Summarizing Long-Form Document with Rich Discourse Information

Sara Moreno
Politecnico di Torino
Student ID: s283832
sara.moreno@studenti.polito.it

Stefano Galvagno
Politecnico di Torino
Student ID: s290191
s290191@studenti.polito.it

***Abstract*—This work tries to implement HEROES [1] that exploits the rich discourse structure of long-form documents in order to empower the extractive summarization task. The approach can be decomposed into two fundamental modules: the Content Ranking Module and the Extractive Summatization Module. The former assigns a score to each section and to each sentence on the basis of how useful the section or the sentence can be for the summary extraction. By ranking sections and sentences according to the respective scores and by selecting only salient sentences from important sections, one can create a brief digest that serves as input to the complex Extractive Summatization Module. The latter is based on the use of heterogeneous graphs, able to reflect the document discourse structure of the document. This algorithm leads to two major achievements: 1) it fully exploits the particular structure of scientific documents; 2) the Content Ranking Module allows the selection of phrases, not according to their position in the text, but rather based on their actual importance.**
Despite the limitations in terms of resources and, consequently, the small size of the data sets, we think that the model achieves results that can be considered proportionally comparable to those of HEROES.

***Index Terms*—Content Ranking, Extractive Summarization, heterogeneous graph, long-form document**

I. Introduction

Summarization refers to the generation of a concise representation of a text. In particular, there are two methods: abstractive and extractive summarization. The first method consists in generating text from scratch, emulating the typical human behaviour. An important drawback is that the generation of new text entails the possibility of producing confusing and out-of-context content. For this reason we adopted the second approach. This, in fact, aims at selecting salient sentences from the text to be summarized, without generating them from scratch.
Traditional techniques tend not to exploit important information such as the structure of the text (i.e. hierarchy in terms of sections and sentences). In addition, such models, due to memory constraints, are forced to cut the text, excluding the final parts. Indeed, the intrinsic problem of long-form documents summarization is the inability to analyze the entire content of a document, potentially loosing summary worthy parts. It cannot be assumed a priori that these parts are less relevant (e.g. the conclusions section of scientific papers tends to contain important information about the whole paper).
In light of this, we implemented from scratch a model able to to mitigate these two problems. The code is available at the following link: https://github.com/MorenoSara/Summarizing-Long-Form-Document-with-Rich-Discourse-Information.
In particular, we exploit a module called Content Ranking, which aims at evaluating the usefulness of sections and sentences in order to extract the final summary. This allows us to handle the problem of memory constraints. With a second module, named Extractive Summarization, starting from the document resulting from the previous module, we finally extract the salient sentences from their sections and concatenate them to form the generated summary for each document in input. The sentence selection is done by exploiting heterogeneous graphs. These structures model the digested text with three types of nodes, for words, sentences and sections, and six types of arcs to maintain the hierarchical relationships between them.
Experiments are carried out on the arXiv and PubMed data sets, which are both commonly used for long-form document summarization.

II. Methodology

In this section we will initially describe the overall pipeline, then we will deeply focus on each module.

A. Overview of the NLP architecture

As shown in Figure 1 the original document is given as input to the Content Ranking Module that extracts the salient m sections, and within them selects the n more informative sentences. The output of this first module is given by the concatenation of these sections and sentences that form the so-called digested document, which maintains the order in which these elements appear in the original document. The second module is fed with the output of the first, in order to extract the summary-worthy sentences that will constitute the final result.

B. Content Ranking Module

This module deals with the extraction of the m sections, each containing n sentences, characterized by the highest importance scores. These are attributed on the basis of the similarity of each section, and sentence contained in it, to the gold summary, i.e. the abstract. The initial representation of

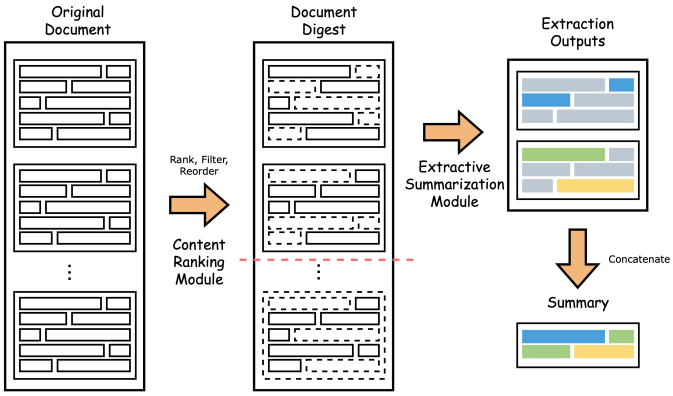

Fig. 1. **The overall architecture of the complete summarization task. Dashed borders indicate sentences and sections that are filtered out by the Content Ranking Module. Colours highlight sentences that are selected by the Extractive Summatization Module**

each word is given by its 300 dimensional GloVe embedding.

