applied sciences

MDPI

Article

Sentence Graph Attention For Content-Aware Summarization

Giovanni Siragusa [1,*] and Livio Robaldo [2,*]

1 Dipartimento di Informatica, Universitá degli Studi di Torino, Corso Svizzera 185, 10149 Torino, Italy
2 Legal Innovation Lab Wales, Swansea University, Singleton Park, Sketty, Swansea SA28PP, UK
* Correspondence: siragusa@di.unito.it (G.S.); livio.robaldo@swansea.ac.uk (L.R.)

Abstract: Neural network-based encoder–decoder (ED) models are widely used for abstractive text summarization. While the encoder first reads the source document and embeds salient information, the decoder starts from such encoding to generate the summary word-by-word. However, the drawback of the ED model is that it treats words and sentences equally, without discerning the most relevant ones from the others. Many researchers have investigated this problem and provided different solutions. In this paper, we define a sentence-level attention mechanism based on the well-known PageRank algorithm to find the relevant sentences, then propagate the resulting scores into a second word-level attention layer. We tested the proposed model on the well-known CNN/Dailymail dataset, and found that it was able to generate summaries with a much higher abstractive power than state-of-the-art models, in spite of an unavoidable (but slight) decrease in terms of the Rouge scores.

Keywords: summarization; knowledge graph; neural networks; pagerank; natural language processing

check for updates

Citation: Siragusa, G.; Robaldo, L. Sentence Graph Attention For Content-Aware Summarization. *Appl. Sci.* **2022**, *12*, 10382. https://doi.org/10.3390/app122010382

Academic Editor: Rocco Furferi

Received: 5 September 2022
Accepted: 10 October 2022
Published: 14 October 2022

1. Introduction

Textual Summarization (TS) is the task of compressing a text into a short form (the summary) that preserves all the relevant information, where the compression rate is chosen by a domain expert. There exist two approaches to TS: *extractive* TS where sentences are extracted from the input text and re-arranged to create the summary, and *abstractive* TS where the summary is generated word-by-word. With the ever-growing adoption of neural networks (NNs) by Natural Language Processing (NLP) techniques and tasks, researchers have found that NN-based encoder–decoder (ED) models [1,2] are well-suited to generating abstractive summaries. In such models, the encoder is trained to select the salient information from the text, whereas the decoder is trained to condense such information into a fixed-length summary. To the best of our knowledge, Refs. [3–5] represent the first research works that have successfully adopted NNs for abstractive TS.

The main drawback of using NN-based models for TS is their inability to capture the salience of each sentence. They are unable to discern the relevant and informative sentences (or words) in a summary from those that are secondary for the context, treating them equally. For instance, suppose that a document talks about both a big snow storm and a man who started to sell snow online; suppose that the correct summary is related to the online snow selling. The network may decide to focus on the snow storm, instead of the online snow selling, creating a summary of this topic and completely missing the content of the article. We think that this problem is generated by the attention mechanism [6,7] of the neural network, i.e., a score distribution over the document words that the network uses to focus on the relevant ones. Since such a mechanism has been defined for machine translation where the focus is limited to about three words at time, in our opinion, it is not able to capture the salient words present in longer texts (such as those of summarization).

To solve this drawback, we examined the recent works on neural network models based on knowledge bases (KBs) [8–14]. In these research works, the knowledge graph created using an external KB was explored by the neural network to extract the relevant information (i.e., it selected one or more vertexes) via the attention mechanism, which focused on the relevant entities of the graph. These models have shown very high performance, taming the problems related to large graphs (e.g., millions of vertexes) and missing the connections between vertexes, for which the neural networks are able to exploit latent information.

However, adopting an external KB in the summarization task was a challenge. We could construct it using the entities that were present in the document to summarize, but the neural network could have the above-described problem, selecting external entities that were neither relevant nor correct for the summary; thus, we decided to create the graph using the sentences present in the input document. The graph selected the sentences that were fundamental to constructing the summary and from which the salient words (i.e., entities) could be copied into the output.

Our contribution is twofold:

- a latent knowledge graph based on document sentences, which was changed according to the generated summary. In detail, the model calculated the sentence-level attention scores using the graph, which were used by the word-level attention to select the relevant entities from the sentences;
- an inference-time score function that removed repetitions and attention errors, integrating two existing techniques: coverage vectors [6,7,15] and reinforcement learning.

We tested our proposed model on the CNN/Dailymail dataset, obtaining interesting results both in the selection of the relevant sentences and the creation of summaries that contained all the salient information of the input document.

The remainder of the article is composed of the following: Section 2 describes the latest research works in neural-based knowledge bases and summarization; Section 3 describes the proposed model, the sentence-level attention, and the inference-time score function; Section 4 reports the results obtained using the proposed model. The article concludes with Section 5.

2. Related Works

Our model belongs to the recent research field that exploits document contents to generate a better summary. In this context, to the best of our knowledge, there are two research directions: (1) retrieve, rank, and rewrite summarization models [16–18] and (2) content selection methods [19–24]. In the former, models are trained to select sentences from the input document or from an external resource (retrieve and rank part) that are salient, i.e., that express the information in a very concise manner. The selected sentences are then rewritten (substituting, adding, and deleting words) to generate the summary. In the latter, researchers have defined layers, masks, and network structures to uncover salient sentences and words while removing redundant information. In this context, our proposed model fuses the sentence graph proposed in Tan et al. [20] with the attention re-score method of Hsu et al. [22]; our idea was to find the relevant sentences via the graph, which takes into account the shared information between sentences, and to select the salient words from them using Hsu et al.'s method. Furthermore, instead of applying the softmax function in the PageRank method as Tan et al., we decided to use the sigmoid function, which could select more than one sentence.

Other interesting works integrated further information into sequence-to-sequence models. Refs. [25–27] used a pre-trained language model to improve the summary generation. Their idea was that the language model carries both fluency and domain style since it is able to deeply understand the meaning of each token. In detail, Liu and Lapata [26] and Song et al. [27] used BERT-based models [28] to solve the summarization task. Kryściński et al. [25], instead, proposed to extend the sequence-to-sequence model proposed by Paulus et al. [6] with a language model that integrates external knowledge. Other authors, such as Narayan et al. [29], preferred to use the topic model to tie the model output to the document.

