heterogeneous graphs. The solution for this problem involves defining one convolution layer for each node type combination occurring within the graph. In the case of the already discussed sentence and word node graph there would be four possible combinations of types, and four convolution layers which would have to be defined if the graph were fully connected.

Convolution is a central aspect of GNNs, but pooling also presents an important and common operation, especially whenever GNNs are used jointly with other models. Pooling in GNNs is achieved by generating a global representation of the graph, or a subset of the graph, by pooling together features of nodes. This is usually done with some function f where f is commonly the *mean*, *max* or *sum*.

We want to explicitly point out to the reader that the construction of GNNs does not require special datasets. All GNN models for ATS use the common benchmark single-document and multi-document summarization datasets such as DUC 2004 or CNN/Dailymail. The only requirement for any ATS, or textual dataset, is for the designer to find an appropriate way of encoding sentences, words, subwords etc. into feature vectors, and finding a sensible way of connecting them.

3 Graph Neural Network Models For Automatic Text Summarization

The current state of research clearly shows that the usage of GNNs for ATS follows one of two patterns,

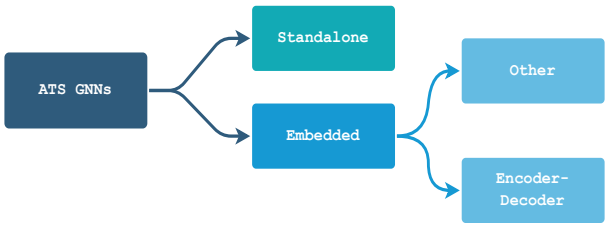

Figure 2: Taxonomy of GNN models used for ATS.

either the GNN is used directly for the ATS task or it is used to support a larger system. These trends justify the taxonomy as seen in Figure 2. More precisely, we classify standalone ATS GNNs as GNN models which are directly responsible for generating a summary. Embedded GNNs on the other hand are used to support a larger system, and are not directly or solely responsible for producing the summary. In addition to this difference, we further differentiate between embedded GNNs used in an encoder-decoder setting and other embedded GNNs.

Note that we do not specifically differentiate between single and multi-document GNN summarization models as some models are by design capable of handling both tasks. Additionally, we do not make a distinction between abstractive and extractive approaches, since embedded GNNs can be part of either approach, while abstractive approaches have not yet been developed for standalone GNNs, to the best of our knowledge.

3.1 Standalone GNNs

We will start our discussion of standalone GNN models with **HeterSumGraph** (HSG), a model proposed by Wang et al. (2020). We will do so due to the fact that this model illustrates concepts and ideas seen throughout GNNs models used for ATS. An illustration of the general concepts presented here can be seen in Figure 3.

The **HSG** model encodes each text into a graph with three node types, sentence nodes, word nodes and document nodes. The connection between these nodes is decided by inclusion i.e. if the word represented by a word node occurs in a sentence then their respective nodes are connected by an edge. The same principle applies to document nodes which are connected depending on whether a word, represented by a word node, occurs within the document. This is a flexible structure, as it can be used in a single-document but also multi-document setting.

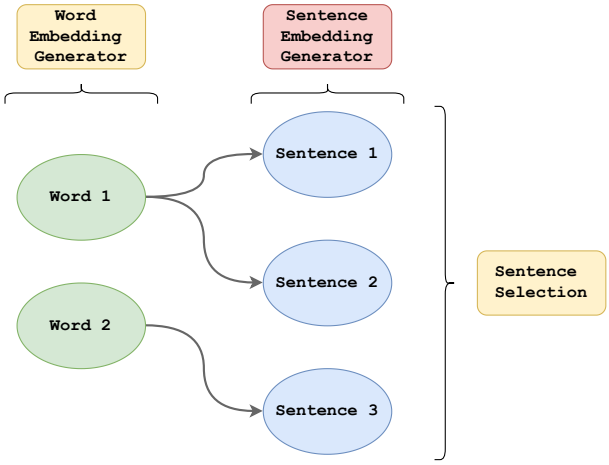

Figure 3: General architecture of standalone GNNs with word nodes and sentence nodes, encoders for both node types and a sentence selection mechanism. Inspired by **HSG**.