Title representation The title can simply be interpreted as a sequence of words $\{w_1^t, w_2^t, ..., w_l^t\}$ and thus be treated as such. In particular, from this first representation, we use a BiLSTM to obtain contextualized word representations $[h_1^t, h_1^t, ..., h_1^t]$. Then we use the *Attention* mechanism, described in Equation (1), in order to highlight the semantically meaningful parts of the input data and diminishing the others to obtain the final title representation v_t.

$$\begin{aligned} u_i &= tanh(W_{att} h_i^t + b_{att}) \\ \alpha_i &= \frac{e^{(u_i^T u_{att})}}{\sum_i e^{(u_i^T u_{att})}} \\ v_t &= \sum_i \alpha_i h_i^t \end{aligned} \tag{1}$$

Content representation The representation of the content is achieved by aggregating important elements according to their hierarchical relationships. In particular, important words are aggregated into sentence representations, then, these representations are aggregated using the same mechanism into section representations. In particular the following steps are applied for each section, according also to [2].

First, a word-level BiLSTM is used to encode the words, then a word-level attention layer is used to obtain the sentence embedding. Given the i-th sentence of the considered section $s_i = \{w_{i1}^c, w_{i2}^c, ..., w_{iL_i}^c\}$ the contextualized word representations h_{ij}^c and the sentence embeddings h_i^c are computed following the steps of Equation (2).

$$\begin{aligned} \{h_{i1}^c, h_{i2}^c, ...h_{iL_i}^c\} &= BiLSTM_w^c(\{w_{i1}^c, w_{i2}^c, ..., w_{iL_i}^c\}) \\ h_i^c &= Attention(\{h_{i1}^c, h_{i2}^c, ...h_{iL_i}^c\}) \end{aligned} \tag{2}$$

Where the *Attention* mechanism is the same described in Equation (1) but uses a different set of parameters, learnt during the training phase.
The obtained sentence embeddings h_i^c are passed through a sentence-level BiLSTM in order to obtain the sentence representations $\hat{h}_i^c$. After having obtained all of them, a sentence-level *Attention* layer is applied on top of all these elements. This operation allows to obtain the context representation of the considered section v_c. This part is outlined in Equation (3).

$$\begin{aligned} \{\hat{h_1^c}, \hat{h_2^c}, ...\hat{h_L^c}\} &= BiLSTM_s^c(\{h_1^c, h_2^c, ..., h_L^c\}) \\ v_c &= Attention(\{\hat{h_1^c}, \hat{h_2^c}, ...\hat{h_L^c}\}) \end{aligned} \tag{3}$$

The whole process followed for the section representation is pictured in Figure 2.

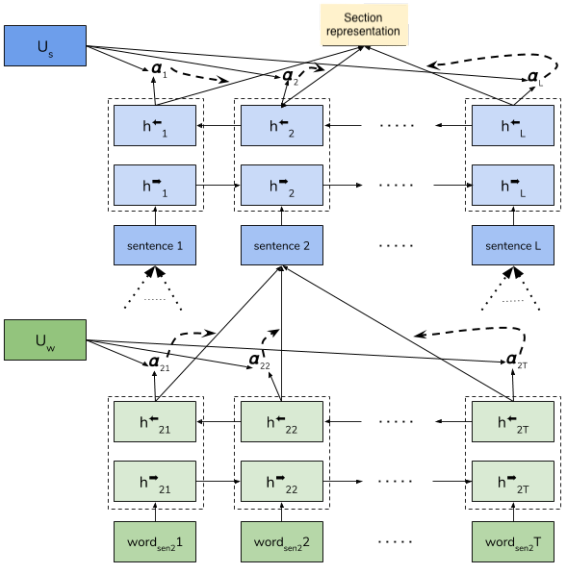

Fig. 2. **Process to obtain section representation $\mathbf{v_c}$. $\mathbf{U_w}$ and $\mathbf{U_s}$ are the context vectors of the word- and sentence-level Attention layers respectively.**

Section and sentence scoring For each sentence, in order to estimate the sentence score, we use a feed forward neural network, followed by a *sigmoid* activation function that bounds the scores in the range $[0, 1]$.