Currently, researchers are studying how to integrate neural networks with knowledge bases. Some research works focused on table-to-text [8–10,30], where a table that depicts a "plain" knowledge base is read by a sequence-to-sequence model and transformed into a text that reports (almost) all information occurring in the table. In Wang et al. [30], the authors constructed both a latent graph over the entries of the table, looking to their position in the input, and an attention over them. This attention was refined in their successive work [8], where they proposed a hybrid attention based on both the <*slot type, slot value*> attention and the *link* attention of the latent graph; the latter represented a sort of embedding and was learned via backpropagation rather than being computed by the model. Hayashi et al. [9] and Liu et al. [10] defined a language model based on a knowledge graph. In the former work, their model first selected from the knowledge base the entity that had to be copied in the output; then, a mechanism in the language model forced it to generate the words of the entity. In the case where a word of the entity was not present in the vocabulary, they used a character-based language model to generate it. In the latter work, however, their model first selected all the entities that could continue a phrase excerpt and encoded them using an LSTM [31]; the resulting vector was then passed by the input to the language model, which copies the words of a chosen entity. Hu et al. [11] used a neural network with a copying mechanism [6,7] over a knowledge graph to generate the text. Their model first encoded the RDF triplets using a stacked Gated Convolutional Neural Network (GCN) in order to obtain a more complex latent graph; then, the vertexes obtained by the GCN were fed by the input to the language model.

Other works, instead, used a knowledge graph in the Question Answering (QA) task [12–14,32]. Given a question, they retrieve from a knowledge base all those entities that are associated with the question; then, both the question and the retrieved entities are passed to a neural network, which selects the answer (i.e., the best retrieved entity). These research works reason on triplets (h, r, t), where h and t are the entities and r is the relation that connects them. In [13,32], Saxena et al. calculated a score for each triplet multiplying together their elements (i.e., h, r, and t); the triplet with the highest score was then selected as the answer to the input question. Huang et al. [12] defined a knowledge embedding based on QA, where their idea was to represent the relation and the entity as low-dimensional vectors. To accomplish their task, they trained their model on a simple QA dataset, where the question was easily answered if the correct entity and predicate (the relation) were identified. Finally, Bosselut and Leskovec [14] proposed an interesting research work: they constructed a knowledge graph using the entities present in the question and the possible answers; then, they defined a model that used attention to focus on the relevant entities of the graph in order to create the context vector. The latter, the relevance of each vertex of the graph, and their relation type were used to predict the answer.

3. Our Approach: A Model with Graph Sentence-Level Attention

Our proposed model is similar to Nallapati et al. [4]'s model. It consists of a hierarchical bidirectional LSTM encoder and an attention-based LSTM decoder. In detail, unlike the classic encoder, the hierarchical one is composed of two encoders, each one formed by a bidirectional LSTM. In this paper, we call the first encoder the *word encoder* and the second one the *sentence encoder*.

The model works as follows: first, the word encoder generates sentence representations by reading all the M words in a sentence; then, the sentence representations are read by the sentence encoder to generate the document representation.

Let $\mathbf{x_1} = [x_1^1, x_1^2, \ldots, x_1^M]$ be the tokens that compose the first sentence of the document and $\mathbf{D} = [\mathbf{x}_1, \ldots, \mathbf{x}_K]$ the set of K sentences that compose the document. At each step i, the word encoder is fed with the input tokens x_1^i and produces two encoded states $\overrightarrow{\mathbf{h}}_1^i$ and $\overleftarrow{\mathbf{h}}_1^i$, where the former is generated by reading the sentence's words from left to right and the latter by reading the sentence's words from right to left. Those two states are distinguished through the jargon terms *forward state* and *backward state*. Equation (1) represents the forward and backward states' construction:

$$\begin{aligned}\overrightarrow{\mathbf{h}}_i^j &= \overrightarrow{LSTM}_W(E(x_i^j), \overrightarrow{\mathbf{h}}_i^{j-1}) \\ \overleftarrow{\mathbf{h}}_i^j &= \overrightarrow{LSTM}_W(E(x_i^j), \overleftarrow{\mathbf{h}}_i^{j-1})\end{aligned} \tag{1}$$

In (1), $\overrightarrow{LSTM}_W$ represents the word-level forward LSTM and $\overleftarrow{LSTM}_W$ the backward one. In addition, $E(\cdot)$ is a function that returns the word-embed [33,34] of an input word.

Once the encoded states of a sentence are generated, the last forward state $\overrightarrow{\mathbf{h}}_i^M$ and the last backward state $\overleftarrow{\mathbf{h}}_i^1$ are concatenated together to generate the sentence representation $\mathbf{h}_i$. Such an operation is represented in Equation (2), in which $||$ is the concatenation operator.

$$\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i^M || \overleftarrow{\mathbf{h}}_i^1] \tag{2}$$

Then, the sentence representations $[\mathbf{h}_1, \dots, \mathbf{h}_K]$ are passed to the sentence encoder, which produces a forward state $\overrightarrow{\mathbf{d}}_i$ and backward state $\overleftarrow{\mathbf{d}}_i$. The document representation $\mathbf{d}$ is constructed concatenating the last forward state with the last backward one:

$$\mathbf{d} = [\overrightarrow{\mathbf{d}}_K || \overleftarrow{\mathbf{d}}_1] \tag{3}$$

Similar to the encoder, at each step t, the decoder (a single unidirectional LSTM) receives as input the embedding of the previous word from the decoder state $\mathbf{s}_t$ and the context vector $\mathbf{c}_t$, and uses them to emit a word in the output (the network emits a probability distribution over the vocabulary that is used to select the next word). During training, it is the embedding of the target word w_{t-1}, whereas in testing it is the embedding of the previous word emitted by the decoder. In the model, the decoder state is initialized with the document representation. The context vector $\mathbf{c}_t$ is calculated as in Bahdanau et al. [35]:

$$\mathbf{c}_t = \sum_{k=1}^{M} \hat{a}_k^t \, \mathbf{h}_k \tag{4}$$

where $\hat{a}_k$ represents an attention score calculated via the attention approach and $\mathbf{h}_k$ is the representation of the t-th word.