The feature vectors for all nodes are obtained by encoders and the edge weights are obtained by computing the TF-IDF score for each word. The neural network consists of a modified GAT layer. The GAT is modified to consider the TF-IDF value of the connecting edge. Additionally a position-wise feed-forward (FFN) layer consisting of two linear transformations is applied to the hidden state after the convolution. In total three convolution layers are used, word-sentence, sentence-word and word-document. The model is then trained on a node-based binary classification task that is predicting whether a sentence node is to be included for the summary or not.

The classification itself is done by a single linear layer. The model then does not directly use the predicted nodes to produce the summary. Instead trigram blocking (Paulus et al., 2018) is used during sentence selection in order to ensure sparsity of the generated summary.

The results for this model are quite impressive as it outperforms non-BERT based models on both single-document and multi-document summarization for the CNN/DailyMail dataset. One should especially note the flexibility and ability to use this model for two tasks.

An older model by Muratore et al. (2010) can be considered a precursor to this architecture. A simple extension to the **HSG** model is proposed by Ya et al. (2021). In their extension they modify the model for query constraints for the summary. This is achieved by adding a query node to the graph structure. Additionally, they introduce a mu-

tual information maximization mechanism during training.

A model which further follows this structure is the one by Linmei et al. (2019). The authors there extend the attention mechanism by adding another layer of attention, allowing it to include information about the type of the node during convolution. The GNN model by Jing et al. (2021) encodes even more information into the graph by considering the relation between sentences on a number of different levels. In particular, they encode the semantic and syntactical relationship between sentences within the graph.

This idea of encoding additional information into the graph is also followed by Antognini and Faltings (2019). They introduce an additional *universal* feature vector which is added to each sentence node embedding. This *universal* feature vector is learned from a large unrelated and general corpus. This model is also unique in that it focuses on the summarization of very small texts, on average less than 100 words.

Taking this basic structure and idea even further is the model called **HAHSum** by Jia et al. (2020a). The construction of the input graph for **HAHSum** is more involved as it aims to significantly reduce semantic sparsity by utilizing named entities. The model uses three types of nodes, named entity nodes, word nodes and sentence nodes, with the named entity nodes being anonymized tokens. The graph is then built as follows, word nodes are connected with a directed edge to a sentence node if they occur within the sentence. Two named entities are connected with an undirected edge if they represent the *same* entity and two sentence nodes are connected with an undirected edge if they share a trigram. Additionally, sequentially occurring words and entities are connected with a directed edge. This setup shows how one can encode a substantial amount of implicit information in an explicit manner.

HAHSum uses a GAT for each of the five node type combinations found within the graph. Just as in **HSG**, a FFN is applied after the multi-head attention and again as in the previous model a linear layer is used to perform the binary classification of the sentence nodes.

The results for **HAHSum** show that GNNs can perform very well. The authors of the paper tested the model on the CNN/Daily Mail, Newsroom and NYT dataset. The model outperforms very powerful models such as **MATCHSUM** (Zhong et al., 2020) and is even able to compete in some metrics with leading abstractive models such as **PEGASUS** (Zhang et al., 2020a). The results of an *Amazon Mechanical Turk* experiment corroborate these results and show that for human readers **HAHSum** produce summaries with superior fluency and conciseness.

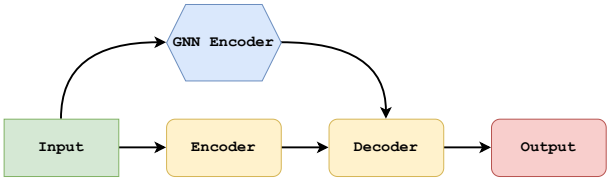

Figure 4: Typical setup for embedded GNNs in the encoder-decoder category. The GNN is used to encode part of the input and then the resulting encodings are forwarded to the decoder.