$$\hat{y}_i^s = \sigma(w^{s^T} \hat{h}_i^c + b^s) \tag{4}$$

For what concerns the section scoring, exploiting both the title and the content representation is crucial, therefore, they are vertically concatenated. On top of them a new *Attention* layer is applied to obtain a merged representation of them. On the basis of that representation the section score is estimated similarly to the sentence one.

$$\begin{aligned} v_f &= Attention(\{v_t, v_c\}) \\ \hat{y}^S &= \sigma(w^{S^T} v_f + b^S) \end{aligned} \tag{5}$$

Training The evaluation metric adopted is the binary cross entropy loss, evaluated considering the estimated importances $\hat{y}_i^s$ and $\hat{y}^S$, with respect to the ground truth labels y_i^s and y^S. The sections true labels y^S are computed through the ROUGE-2 recall of the section against the abstract. Whereas the ground truth labels of the sentences y_i^s are found by computing the average value of the ROUGE-1/-2/-L F1 for each sentence. The total loss of the Content Ranking Module is the sum of the BCE losses between the section scores and the one between the sentence scores.

Inference During inference we took the most informative m sections on the basis of the estimated score $\hat{y}^S$. Within these sections we selected the n best sentences, again according to the computed value $\hat{y}_i^s$. We concatenated all these sentences of the sections in a new short digested document that preserves the same order of the original one. For our experiments we considered m = 2 and n = 4.

C. Extractive Summatization Module

For each digested document we create a heterogeneous graph with three semantic nodes and six edge types. We build six different Graph Attention Networks[1] (GAT) to correctly handle all the possible sub-graphs. Each GAT has a specific number of heads that are merged together, producing new representations of the target nodes. The representations of the semantic nodes are iteratively updated applying the algorithm T times, in our experiments we used T = 2. Nodes representations that are produced by multiple GATs are merged by means of the $fusion$ [3] operation. Each obtained representation is passed through a Position-Wise Feed Forward neural network [4], that takes the new node representation plus the residual connection of the original representation.

Heterogeneous graph Each graph has three different semantic nodes for words, sentences and sections. And six elaborately designed directed edges:

1) *word-to-sentence* edges: these edges go from the word to the sentence nodes representing sentences that contain that word. The aim is to preserve the hierarchical relationships between words and sentences;
2) *sentence-to-word* edges: are edges from the sentence to the words contained into it. The idea is to exploit the word overlapping between sentences to establish relationships between sentences;
3) *intra-section sentence-to-sentence* edges: for each section we create connections for each pair of sentences within that section in both directions. By means of these edges, that allow free interaction of sentences within the section, one can understand the local importance of the sentences;
4) *cross-section section-to-sentence* edges: for each sentence of a given section build edges that go from all the other sections to that sentence. In that way the global context is taken into account for each sentence representation update;
5) *intra-section sentence-to-section* edges: for each sentence of a section there are edges from the sentences appearing in the section to the corresponding section node. With these edges the hierarchical structures present among sentence and section are preserved;
6) *section-to-section* edges: we build edges for each pair of sections in both directions. The idea is that each section is related to a specific topic, and with these connections relationships between the topical information are maintained.

By means of this complex representation it is possible to extract sub-graphs on the basis of the edge types, each one identified with a unique id.

Node representation initialization Each semantic node has an initial representation.
For what concerns the word nodes we exploited the pre-trained `GloVe` embeddings, already used as word initialization for the Content Ranking Module.
Each sentence initial representation is created by means of a CNN encoder with kernel sizes that range from 2 to 7, each one with 50 channels as in [5]. After that, the representations are passed through a 2-layer `BiLSTM`. The obtained representation is then concatenated with the vector representation of the boundary distance feature.
The boundary distance feature is computed as follows:

$$f = \frac{|p - (l/2)|}{l/2} \tag{6}$$

where p is the position of the sentence in the corresponding section and l is the total number of sentences within the section.
For the sections, instead, we take the aforementioned initial representation of the sentences, we apply the same $Attention$ mechanism described in Equation (1) to highlight the most informative sentences. Then, on top of these representations a 2-layer `BiLSTM` is applied.
All these representations are later mapped into same dimension representations using different Linear layers for each semantic node, finally obtaining $H_{w|s|S}$.