Inspired by the works of [4,20,22], we decided to combine the word-level attention with the sentence-level scores to select only those words coming from sentences that were relevant for the summary. Given $\mathbf{b}$ was the sentence-level scores and $\mathbf{a}$ was the word-level ones, we computed the attention scores $\hat{\mathbf{a}}$ as follows:

$$\hat{a}_i = \frac{a_i * b_{s(i)}}{\sum_{k=1}^{M} a_k * b_{s(k)}} \tag{5}$$

In (5), the $\mathbf{a}$ scores were computed by Equation (6) (the symbol $\intercal$ in $\mathbf{v}^\intercal$ is the transpose operator). On the other hand, the $\mathbf{b}$ scores were calculated using a graph-based neural network, which took into account both the current decoder state $\mathbf{s}_t$ and the sentence representations. The graph-based network is described in Section 3.1. The full model is depicted in Figure 1.

$$\begin{aligned}e_j &= \mathbf{v}^\intercal \tanh(\mathbf{W}_e \, [\mathbf{s}_t || \mathbf{h}_j] + b_e) \\ a_i^t &= \frac{\exp(e_j^t)}{\sum_{k=1}^{M} \exp(e_j^t)}\end{aligned} \tag{6}$$

For the output, we adopted the mixture model proposed by See et al. [7], which combines the probability of a vocabulary word with its attention scores. Thus, the probability of emitting a word y_i at step t is defined as follows:

$$P(y_i^t) = \beta\, softmax(\mathbf{W}_o\,(\mathbf{W}_c\,[\mathbf{c}_t||\mathbf{s}_t] + b_c) + b_o) + (1-\beta)\sum_{w=y_t} \hat{a}_w^t \tag{7}$$

where $\mathbf{W}_o$, $\mathbf{W}_c$, b_o, and b_c are learnable parameters. β is a value within the range $[0, 1]$ used to mix the two distributions and it is calculated using Equation (8), where $\mathbf{W}_b$ and b_b are learnable parameters and y_{t-1} is the previously emitted word.

$$\beta = sigmoid(\mathbf{W}_b\,[E(y_{t-1})||\mathbf{c}_t||\mathbf{s}_t] + b_b) \tag{8}$$

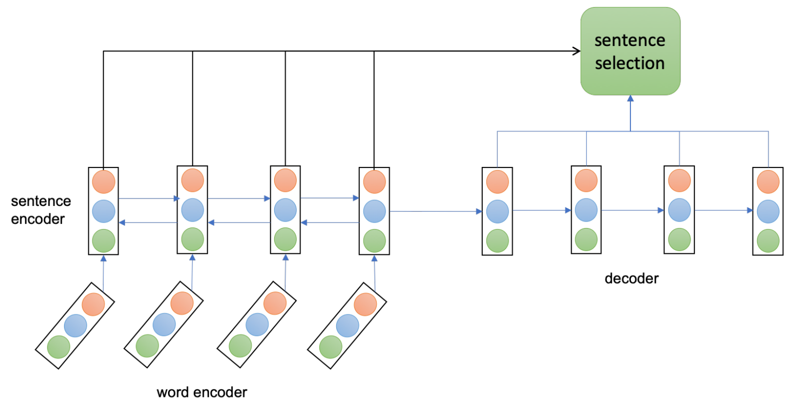


Figure 1. Architecture of the model. The *sentence-selection* method is described in Section 3.1.

In Equation (7), β was used to control how much information coming from the attention distribution will enter into the vocabulary distribution. If β is close to 1, the model generates the next word only using the vocabulary (i.e., selecting the word with the highest probability); if β is close to 0, then the model uses the attention distribution on the input document.

We trained the model to minimize the sum of the negative log likelihood of the sequence of target words w^*:

$$loss = \frac{1}{N}\sum_{t=1}^{N} -log\, P(w_t^*) \tag{9}$$

We found that the model tended to generate summaries that had repeated words (or sentences). Thus, in order to minimize the repeated words (sentences), we decided to apply some heuristics both in the training and inference steps. In the training step, we applied the coverage method defined by See et al. [7] (it is described in Section 3.2). At inference time, we adopted the scoring function of Gehrmann et al. [21] for the beam search, which uses a hyperparameter α to control the length of the summary. We also set a minimum summary length based on the training data. Additionally, following Paulus et al. [6], we restricted the beam search to never repeat the same trigram in the summary.

3.1. Sentence-Level Scores

Inspired by recent works on knowledge bases [9,10,13,14], our idea was to create a knowledge graph (KG) from which the neural network could select the relevant entities to insert in the summary. However, creating a KG for the summarization task was challenging: let us suppose we extracted all the entities present in the text and we created the KG from them, the model could select the entities that were not relevant for the summary, generating a text that was not related to the document's content.

As mentioned in the Introduction, we decided to create a latent graph looking at the informativeness of each sentence. The salient words could be copied into the summary via the pointer network [6,7]; the sentences were then seen as containers whose information must be unveiled.

The importance of a sentence to the summary was calculated by comparing it with *both* the current decoder state *and* the other sentences. The graph was, in particular, needed to compare the sentences to one another, specifically to understand whether some sentences were more informative than others.

For instance, let us suppose that we want to calculate the score for the sentence d_1. If we compare d_1 with the decoder state d, we obtain a score that highlights the relevance of d_1 with respect to the generated summary. Now, let us suppose that we want to calculate the score for the sentence d_2, which contains the same facts as d_1 as well as some additional ones. If we compare d_2 and d, we obtain its relevance with respect to the summary but not with respect to d_1. In other words, in this simpler setting, we cannot know whether a sentence is more informative than another one.

We thus constructed a graph where the vertexes were the sentences and the edge between two sentences was weighted by their cosine similarity. For simplicity, we treated the current decoder state, i.e., what the model summarized so far, as a sentence. Following the idea of Tan et al. [20], we then applied the PageRank algorithm to the graph to calculate the score of each vertex, assigning high scores to informative and relevant sentences while pushing the scores of redundant sentences down; the pointer network of the model (defined in Equation (7)) then copied the salient words coming from the high-score sentences. Figure 2 depicts an example of the graph score calculation:

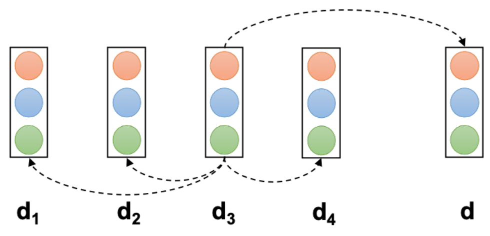
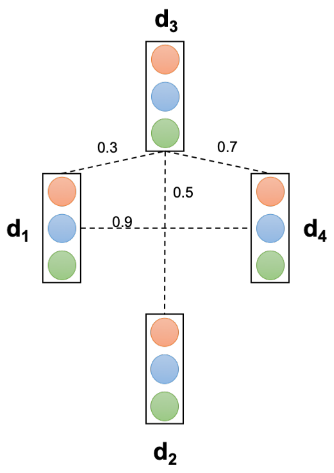
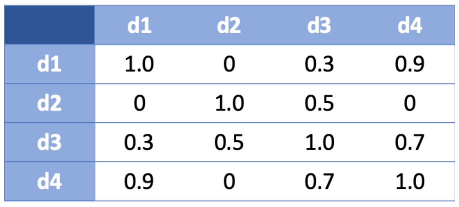


Figure 2. A graphical example of the score calculation of sentence $\mathbf{d}_3$. Dotted arrows represent the relationship between the current sentence and the other representations.