Another recent GNN model which has achieved great performance in the task of multi-document summarization is the **SgSum** model by Chen et al. (2021). Different to the approaches outlined above, the **SgSum** model uses graph pooling to extract sub-graphs from encoded documents. That is, it first transforms the documents into a large graph, then generates a number of sub-graphs via pooling and convolution. These sub-graphs are then ranked and thereby selected for a summary. This is quite an innovative approach as it casts the problem of multi-document summarization as a simple sub-graph selection problem. Additionally, it outputs an integral summary, that is the entire summary is output by the model in the form of the sub-graph of sentences.

3.2 Embedded GNNs

The first embedded GNN model we will present is the **GRU-GCN** model by Yasunaga et al. (2017). Despite being an older model it provides an illustrative introduction as to how one can effectively incorporate GNNs into established deep learning methods. As this model utilizes GNNs within a sequence-sequence architecture it falls into the encoder-decoder category of embedded GNNs. The model works exclusively with multi-document summarization. We want to note that due to the age of this model the GNN uses spectral-based convolution instead of spatial-based convolution.

As the model is an encoding-decoding embedded GNN it features three parts: the encoding part, the GNN and a decoding part. The encoding and decod-

GNN CNN/DailyMail Performance Overview					
Model Name	Type	Description	R-1	R-2	R-L
BERTSUMEXT (Liu and Lapata, 2019)	Extractive BERT baseline	BERT-based	43.85	20.34	39.90
Topic-Graphsum (Cui et al., 2020)	Extractive (Embedded)	NTM + GNN	44.02	20.81	40.55
DSGSUM (Bi et al., 2021)	Abstractive (Embedded)	Seq-Seq with GNN	41.96	19.29	38.98
Syntactic-Graph (Xu et al., 2020a)	Abstractive (Embedded)	Syntactic graph	41.79	19.06	38.56
HAHSUM (Jia et al., 2020a)	Extractive (Standalone)	Entity-relations + words and sentences	**44.68**	**21.30**	**40.75**
Multi-Gras (Jing et al., 2021)	Extractive (Standalone)	Sentence information encoding	43.16	20.14	39.49
HSG (Wang et al., 2020)	Extractive (Standalone)	Multi-layer encoding of documents	42.95	19.76	39.23
DISCOBERT (Xu et al., 2020b)	Extractive (Embedded)	BERT+GNN	43.77	20.85	40.67
GNN DUC-2004 Performance Overview					
GRU-GCN (Yasunaga et al., 2017)	Extractive (Embedded)	GRU+GNN	38.23	9.48	NA
SGSum (Chen et al., 2021)	Extractive (Standalone)	Subgraph extraction	**39.41**	**10.42**	**35.41**

Table 1: Combined results as reported in their respective papers. We only report the best performing model type and only ROUGE-1 (R-1), ROUGE-2 (R-2) and ROUGE-L (R-L) scores. We show results on the CNN/Dailymail dataset for single-document summarization and results on the DUC-2004 dataset for multi-document summarization. *NA* denotes that the authors have not reported this score. Best results marked with boldface.

ing are done by gated recurrent networks (GRUs). More specifically, sentence encodings are produced by the encoding GRU network. These sentence encodings are then used by the GNN whose input graph consists solely of sentence nodes. The edges are determined by semantic relatedness of the sentences. The resulting sentence node feature vectors produced by the GNN are passed to the decoding GRU which computes salience scores for each sentence. The results for this model have been surpassed by other models.