Iterative nodes update using GAT
Given the previously described graph and initial representations we use the Graph Attention Network to update the node representations. We create a different GAT layer for each of the six sub-graphs, every layer follows the operations of Equation (7).

$$\begin{aligned} e_{ij} &= LeakyReLU(\vec{a}^T[Wh_i||Wh_j]) \\ \alpha_{ij} &= \frac{exp(e_{ij})}{\sum_{j' \in \mathcal{N}_i} exp(e_{ij'})} \\ u_i &= ELU(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W h_j) \end{aligned} \tag{7}$$

[1]Code taken from the DGL tutorial

where W and $\vec{a}$ are trainable parameters, $||$ denotes the concatenation, $\mathcal{N}_i$ denotes the neighbouring nodes of i. This is a single-head GAT layer, in our implementations we used $K = 6$ heads for the layers that contribute to word updates and $K = 8$ head for those related to sentences and sections representations. To deal with the multi-head attention, we performed K independent single-head layers and concatenated their outputs as follows:

$$u_i = ||_{k=1}^{K} ELU(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k h_j) \tag{8}$$

The representations of each semantic node are updated iteratively.
First let's consider the word nodes. Only one edge type has as target the words, the *sentence-to-word* edges. Consequently, the update only depends on the sentence states. Particularly, it is computed as follows:

$$\begin{aligned} U_{w \leftarrow s}^{t+1} &= GAT(H_w^t, H_s^t, H_s^t | G_{w \leftarrow s}) \\ H_w^{t+1} &= FFN(U_{w \leftarrow s}^{t+1} + H_w^t) \end{aligned} \tag{9}$$

where t represents the iteration, H_w and H_s denote the representations of word and sentence semantic nodes, respectively. $G_{w \leftarrow s}$ represents the sub-graph obtained considering the edges from sentences to words only.
Furthermore, $GAT(H_w^t, H_s^t, H_s^t | *)$ means that H_w^t is used as the attention query and H_s^t as key and value.
For what concerns sentence nodes, differently from the previous case, there are several edges that present sentences as destinations. Specifically, *word-to-sentence* edges, *intra-section sentence-to-sentence* edges, *cross-section section-to-sentence* edges. Consequently, the updating process for sentence nodes is determined by the states of all these three entities.
Analogously as before:

$$\begin{aligned} U_{s \leftarrow w}^{t+1} &= GAT(H_s^t, H_w^t, H_w^t | G_{s \leftarrow w}) \\ U_{s \leftarrow s}^{t+1} &= GAT(H_s^t, H_s^t, H_s^t | G_{s \leftarrow s}) \\ U_{s \leftarrow S}^{t+1} &= GAT(H_s^t, H_S^t, H_S^t | G_{s \leftarrow S}) \end{aligned} \tag{10}$$

Being a threefold contribution, all the representations are merged by means of a (double) fusion operation, that is defined as follows:

$$\begin{aligned} z &= \sigma(W_f[x||y] + b_f) \\ \textit{fusion} &= zx + (1 - z)y \end{aligned} \tag{11}$$

where σ denotes the *sigmoid* function, x and y are the vectors to be merged (e.g. $U_{s \leftarrow w}^{t+1}$ and $U_{s \leftarrow S}^{t+1}$). To summarize:

$$U_s^{t+1} = fusion(fusion(U_{s \leftarrow w}^{t+1}, U_{s \leftarrow S}^{t+1}), U_{s \leftarrow s}^{t+1}) \tag{12}$$

and, like before:

$$H_s^{t+1} = FFN(U_s^{t+1} + H_s^t) \tag{13}$$

To conclude, section nodes receive updates through *intra-section sentence-to-section* edges and *section-to-section* edges.

```
{
    'abstract_text': List[str],
    'section_names': List[str],
    'sections': List[List[str]]
}
```

Listing 1: JSON structure of each document.

Consequently these nodes are updated with the same criterion used so far:

$$\begin{aligned} U_{S \leftarrow s}^{t+1} &= GAT(H_S^t, H_s^t, H_s^t | G_{S \leftarrow s}) \\ U_{S \leftarrow S}^{t+1} &= GAT(H_S^t, H_S^t, H_S^t | G_{S \leftarrow S}) \\ U_S^{t+1} &= fusion(U_{S \leftarrow s}^{t+1}, U_{S \leftarrow S}^{t+1}) \\ H_S^{t+1} &= FFN(U_S^{t+1} + H_S^t) \end{aligned} \tag{14}$$

Sentence extractor Once the updates of the different states of the nodes are concluded, the final states of the sentence nodes H_s^T are obtained. From these, the extraction probabilities are computed through a classification layer with $sigmoid$ activation function.
The j-th sentence of the i-th section has extraction probability equal to:

$$p_{ij} = \sigma(W_p h_{ij} + b_p) \tag{15}$$

Training The Extractive Summarization Module outputs an extraction probability to each sentences and is trained by means of the cross entropy loss between its output and the ground truth labels. These labels are found in the following way: for each pair of sentences, one belonging to the document and the other to its golden summary, it is computed the average ROUGE F1 score. Then the k sentences related to the highest values are selected and labeled with 1. All the other sentences have ground truth labels equal to 0. The parameter k has a default value of 2.