We defined the adjacency matrix A of the graph via a bi-linear product of the sentences over which we applied the cosine similarity function, similar to the work proposed by Yao et al. [36]. The cosine similarity ensured that the diagonal of A only contained values equal to 1 (because we were comparing a vector with itself), as required by graph-based models (they require that the identity matrix be added to the adjacency matrix in order to normalize the values). Figure 3 shows an example of a graph and its adjacency matrix.





	d1	d2	d3	d4
d1	1.0	0	0.3	0.9
d2	0	1.0	0.5	0
d3	0.3	0.5	1.0	0.7
d4	0.9	0	0.7	1.0

Figure 3. The graph shows an example of a handmade graph with its adjacency matrix obtained by applying the ReLU function over it. Negative weights, i.e., negative cosine similarity scores, were set to zero thanks to the ReLU function, for example, between d_1 and d_2 (and between d_2 and d_4).

In detail, A is defined as follows, in which d_i and d_j are two sentence representations.

$$A[i,j] = cosine(\mathbf{d}_i, \mathbf{d}_j) \tag{10}$$

We also applied the ReLU function to the matrix A in order to remove all edges between dissimilar sentences (i.e., those with a negative cosine score). The mathematical calculation of the sentence scores is defined by Equation (11):

$$\mathbf{b} = \sigma((1-\lambda)(I - \lambda\, ReLU(A)\, D^{-1})^{-1}\mathbf{y}) \tag{11}$$

where I is the identity matrix, A is the adjacent matrix of the graph, D is the diagonal matrix where the i-th diagonal element is equal to the sum of the i-th column of the matrix A, and y is a one-hot vector of size $K+1$. It has all values equal to zero, except for the one that represents the decoder state. The scalar λ is the dumping factor, whereas $\sigma(\cdot)$ is the sigmoid function that assigns a score in the range $[0,1]$ to each sentence. Unlike the softmax function that could assign the probability mass to a single sentence, the sigmoid function selects more than one sentence, allowing the model to use them either for the context vector or the pointer network.

3.2. Coverage Method

In order to avoid word and sentence repetitions in our model, we adopted the coverage vector of See et al. [7]. In other words, we modified Equation (6) to include it, thus obtaining:

$$e_j^t = \mathbf{v}^T \tanh(\mathbf{W}_e\,[\mathbf{s}_t||\mathbf{h}_j||\mathbf{cov}_t] + b_e) \tag{12}$$

Finally, we used the auxiliary loss proposed by See et al. [7] to penalize the overlapping between the attention distribution and the coverage vector:

$$covloss = \sum_{i=1}^{N}\sum_{j=1}^{M} min(a_j^i, cov_j^i) \tag{13}$$

During the experiments with the coverage loss, we noticed that the model was producing semantically incorrect summaries. In our opinion, this was because the coverage loss had an impact on the attention and an indirect impact on the sentence-level scores and representations, invalidating part of the training.

We therefore decided to use such a loss in the inference time in order to guide the model in selecting the words that minimized the overlap between the attention and the coverage vector. We combined *covloss* with the reinforcement learning q-function since the latter was capable of adapting to the environment, selecting the best action (in this case, the best word to generate) at each time step. We had to specify that the resulting function was not a q-learning one, i.e., that it did not calculate a score for each state (time step) and action (generated word). We then defined the score function, called *covscore*, as in Equation (14). *Covscore* was added to the score function of the beam search; at $t = 0$, $covscore_t = 0$.

$$covscore_t = covscore_{t-1} + \frac{1}{t}(covloss_t - covscore_{t-1}) \tag{14}$$

4. Evaluation

We trained and tested the proposed model on the *CNN/DailyMail* dataset [37], which contains online articles paired with multi-sentence summaries. We used the script supplied by Nallapati et al. [4] to obtain the same version of their dataset that consists of 287,226 training pairs, 13,368 validation pairs and 11,490 test pairs. Each article has 781 tokens on average, whereas each summary has 3.75 sentences and 56 tokens on average. We used the *non-anonymized* version of the dataset (please see Chen et al. [38] for problems regarding the anonymized version).

The model has 256 dimensional hidden layers and 128 dimensional embedding layers. Following See et al. [7], we used a small vocabulary that included 50 k tokens for both the source and target. We set the dumping factor λ to 0.9. All the weights were initialized with a normal distribution, with a mean of 0 and a standard deviation of 0.1. The model was trained using AdaGrad [39], with a learning rate of 0.15. We also used gradient clipping with a gradient norm of 2.0 and dropout [40] with a probability of 0.2 (keep probability of 0.8) to improve model generalization. We used the loss on the validation set for early stopping.

The training was performed on a single GPU RTX 2080Ti, with a batch size of 8. At testing time, we used a beam size of 5 and an α value of 1.4 for Gehrmann et al. [21]'s score function. We set the maximum number of encoder input sentences to 8. We truncated the length of each sentence to 50 tokens and the length of the summary to 100 tokens in order to speed up the convergence of the model. We trained the model for about 1,200,000 iterations. The training of the model took about 5 days of computation. Table 1 reports the hyperparameters and their setting.

Table 1. The hyperparameters (i.e., parameters set by the user) and their values.

Parameter	Value
Embedding size	128
Hidden size	256
λ	0.9
Gradient clipping	2.0
Dropout	0.2
Optimizer	Adagrad
Learning rate	0.15
Batch size	8
Beam size	5
Gehrmann et al. [21]'s α	1.4
# words per sentence	50
# sentences	8
Decoder output length	100

We compared the proposed model with the following state-of-the-art works:

- See et al. [7]'s pointer generation model, which used the pointer network and the coverage method as in our proposal;
- Nallapati et al. [4]'s models, as it employed the hierarchical encoder with the two-level attention mechanism and a pointer network;
- Hsu et al. [22]'s model, as it used the same equation to combine the two attention distributions;
- Tan et al. [20]'s abstractive model, for its hierarchical structure, which was also extended to the decoder.

4.1. Results

Table 2 shows the Rouge scores (Rouge-1, Rouge-2, and Rouge-L) [41] of the models. Our models "*our model + coverage*" and "*our model + coverage + covscore*" obtained higher R-1 and R-2 scores than Nallapati et al.'s models. We suspect that our models had a low R-L due to their high abstraction power since the generated summaries contained tokens that were not present in the reference one; the longest common sub-sequence was short.

We noticed that the auxiliary loss did not improve the Rouge scores. Since the auxiliary loss modified the attention scores, it also impacted on the sentence-level scores and shook the sentence representations, invalidating part of the training. This was supported by the fact that the covloss incremented the loss in the validation set from 3.5 to 3.8 rather than decreasing it.