Similar to the **GRU-GCN** model, the **Topic-GraphSum** model by Cui et al. (2020) combines an established deep neural network method with a GNN. In it a variational autoencoder is utilized for modeling topics within a given text, that is, it learns latent topics via encoding-decoding. The GNN is fed a graph consisting of topic nodes and sentence nodes. The topic node embeddings are produced by the autoencoder and the sentence node embeddings are produced by **BERT**. The GNN utilizes a GAT for the prediction of sentences to be used for the summary. Note that the autoencoder and the GNN are trained jointly, which is why this model is classified as an embedded model. However, as it does not utilize the encoder-decoder architecture it falls into the category of other embedded GNNs.

Another embedded GNN which does not follow the encoder-decoder schema is the model by Xu et al. (2020b). Their **DISCOBERT** model incorporates a GNN into a **BERTSUM** (Liu and Lapata, 2019) like architecture.

The **DSGSum** model developed by Bi et al. (2021) utilizes a GNN to enhance the semantic information provided to the model. **DSGSum** uses the GNN mainly for encoding entity information. The input to the GNN consists of an entity graph which has been enriched with a knowledge graph (KG); specifically, the entities and their relations as defined in the KG are encoded into the graph. The GNN then utilizes GAT to produce entity embeddings which are directly used by the decoder part of the architecture. Hence **DSGSum** is an embedded encoder-decoder model. All of the following embedded GNN ATS models follow the ATS GNN encoder-decoder pattern. The general principle of these models is illustrated by Figure 4. This principle being the usage of the GNN to supplement the information provided to the decoder, while also utilizing an encoder for generating the input to the GNN.

The authors of (Wu et al., 2021b) produce a graph based on the semantics of each sentence. The encoding produced by the GNN and a textual encoding of each sentence are then passed to a decoder. Similarly, the model by Xu et al. (2020a) uses a GNN as an encoding component. Their model utilizes the dependency tree of the input as a graph input to the GNN. It also uses a modified attention mechanism which is used to decode the attention of the GAT directly into the decoder part of the architecture. Another model in this category is the model by Liang et al. (2021). Following this idea even further is the model by Li et al. (2020). However, in contrast to **DSGSum** or others, their model uses a GNN directly within the transformer based encoder and decoder blocks, and not as an outside component providing additional encodings.

Another area where GNNs have found some usage is in the abstractive summarization of di-

alogues, which although a niche area, is still part of the ATS task. The model by Zhao et al. (2020) uses a GNN to encode the structure of the conversation into their sequence-to-sequence architecture. This is also done by Feng et al. (2020). Also for dialogue summarization Feng et al. (2021) introduce a special type of graph encoding to a sequence-to-sequence architecture. They utilize a GNN as an encoder and their input graph links information about the speaker, the spoken sentences and other information together.

At this point we also want to shortly mention a few models which do not perform classical ATS but do perform a specialized form of summarization with the help of a GNN. The models by Liu et al. (2020a) and LeClair et al. (2020) both perform code summarization, that is they generate a natural language description/summarization of code written in a programming language. They both utilize a GNN, and also both leverage the abstract syntax tree of the program given. Another interesting model is the model by Wu et al. (2019). Their model performs multi-video summarization with the help of a GNN.

4 Conclusions and Outlook for Future

We have surveyed in this paper the main developments in the area of GNNs applied to the automatic text summarization task, first describing how GNNs work and then discussing the prominent models used for ATS. We have also provided simple categories of the models. Finally, in Table 1 we have combined results from the models mentioned in our survey. Note that this is just a simple enumeration of the best *ROUGE* scores reported, which does not fully capture other important aspects of summaries such as fluency, conciseness, relevancy and in the case of abstractive summarization, factual accuracy.

We have already highlighted some general advantages of GNNs for ATS. Now considering the models presented and the results shown in Table 1 we want to give some pointers towards the future of GNNs for ATS.

- **Outside the Box.** Models such as **SGSum** and **HAHSUM** show how one can achieve great performance by rethinking parts of the ATS task. GNNs provide with their direct acceptance of graphs a lot of freedom with regards to the design of the input as well as the output. The idea of reconsidering multi-document ATS as a subgraph ranking task is a worthwhile approach to consider further.