Inference During inference the sentences that are assigned with the highest probabilities are selected and concatenated into a summary. The summary is limited to a maximum length of 200 word. To allow a better qualitative evaluation of the model, during inference a new file is created with a pair of texts for each document. For every golden summary it is reported its text and the respective extracted one.

III. Experiments

A. Data description

The model is designed to work with documents in the JSON format. Each document has to contain the fields reported in Listing 1. Two large-scale scientific data sets have been used, both presented by [6]: *PubMed* and *arXiv*.
We used the same data structure for both data sets. It consists of a list of nested classes: the outermost one is the Document class, characterized by a list of Section objects. Similarly,

each instance of the class Section contains in turn a list of Sentence elements. A more detailed description is proposed in Appendix A.

B. Experimental design

Both the components of the model exploit the GPU. The framework with which the model is developed is `PyTorch` and the main libraries needed are: `numpy` [7], `rouge` to use the ROUGE score [8] for the model evaluation, `dgl` to exploit the Deep Graph Library [9] for the extractive summarization module and `dgl-cu101` to allow its usage on the GPU.

For what concerns the Content Ranking Module the batch size is set to 25, the learning rate to 10^{-4}, the module is trained for 5 epochs. The number of sections to be extracted, m is set to 2 by default, whereas the number n of sentences to extract from each of the m sections is set to 4.

For what concerns the Extractive Summarization Module, the values of batch size, learning rate and number of epochs are the same of the previous module. To deal with a different number of heads for the various GAT layers we want to highlight that it is necessary to set the output size of the model equal to a common multiple of the two chosen numbers, in our case it is set to 264.

Both the modules are trained to minimize the Cross Entropy loss.

The validation approach is different for the two modules. The Content Ranking selects the model with the lowest loss, while the Extractive Summarization Module saves the model with the highest average ROUGE F1.

The entire model is evaluated on the test set by means of ROUGE score. Since we are not dealing with the case of summaries of fixed length, to quantify performance, we mainly considered the F1 score. Indeed, the summaries are unlikely to be as large as the threshold, since the model is set up not to include sentences that would exceed the allowed word limit in the summary. Furthermore, we opted for focusing on the ROUGE-L score to measure the fluency of the summary.

C. Extensions

As an extension we opted for **data enrichment** by merging the subsets of *PubMed* and *arXiv*. To do that, we extracted half of the documents from the former and half from the latter. Consequently, the dimensions of the training, test and evaluation sets are kept constant throughout all the experiments, in order to be consistent in the performance evaluation. On each set, a shuffle was performed so that there was no long sequences of data from the same source.

As a further extension we also performed is the **ablation study**. In order to investigate the effectiveness of the main components of the model, one can try to disregard the Content Ranking Module and compare the results of the test set, without re-training the network. Particularly, the very first m sections and the respective n beginning sentences of each document are considered as the input for the Extractive Summarization Module. Furthermore, we excluded the boundary distance feature during the sentence embedding computation and in addition, set the number of node updates to just one.

D. Execution times

To obtain and apply the entire model on arXiv, a total of 56 minutes were required, of which 41 were needed to load the documents into the data structure and create the tokenizer, 2 for the Content Ranking Module training and the remaining 13 for the training of the Extractive Summarization module. From this we can conclude that the creation of the data structure represents the bottleneck. For what concerns the Extractive Summarization module, disabling the update of the node states, we observe a reduction of the training time to 10 minutes.

Using PubMed the acquisition of data required 29 minutes, less than those needed with arXiv probability due to their lower length on average. The training of the Content Ranking took 2 minutes and the one of the Extractive Summarization module took 21 minutes, for a total time of 52 minutes. Disabling the iterative update of the node states in the Extractive Summarization Module a training time of 17 minutes is reached.

To conclude, considering the combination of PubMed and arXiv, the execution of the model took 57 minutes. Specifically, 38 were needed for loading the data set and creating the tokenizer, 2 minutes for training the Content Ranking module and 17 for the Extractive Summarization. Disregarding the updates of the latter module, the training phase took 13 minutes.