In light of these results, we can conclude that the proposed graph-based sentence-level attention generally performed better than the sentence-level attention proposed by Nallapati et al. [4]. It allowed for the generation of more abstractive summaries, as reported in Section 4.3, at the cost of sacrificing recall. In more detail, since the Rouge score only evaluates the presence of tokens in the generated summary of the reference model, it indirectly penalizes models that use synonyms and periphrasis. Indeed, our best model had very high precision (41.76 for R-1, 16.27 for R-2, and 27.16 for R-L) but very low recall (33.22 for R-1, 13.03 for R-2, and 21.50 for R-L). This is also the reason why our models had lower Rouge scores than those of See et al. [7], whose models were more oriented toward copying words from the input document than generating novel ones.

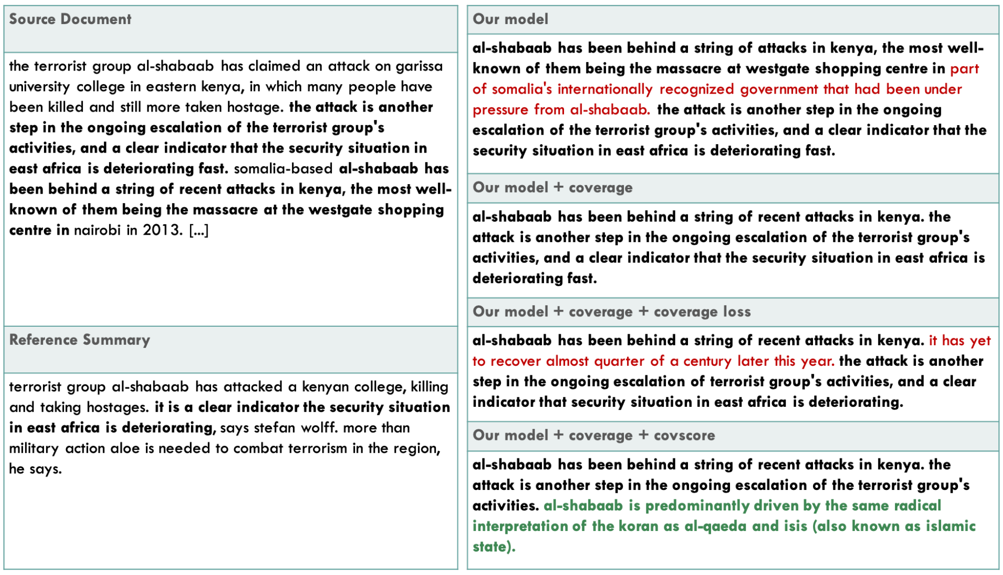
Finally, our models underperformed with respect to those in Hsu et al. [22] and Tan et al. [20]. This was expected since their models had a more complex architecture and used more complex features. The model of Hsu et al. [22] is composed of two trained models: an extractive model to select the sentences that are useful for the summary and an abstractive one to digest them; our models, instead, were jointly trained to compute a score for each sentence and generate a summary. The model of Tan et al. [20] first generates a sentence decoder state then generates the words of a sentence using the word-level attention and the pointer network. Compared to our models, which only combine both attentions and require fewer resources to be trained, theirs is resource-intensive because the hierarchical decoder adds a further million parameters, having a positive impact on the results.

Figure 4 shows an example of the generated summaries. From the table, it can be seen that our model produced a good summary. There was only an error caused by the phrase "*somalia's internationally recognized government that had been under pressure from al-shabaab*" because it was attached to the previous sentence, which was semantically distinct from the latter phrase. The adoption of the coverage vector corrected the error and the use of the coverage loss slightly improved the summary. The use of the *covscore* function modified part of the summary, including more details about the "beliefs" of the terrorist group.

Table 2. The Rouge-1 (R-1), Rouge-2 (R-2), and Rouge-L (R-L) F1 scores. † indicates that the model could not be directly compared to ours because it was trained and tested on the anonymized version of the dataset.

Model	**R-1**	**R-2**	**R-L**
See et al.'s pointer generation	29.7	9.21	23.24
See et al.'s pointer generation with coverage	36.44	15.66	33.42
Nallapati et al.'s pointer model †	32.1	11.7	29.2
Nallapati et al.'s hierarchical model †	31.8	11.6	28.7
Hsu et al.'s model	**40.68**	**17.97**	**37.13**
Tan et al.'s model	38.1	13.9	34.0
Our model	26.77	9.67	19.04
Our model + coverage	34.50	13.47	22.26
Our model + coverage + covscore	**34.67**	**13.61**	**22.36**
Our model + coverage + covloss	25.14	7.56	16.78

Figure 5 shows the average percentage of duplicate n-grams. The measure gives an insight into both the reasons behind the low Rouge F1 scores and the impact of the coverage vector; in detail, a model with duplicate tokens in the summary could have low Rouge F1 scores because those repetitions prevented the generation of relevant terms while stretching the summary until it satisfied the minimum length. We computed the average percentage of duplicate n-grams as follows: for each method, we counted the number of repeated n-grams (from 1-grams to 4-grams) in the generated summaries, and we divided this by the length of the summary. From the figure, it is possible to observe that the coverage vector reduced the repetitions. Such a value was further reduced with the application of the coverage loss but it did not improve the Rouge scores as reported in Table 2. Finally, the application of the *covscore* function produced summaries with an average n-gram repetition that was close to the reference ones.



Source Document

the terrorist group al-shabaab has claimed an attack on garissa university college in eastern kenya, in which many people have been killed and still more taken hostage. **the attack is another step in the ongoing escalation of the terrorist group's activities, and a clear indicator that the security situation in east africa is deteriorating fast.** somalia-based **al-shabaab has been behind a string of recent attacks in kenya, the most well-known of them being the massacre at the westgate shopping centre in** nairobi in 2013. [...]

Reference Summary

terrorist group al-shabaab has attacked a kenyan college, killing and taking hostages. **it is a clear indicator the security situation in east africa is deteriorating,** says stefan wolff. more than military action aloe is needed to combat terrorism in the region, he says.

Our model

al-shabaab has been behind a string of attacks in kenya, the most well-known of them being the massacre at westgate shopping centre in part of somalia's internationally recognized government that had been under pressure from al-shabaab. **the attack is another step in the ongoing escalation of the terrorist group's activities, and a clear indicator that the security situation in east africa is deteriorating fast.**

Our model + coverage

al-shabaab has been behind a string of recent attacks in kenya. the attack is another step in the ongoing escalation of the terrorist group's activities, and a clear indicator that the security situation in east africa is deteriorating fast.