- **Encoding and Enhancing.** GNNs are able to efficiently and effectively produce encodings of graph structures, which are prevalent throughout texts. Models like **DISCOBERT** show how one can enhance the performance of *traditional* models by incorporating GNNs and leveraging their encoding ability. We believe many models can be improved by incorporating GNNs in ways presented in the survey.

- **Text Length.** GNNs are capable of scaling to very large graph sizes. Summarizing large texts is an important task, but one which has been neglected in the literature. GNNs are in our estimation in a position to perform very well on large text summarization.

- **Explainability.** No GNN ATS model so far has leveraged the explainability aspect of GNNs. We believe that this is a very unique and unexplored research avenue that could reveal valuable insights into ATS and GNNs used for ATS.

References

Sergi Abadal, Akshay Jain, Robert Guirado, Jorge López-Alonso, and Eduard Alarcón. 2021. Computing graph neural networks: A survey from algorithms to accelerators. *ACM Computing Surveys (CSUR)*, 54(9):1–38.

Diego Antognini and Boi Faltings. 2019. Learning to create sentence semantic relation graphs for multi-document summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 32–41.

Qiwei Bi, Haoyuan Li, Kun Lu, and Hanfang Yang. 2021. Augmented abstractive summarization with document-level semantic graph. *Journal of Data Science*, 19(3).

Shaked Brody, Uri Alon, and Eran Yahav. 2021. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*.

Moye Chen, Wei Li, Jiachen Liu, Xinyan Xiao, Hua Wu, and Haifeng Wang. 2021. Sgsum: Transforming multi-document summarization into sub-graph selection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4063–4074.

Peng Cui, Le Hu, and Yuanchao Liu. 2020. Enhancing extractive text summarization with topic-aware graph neural networks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5360–5371.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. 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)*, pages 4171–4186.

Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Cogltx: Applying bert to long texts. *Advances in Neural Information Processing Systems*, 33:12792–12804.

Wafaa S. El-Kassas, Cherif R. Salama, Ahmed A. Rafea, and Hoda K. Mohamed. 2021. Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, 165:113679.

Xiachong Feng, Xiaocheng Feng, and Bing Qin. 2021. Incorporating commonsense knowledge into abstractive dialogue summarization via heterogeneous graph networks. In *China National Conference on Chinese Computational Linguistics*, pages 127–142. Springer.

Xiachong Feng, Xiaocheng Feng, Bing Qin, and Xinwei Geng. 2020. Dialogue discourse-aware graph model and data augmentation for meeting summarization. *Dialogue*, 1:U2.

Daniele Grattarola and Cesare Alippi. 2021. Graph neural networks in tensorflow and keras with spektral [application notes]. *IEEE Computational Intelligence Magazine*, 16(1):99–106.

Ruipeng Jia, Yanan Cao, Hengzhu Tang, Fang Fang, Cong Cao, and Shi Wang. 2020a. Neural extractive summarization with hierarchical attentive heterogeneous graph network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3622–3631.

Zhihao Jia, Sina Lin, Mingyu Gao, Matei Zaharia, and Alex Aiken. 2020b. Improving the accuracy, scalability, and performance of graph neural networks with roc. *Proceedings of Machine Learning and Systems*, 2:187–198.

Baoyu Jing, Zeyu You, Tao Yang, Wei Fan, and Hanghang Tong. 2021. Multiplex graph neural network for extractive text summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 133–139.

Wojciech Kryściński, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551.

Wojciech Kryściński, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346.

Alexander LeClair, Sakib Haque, Lingfei Wu, and Collin McMillan. 2020. Improved code summarization via a graph neural network. In *Proceedings of the 28th International Conference on Program Comprehension*, pages 184–195.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. 2021. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pages 6437–6449. PMLR.