E. Obtained results and their analysis

Regarding the results related to arXiv, we can observe that they tend to be worse than those obtained with PubMed. A conceivable reason can be that the former is characterized with a greater number of formulas therefore, on average, less similar to the respective golden summaries.

More specifically, with arXiv we obtain a ROUGE-L F1 of 0.231, versus 0.300 in PubMed. Eliminating instead the Content Ranking Module we obtain 0.255 for arXiv and 0.311 with PubMed. As you can see, the model seems to slightly improve. Probably the reason is the inadequacy of the size of the data sets for training imposed by the limitation of resources or the limited number of training epochs.

For the same reason, not updating node states does not result in a visible deterioration in performance using both the data sets.

Instead, concerning the boundary distance feature, it can be seen that removing it, there is a deterioration of performance using PubMed but not with arXiv. In fact, with PubMed we obtain a ROUGE-L F1 of 0.296, whereas with arXiv of 0.234. More detailed results are reported in Appendix B.

Combining arXiv and PubMed samples, we obtain results

that are halfway between those resulting from the use of the individual data sets considered separately. More precisely, the ROUGE-L F1 score assumes a value of 0.271 for the complete module, 0.282 disregarding the Content Ranking Module, 0.271 locking the node updates and 0.271 without considering the boundary distance feature.
A more direct comparison can be done by means of Table I.

	PubMed	arXiv	PubMed + arXiv
Complete	0.300	0.231	0.271
No CR	0.311	0.255	0.282
T = 1	0.300	0.232	0.271
No BD	0.296	0.234	0.271

TABLE I
COMPARISON BETWEEN ROUGE-L F1 SCORES OBTAINED USING ALL THE DATA SETS ALONG WITH ALL THE EXTENSIONS. CR STAND FOR CONTENT RANKING MODULE AND BD IS USED TO INDICATE THE BOUNDARY DISTANCE FEATURE.

IV. CONCLUSION AND TAKEAWAYS

We implemented from scratch a model composed by two modules. The first one is the Content Ranking Module, which allowed us to select in an appropriate way the sentences of a long-form document, in order to provide them as input to the second module. The latter, the Extractive Summarization module, exploits the heterogeneous graph that considers nodes at three different semantic levels and six different types of edges to model the interactions between the nodes.
Analyzing the execution times of each step of the model it is possible to easily notice that the bottleneck is the data structure, its creation takes a lot of time with respect to the training time. A possible further work could be to find a better structure to speed up the whole model.
Unfortunately, the ablation study did not report the expected results. We think that the reason is related to the limited computational resources that forced us to consider only a subset of the data sets, considerably limited compared to those adopted by the authors of the paper or it can be due to the reduced number of training epochs.
Overall, this project allowed us to learn how to manage a deep learning framework like `PyTorch` and to exploit the huge potential of graphs in the NLP context, enriching our knowledge.

REFERENCES

[1] Tianyu Zhu, Wen Hua, Jianfeng Qu, and Xiaofang Zhou. *Summarizing Long-Form Document with Rich Discourse Information*, page 2770–2779. Association for Computing Machinery, New York, NY, USA, 2021.

[2] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California, June 2016. Association for Computational Linguistics.

[3] Hanqi Jin, Tianming Wang, and Xiaojun Wan. Multi-granularity interaction network for extractive and abstractive multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6244–6254, Online, July 2020. Association for Computational Linguistics.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[5] Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. Heterogeneous graph neural networks for extractive document summarization. *CoRR*, abs/2004.12393, 2020.

[6] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, W. Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In *NAACL*, 2018.

[7] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[8] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[9] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks, 2019.

APPENDIX

A. Data structure

Since creating the data structure takes a considerable amount of computational time, it is worth describing it. As we can see in Figure 3, the structure looks like a list of instances of the **Document** class. Each of them has two fields inside:

- *abstract*: instance of the Section class, represents the golden summary
- *sections*: list of instances of class Section

Each **Section** object consists of:

- *all sentences*: string that contains the concatenation of the sentences of the section;
- *title*: title of the section considered;
- *y_S*: ground truth label for the section in the Content Ranking Module;
- *sentences*: list of objects of the Sentence class.

Each **Sentence** object is divided into:

- *sentence*: string representing the sentence itself;
- *tokenized sentence*: tokenized version truncated/padded to a predefined length;
- *label*: ground truth for the Extractive Summarization Module;
- *y_s*: ground truth label for the sentence in the Content Ranking Module.

B. Complete results

Here the complete ROUGE scores are reported, for the three the data sets along with all the performed extensions.