Our model + coverage + coverage loss

al-shabaab has been behind a string of recent attacks in kenya. it has yet to recover almost quarter of a century later this year. **the attack is another step in the ongoing escalation of terrorist group's activities, and a clear indicator that security situation in east africa is deteriorating.**

Our model + coverage + covscore

al-shabaab has been behind a string of recent attacks in kenya. the attack is another step in the ongoing escalation of the terrorist group's activities. al-shabaab is predominantly driven by the same radical interpretation of the koran as al-qaeda and isis (also known as islamic state).

Figure 4. Examples of the generated summaries including the reference summary and an excerpt of the document.

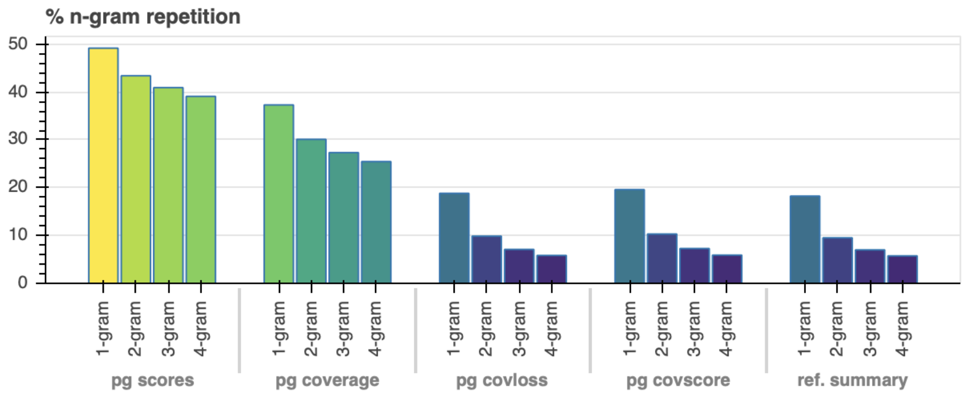


Figure 5. The average percentage of repeated n-grams (from 1-grams to 4-grams) for each sentence-level attention method and the reference summary. The term *pg* is the abbreviation of *PageRank*. The PageRank scores (or pg scores) refer to the base method, whereas *pg coverage* refers to the adoption of the coverage method in the base model.

Finally, we compared the summaries in terms of the *readability* and *relevance* scores; the former expresses the quality of the summary in terms of punctuation, syntax, and semantic coherency, whereas the latter expresses whether the summary captured all the salient information from the input document. Both metrics were in the range $[1, 10]$. Following Paulus et al. [6], we randomly selected 100 generated summaries by "*our model + coverage + covscore*" and we asked three annotators to evaluate them. For a better comparison, we also extracted the same summaries from Hsu et al. [22]'s model and Tan et al. [20]'s model.

Table 3 shows the inter-annotator agreement for both *readability* and *relevance*. From the table, we can see that our model had, in general, very high agreement, i.e., the annotators agreed on the same scores, with a *p*-value of 0.26. This was only surpassed by Hsu et al.'s model for the *relevance* score; however, in this latter model, the annotators were discordant for the *readability* score. Indeed, it had a *p*-value of 0.25. Finally, Tan et al.'s model had very low agreement, meaning that the annotators did not agree on the scores, with a *p*-value lower than 1×10^{-5}.

Table 3. The inter-annotator agreement on both the *readability* and *relevance* scores.

Model	Readability	Relevance
Our model + coverage + covscore	0.396	0.338
Hsu et al.'s model	0.064	0.471
Tan et al.'s model	0.148	0.017

Table 4 reports the results of the two metrics. The results show that Hsu et al. [22]'s model obtained very high scores because it selected the relevant sentences in an offline manner (the sentence extractor model). It is interesting to notice that the model of Tan et al. [20] had very low readability and relevance scores, despite obtaining very high Rouge scores. This means that the model generated summaries that would not completely satisfy human readers. Our model instead had a very high relevance score since it copied portions of the input document into the summary. However, the readability was not so high, meaning that it mostly truncated the copied sentences without merging them into a coherent summary. Also, our model outperformed Tan et al. [20]'s model for both the *readability* and *relevance* scores, despite obtaining lower Rouge scores.

Table 4. The average scores for readability and relevance for the 100 randomly extracted summaries. A higher score is better.

Model	Readability	Relevance
Our model + coverage + covscore	5.78	6.73
Hsu et al.'s model	7.27	7.85
Tan et al.'s model	5.06	4.84

4.2. Attention Scores Analysis

In this section, we evaluate the scores generated by the model both at the sentence level and word level. Plotting the scores helps to understand the model's behavior since neural networks are blackboxes. For our analysis, we selected a summary generated by "*our model + coverage + covscore*" from the testset.

Figure 6 shows the sentence-level scores generated by the network; it can be seen that the model mainly focused on the first sentence (dark colors) since it found it relevant for the summary. Thus, the model copied part of this sentence into the summary, as shown in Figure 7. These figures demonstrate the ability of the network to find the most informative sentence and to select the relevant words from it.

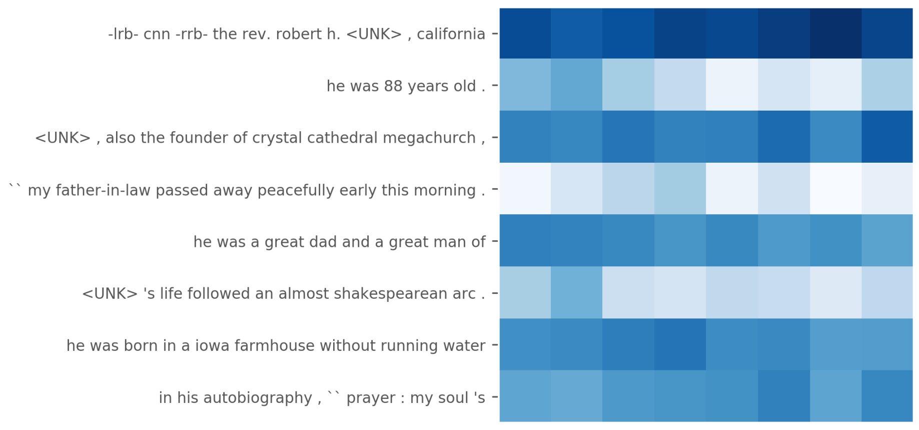


Figure 6. The ability of the model to select the most informative sentence.