Wei Li, Xinyan Xiao, Jiachen Liu, Hua Wu, Haifeng Wang, and Junping Du. 2020. Leveraging graph to improve abstractive multi-document summarization. *arXiv preprint arXiv:2005.10043*.

Zeyu Liang, Junping Du, Yingxia Shao, and Houye Ji. 2021. Gated graph neural attention networks for abstractive summarization. *Neurocomputing*, 431:128–136.

Hu Linmei, Tianchi Yang, Chuan Shi, Houye Ji, and Xiaoli Li. 2019. Heterogeneous graph attention networks for semi-supervised short text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4821–4830.

Shangqing Liu, Yu Chen, Xiaofei Xie, Jing Kai Siow, and Yang Liu. 2020a. Retrieval-augmented generation for code summarization via hybrid gnn. In *International Conference on Learning Representations*.

Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. 2020b. Tensor graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8409–8416.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740.

Rui Luo, Shan Zhao, and Zhiping Cai. 2020. Application of graph neural network in automatic text summarization. In *National Conference of Theoretical Computer Science*, pages 123–138. Springer.

Masoud Malekzadeh, Parisa Hajibabaee, Maryam Heidari, Samira Zad, Ozlem Uzuner, and James H Jones. 2021. Review of graph neural network in text classification. In *2021 IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0084–0091. IEEE.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Donatella Muratore, Markus Hagenbuchner, Franco Scarselli, and Ah Chung Tsoi. 2010. Sentence extraction by graph neural networks. In *International Conference on Artificial Neural Networks*, pages 237–246. Springer.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. In *ACL*.

Jiaxin Wu, Sheng-Hua Zhong, and Yan Liu. 2019. Mvsgcn: A novel graph convolutional network for multi-video summarization. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 827–835.

Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, and Bo Long. 2021a. Graph neural networks for natural language processing: A survey. *arXiv preprint arXiv:2106.06090*.

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2020a. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)*.

Wenhao Wu, Wei Li, Xinyan Xiao, Jiachen Liu, Ziqiang Cao, Sujian Li, Hua Wu, and Haifeng Wang. 2021b. Bass: Boosting abstractive summarization with unified semantic graph. *arXiv preprint arXiv:2105.12041*.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020b. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

Haiyang Xu, Yun Wang, Kun Han, Baochang Ma, Junwen Chen, and Xiangang Li. 2020a. Selective attention encoders by syntactic graph convolutional networks for document summarization. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8219–8223. IEEE.

Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020b. Discourse-aware neural extractive text summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5021–5031.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? In *International Conference on Learning Representations*.

Mingzhou Xu, Liangyou Li, Derek F Wong, Qun Liu, and Lidia S Chao. 2021. Document graph for neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8435–8448.

Jing Ya, Tingwen Liu, Jiangxia Cao, and Li Guo. 2021. Heterogeneous graph neural networks for query-focused summarization. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pages 720–728. SIAM.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462.

Yongjing Yin, Fandong Meng, Jinsong Su, Chulun Zhou, Zhengyuan Yang, Jie Zhou, and Jiebo Luo. 2020. A novel graph-based multi-modal fusion encoder for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3035.

Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32.

Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2020. Explainability in graph neural networks: A taxonomic survey. *arXiv preprint arXiv:2012.15445*.

Hang Zhang, Yeyun Gong, Yelong Shen, Weisheng Li, Jiancheng Lv, Nan Duan, and Weizhu Chen. 2021. Poolingformer: Long document modeling with pooling attention. In *International Conference on Machine Learning*, pages 12437–12446. PMLR.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020a. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Yufeng Zhang, Xueli Yu, Zeyu Cui, Shu Wu, Zhongzhen Wen, and Liang Wang. 2020b. Every document owns its structure: Inductive text classification via graph neural networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 334–339.

Lulu Zhao, Weiran Xu, and Jun Guo. 2020. Improving abstractive dialogue summarization with graph structures and topic words. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 437–449.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuan-Jing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.