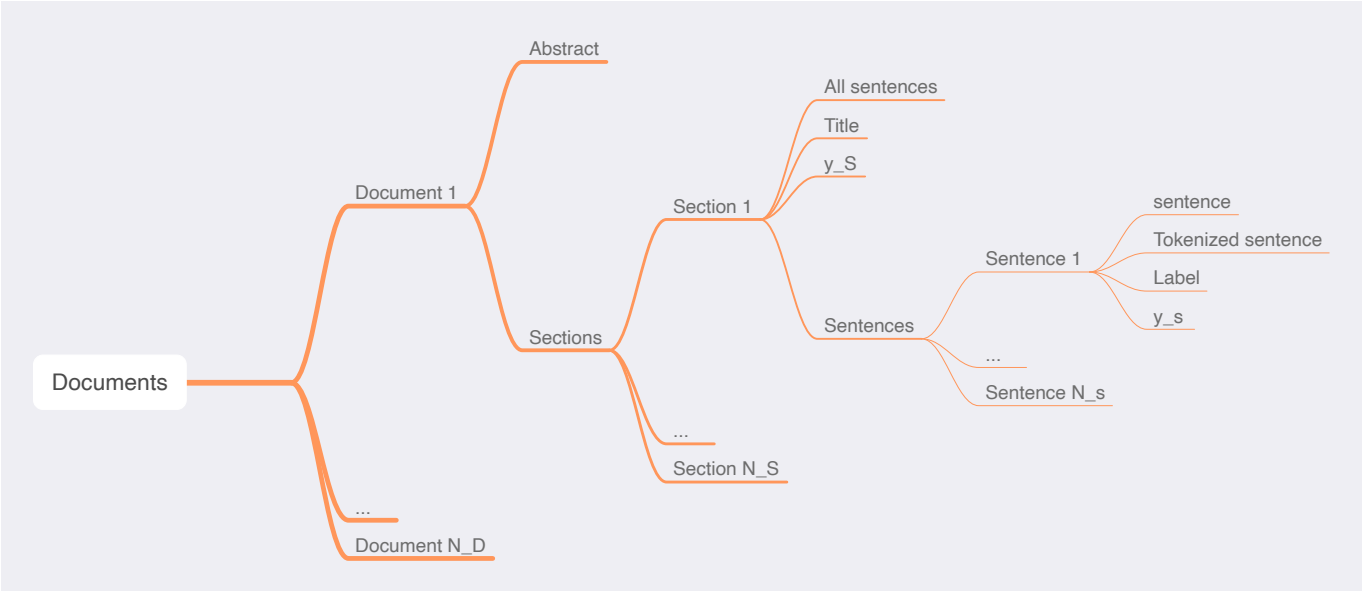

Fig. 3. Detailed data structure of both the data sets.

	Precision	Recall	F1
ROUGE-1	0.236	0.326	0.266
ROUGE-2	0.059	0.083	0.066
ROUGE-L	0.206	0.283	0.231

TABLE II
MODEL TRAINED WITH ARXIV

	Precision	Recall	F1
ROUGE-1	0.254	0.368	0.293
ROUGE-2	0.070	0.107	0.081
ROUGE-L	0.221	0.321	0.255

TABLE III
MODEL PERFORMANCE WITHOUT CONTENT RANKING MODULE USING THE ARXIV DATA SET

	Precision	Recall	F1
ROUGE-1	0.237	0.324	0.266
ROUGE-2	0.060	0.083	0.067
ROUGE-L	0.207	0.282	0.232

TABLE IV
MODEL PERFORMANCE WITHOUT ITERATIVE UPDATING (T=1) IN THE EXTRACTIVE SUMMARIZATION MODULE USING THE ARXIV DATA SET

	Precision	Recall	F1
ROUGE-1	0.240	0.320	0.267
ROUGE-2	0.060	0.0.080	0.067
ROUGE-L	0.210	0.281	0.234

TABLE V
MODEL PERFORMANCE WITHOUT BOUNDARY DISTANCE FEATURE USING THE ARXIV DATA SET

C. Example of obtained summary compared with the golden one

Here is reported a single example of obtained summary using the PubMed data set.