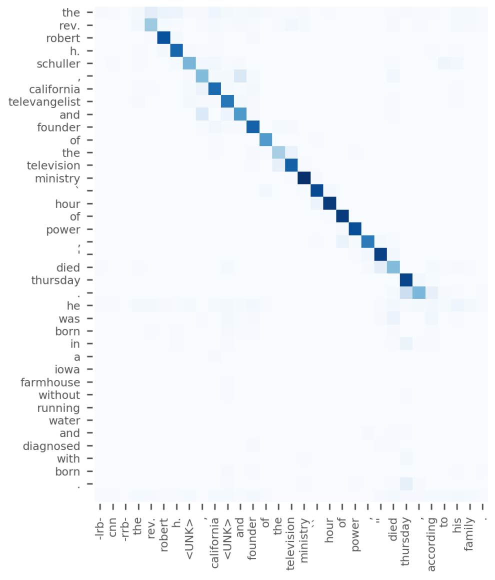


Figure 7. This shows that the model was able to copy part of the selected sentence into the summary.

Nevertheless, the model copied a large portion of the sentence rather than just selecting the relevant words. This could lead to swiftly generated summaries that miss the salient information since the model arrived at the maximum permitted length for the summary (which was set to 100 tokens) without being able to incorporate all relevant text and entities.

4.3. Abstractiveness of the Models

One important point to evaluate is the capability of the models to generate abstractive summaries. This requires the model to perform complex operations such as using synonyms and periphrasis, which could decrease the likelihood of matching the reference summaries. To evaluate the abstractiveness of the models, we calculated the average percentage of novel n-grams, which measures the capability of a neural network to generate n-grams that are not present in the reference summary. A low percentage of this measure means that the network is more conservative and tends to use words that appear in the document, whereas a high percentage means that the network is abstractive and tends to generate summaries that contain synonyms and periphrasis.

Figure 8 depicts the percentage of novel n-grams occurring in the generated summaries. We computed these percentages in the following way: for each n-gram (1-grams to 4-grams), we counted how many (unique) n-grams in the generated summary were not present in the reference summary. Then, we divided this by the total number of (unique) n-grams in the generated summary. The figure shows that our model had a high percentage of novel n-grams. These values remained stable with the adoption of the coverage vector, meaning that the model was using all the relevant words. Finally, the coverage loss increased the number of novel n-grams due to the generation errors of the model, i.e., the model tended to generate summaries that contained words not related to the topic of the document. The application of the *covscore* function slightly increased the number of novel n-grams.

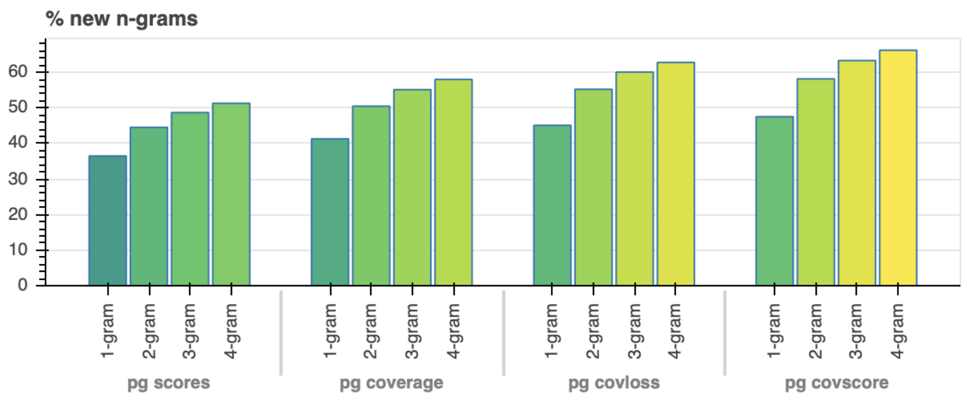


Figure 8. The average percentage of novel n-grams (from 1-grams to 4-grams) for each sentence-level attention method. The term *pg* is an abbreviation for *PageRank*. The PageRank score (or pg scores) refers to the base method, whereas *pg coverage* refers to the adoption of the coverage method in the base model.

We also computed the average β value of Equation (7) to check if the models were more oriented toward the vocabulary or the pointer network. Table 5 shows the average β value for each model, which was over 0.90. This value shows that the models used the vocabulary to generate the summaries, rarely relying only on the pointer network.

Table 5. The average β value for each model. High β values denote that the model tended to use the vocabulary distribution in the generation of the next word, whereas low β values denote that the model tended to copy words from the input document through the pointer network.

Model	**Avg. β Value**
Our model	0.92 ± 0.04
Our model + coverage	0.93 ± 0.03
Our model + coverage + covscore	0.93 ± 0.03
Our model + coverage + coverage loss	0.95 ± 0.03

5. Conclusions

In this paper, we presented a novel NN-based model for TS. The model used a hierarchical encoder to obtain both word and sentence representations. Then, it computed a score for each sentence that expressed its importance with respect to the other sentences. These scores were then used to adjust the word-level attention scores, ensuring that the model focused only on the important words. Furthermore, since we used a pointer network model, the model only copied the words coming from the informative sentences into the summary.

The model that used the coverage vector performed better than the models of Nallapati et al. [4]; however, our models had lower scores (about 2 Rouge points) than See et al. [7]'s models and did not surpass Hsu et al. [22]'s model and Tan et al. [20]'s model.

We found our model had high abstractive power, i.e., the ability to use synonyms and periphrases, as confirmed by both the novelty scores (see Figure 8) and the β values (see Table 5). Finally, we outperformed Tan et al. [20]'s model in terms of the *readability* and *relevance* scores [6].

In future works, we are interested in improving the recall of the model, which was its weakness. To increment the recall, we think that the use of topics generated by an LDA model [42] could be useful since they capture the semantic content and the domain style of the document, tying the summary to it. We also plan to improve the graph-based sentence-level attention, which showed very good results. We intend to follow Hsu et al. [22]'s idea, training the sentence-level attention (via loss function) to recognize the best sentences for the summary. However, the method proposed by Hsu et al. [22] cannot be directly adopted; we have to define a loss that can adapt to the generated summary, identifying the best sentences and penalizing the model in case it does not select them.

Another interesting research direction would be to integrate an external knowledge base (KB) with the model; in this case, we have to develop a classifier to discern the entities that are relevant to the summary from those that are not. This task would require the annotation of a dataset with the correct KB entities for the summary. We think that reasoning on the graph would reduce the copying behavior demonstrated by the pointer-network.

Finally, we plan to study how to improve the *covscore* function since it slightly increased the performance of the model while reducing the redundancy and attention errors.

Author Contributions: Conceptualization, G.S. and L.R.; methodology, G.S.; software, G.S.; validation, G.S. and L.R.; formal analysis, G.S. and L.R.; investigation, G.S.; writing—review and editing, G.S. and L.R.; visualization, G.S.; supervision, L.R.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Acknowledgments: Livio Robaldo has been supported by the Legal Innovation Lab Wales operation within Swansea University's Hillary Rodham Clinton School of Law. The operation has been partly funded by the European Regional Development Fund through the Welsh Government.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 3104–3112.
2. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 1724–1734.
3. Rush, A.M.; Chopra, S.; Weston, J. A neural attention model for abstractive sentence summarization. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 379–389.
4. Nallapati, R.; Zhou, B.; Gulcehre, C.; Xiang, B. Abstractive text summarization using sequence-to-sequence rnns and beyond. In Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL), Berlin, Germany, 11–12 August 2016; pp. 280–290.
5. Chopra, S.; Auli, M.; Rush, A.M. Abstractive sentence summarization with attentive recurrent neural networks. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016; pp. 93–98.
6. Paulus, R.; Xiong, C.; Socher, R. A deep reinforced model for abstractive summarization. *arXiv* **2018**
7. See, A.; Liu, P.J.; Manning, C.D. Get to the point: Summarization with pointer-generator networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, CR, Canada, 30 July–4 August 2017; Volume 1, pp. 1073–1083.
8. Wang, Y.; Zhang, H.; Liu, Y.; Xie, H. KG-to-text generation with slot-attention and link-attention. In *CCF International Conference on Natural Language Processing and Chinese Computing*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 223–234.
9. Hayashi, H.; Hu, Z.; Xiong, C.; Neubig, G. Latent relation language models. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 7911–7918.
10. Liu, Q.; Yogatama, D.; Blunsom, P. Relational Memory-Augmented Language Models. *Trans. Assoc. Comput. Linguist.* **2022**, *10*, 555–572.
11. Hu, Z.; Cheng, L.; Wang, D.; Niu, W.; Ma, J.; Mo, F. A Novel GCN Architecture for Text Generation from Knowledge Graphs: Full Node Embedded Strategy and Context Gate with Copy and Penalty Mechanism. In Proceedings of the International Conference on Frontiers of Electronics, Information and Computation Technologies, Wuhan, China, 19–21 August 2021; pp. 1–5.
12. Huang, X.; Zhang, J.; Li, D.; Li, P. Knowledge graph embedding based question answering. In Proceedings of the Twelfth ACM International Conference on Web Search and Data MINING, Melbourne, VIC, Australia, 11–15 February 2019; pp. 105–113.
13. Saxena, A.; Chakrabarti, S.; Talukdar, P. Question answering over temporal knowledge graphs. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics, Virtual Event, 1–6 August 2021; pp. 6663–6676.
14. Bosselut, M.Y.H.R.A.; Leskovec, P.L.J. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. *arXiv* **2021**
15. Koehn, P. *Statistical Machine Translation*; Cambridge University Press: Cambridge, UK, 2009.
16. Chen, Y.C.; Bansal, M. Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; pp. 675–686.
17. Cao, Z.; Li, W.; Li, S.; Wei, F. Retrieve, rerank and rewrite: Soft template based neural summarization. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; pp. 152–161.
18. Wang, K.; Quan, X.; Wang, R. BiSET: Bi-directional Selective Encoding with Template for Abstractive Summarization. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 2153–2162.
19. Zhou, Q.; Yang, N.; Wei, F.; Zhou, M. Selective encoding for abstractive sentence summarization. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, MN, Canada, 30 July–4 August 2017; Volume 1, pp. 1095–1104.
20. Tan, J.; Wan, X.; Xiao, J. Abstractive document summarization with a graph-based attentional neural model. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, MN, Canada, 30 July–4 August 2017; Volume 1, pp. 1171–1181.
21. Gehrmann, S.; Deng, Y.; Rush, A.M. Bottom-up abstractive summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 4098–4109.
22. Hsu, W.T.; Lin, C.K.; Lee, M.Y.; Min, K.; Tang, J.; Sun, M. A Unified Model for Extractive and Abstractive Summarization using Inconsistency Loss. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; pp. 132–141.
23. Li, W.; Xiao, X.; Lyu, Y.; Wang, Y. Improving Neural Abstractive Document Summarization with Structural Regularization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 4078–4087.

24. Li, W.; Xiao, X.; Lyu, Y.; Wang, Y. Improving Neural Abstractive Document Summarization with Explicit Information Selection Modeling. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 1787–1796.
25. Kryściński, W.; Paulus, R.; Xiong, C.; Socher, R. Improving Abstraction in Text Summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 1808–1817.
26. Liu, Y.; Lapata, M. Text summarization with pretrained encoders. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Hong Kong, China, 3–7 November 2019; pp. 3730–3740.
27. Song, K.; Tan, X.; Qin, T.; Lu, J.; Liu, T.Y. Mass: Masked sequence to sequence pre-training for language generation. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019.
28. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. 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), Minneapolis, MN, USA, 2–7 June 2019; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; pp. 4171–4186. https://doi.org/10.18653/v1/N19-1423.
29. Narayan, S.; Cohen, S.B.; Lapata, M. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 1797–1807.
30. Wang, Q.; Pan, X.; Huang, L.; Zhang, B.; Jiang, Z.; Ji, H.; Knight, K. Describing a Knowledge Base. In Proceedings of the 11th International Conference on Natural Language Generation, Tilburg, The Netherlands, 5–8 November 2018; Association for Computational Linguistics, Tilburg University: Tilburg, The Netherlands, 2018; pp. 10–21. https://doi.org/10.18653/v1/W18-6502.
31. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780.
32. Saxena, A.; Tripathi, A.; Talukdar, P. Improving multi-hop question answering over knowledge graphs using knowledge base embeddings. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 4498–4507.
33. Mikolov, T.; Yih, W.t.; Zweig, G. Linguistic regularities in continuous space word representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MI, USA, 9–14 June 2013; pp. 746–751.
34. Pennington, J.; Socher, R.; Manning, C. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 26–28 October 2014; pp. 1532–1543.
35. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* 2014
36. Yao, L.; Mao, C.; Luo, Y. Graph convolutional networks for text classification. In Proceedings of the AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 8–12 October 2019; Volume 33, pp. 7370–7377.
37. Hermann, K.M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; Blunsom, P. Teaching machines to read and comprehend. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 1693–1701.
38. Chen, D.; Bolton, J.; Manning, C.D. A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task. *arXiv* **2016**, arXiv:1606.02858
39. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
40. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
41. Lin, C.Y. Rouge: A package for automatic evaluation of summaries. *Text Summ. Branches Out* **2004**, *2004*, 74–81.
42. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.