Reference:

Acute occlusion of the superior mesenteric artery (sma) causes extensive bowel necrosis, resulting in a poor prognosis with an extremely high mortality rate. An 82-year-old woman was admitted to our hospital with the complaint of abdominal pain. She was diagnosed as having acute sma occlusion by enhanced ct. Five hours from onset, the first thrombolytic therapy with urokinase was performed, but failed to complete thrombolysis and recanalization of peripheral blood flow. An exploratory laparotomy following the first thrombolytic therapy showed a mild ischemic change in the affected intestine and mesentery, but no sign of necrosis. After the laparotomy, local thrombolytic therapy with angiographic evaluation of blood flow at 24, 36 and 48 h from the first thrombolysis was performed. As a result, the residual thrombus disappeared and all branches of the sma became well visualized. The patient was discharged well without a second-look operation or major bowel resection. Sequential

	Precision	Recall	F1
ROUGE-1	0.314	0.373	0.331
ROUGE-2	0.117	0.134	0.119
ROUGE-L	0.285	0.339	0.300

TABLE VI
MODEL TRAINED WITH **PUBMED**

	Precision	Recall	F1
ROUGE-1	0.331	0.382	0.340
ROUGE-2	0.146	0.148	0.136
ROUGE-L	0.304	0.349	0.311

TABLE VII
MODEL PERFORMANCE WITHOUT CONTENT RANKING MODULE USING THE **PUBMED** DATA SET

intermittent thrombolytic therapy with meticulous angiographic evaluation of blood flow is effective for early-stage acute sma occlusion.

Extracted summary:
Acute occlusion of the superior mesenteric artery (sma) causes extensive intestinal necrosis due to the difficulty of early diagnosis, resulting in poor prognosis, with a high postoperative mortality rate of 65.2%. A mild ischemic change was observed at the intestinal wall from the jejunum 40 cm distal from the treitz ligament to the ascending colon, but no apparent necrosis. Early thrombolytic therapy can not always induce complete thrombolysis, and even if intestinal necrosis is avoided by administration of initial thrombolytic therapy, indications for additional thrombolytic therapy or the method for monitoring intestinal viability during subsequent follow-up have not been established. In this report, we present a case of acute sma occlusion diagnosed early after onset that was successfully treated by sequential and intermittent thrombolytic therapy by intraarterial urokinase infusion with angiographic evaluation of blood flow, thereby avoiding intestinal resection. Recent reports indicate that selective thrombolytic therapy with intraarterial infusion of urokinase is effective for acute sma occlusion diagnosed early after onset. 24 h after the laparotomy, a second angiography was performed via the catheter that remained in place after first angiography, because clinical signs such as abdominal pain suggested

	Precision	Recall	F1
ROUGE-1	0.317	0.371	0.331
ROUGE-2	0.118	0.134	0.120
ROUGE-L	0.288	0.337	0.300

TABLE VIII
MODEL PERFORMANCE WITHOUT ITERATIVE UPDATING (T=1) IN THE EXTRACTIVE SUMMARIZATION MODULE USING THE **PUBMED** DATA SET

	Precision	Recall	F1
ROUGE-1	0.312	0.365	0.326
ROUGE-2	0.113	0.127	0.114
ROUGE-L	0.283	0.331	0.296

TABLE IX
MODEL PERFORMANCE WITHOUT BOUNDARY DISTANCE FEATURE USING THE **PUBMED** DATA SET

	Precision	Recall	F1
ROUGE-1	0.293	0.335	0.302
ROUGE-2	0.090	0.100	0.090
ROUGE-L	0.262	0.300	0.271

TABLE X
MODEL TRAINED WITH **PUBMED+ARXIV** DATA SET

	Precision	Recall	F1
ROUGE-1	0.294	0.366	0.314
ROUGE-2	0.105	0.121	0.105
ROUGE-L	0.265	0.328	0.282

TABLE XI
MODEL PERFORMANCE WITHOUT CONTENT RANKING MODULE USING THE **PUBMED+ARXIV** DATA SET

	Precision	Recall	F1
ROUGE-1	0.294	0.335	0.303
ROUGE-2	0.090	0.100	0.091
ROUGE-L	0.263	0.300	0.271

TABLE XII
MODEL PERFORMANCE WITHOUT ITERATIVE UPDATING (T=1) IN THE EXTRACTIVE SUMMARIZATION MODULE USING THE **PUBMED+ARXIV** DATA SET

	Precision	Recall	F1
ROUGE-1	0.293	0.335	0.302
ROUGE-2	0.090	0.100	0.090
ROUGE-L	0.262	0.300	0.271

TABLE XIII
MODEL PERFORMANCE WITHOUT BOUNDARY DISTANCE FEATURE USING THE **PUBMED+ARXIV** DATA SET

the progression of intestinal ischemia .Blood flow in the mesentery was well palpable at the central portion of the sma, while peripheral blood flow was not palpable. The operation was completed without intestinal resection or direct removal of the thrombus from the sma.