POLITECNICO DI TORINO

Master's Degree in Ingegneria Informatica

Master's Degree Thesis

Scientific Papers Slide Generation using Abstractive Text Summarization

Supervisors

Prof. Luca CAGLIERO

Dott. Moreno LA QUATRA

Candidate

Simone MANNI

December 2021

Abstract

Slides-based presentations are increasingly important in scientific dissemination as they incorporate several useful information for publication understanding. They usually contains short summaries of the main paper contributions and cover all the sections of the original publication. Manually generating slides content, however, is an expensive task. Recent advancements in machine learning and artificial intelligence allowed the creation of automatic systems that aims at generating summaries from scientific articles. Those summaries can be used to reduce the amount of content that requires manual analysis. Limited research efforts have been devoted to the scope of generating presentation slides from document. Indeed, this task faces the scarcity of publicly available material for benchmarking. This master thesis proposes a new dataset, APPreD, consisting of pairs of papers and their corresponding presentation slides crawled from ACL online anthology. It also proposes a deep-learning based approach that addresses document-to-slide task. The proposed methodology entails (i) the classification of academic content in IMRaD classes, (ii) the fine-tuning of a pre-trained model for abstractive summarization of section content. The proposed methodology has been trained and tested using benchmark data collections. The implemented system relies only on textual domain and can be further developed in order to be able to address multimodal domain including multimedia objects. It outperforms state-of-the-art summarization baselines according to several standard evaluation metrics.

Sommario

Le presentazioni sono sempre più importanti nel campo scientifico, in quanto permettono di incorporare le numerose informazioni disponibili in brevi riassunti, ma la loro creazione rappresenta solitamente un lavoro dispendioso e laborioso. Lo sviluppo della tecnologia nell'ambito del machine learning ha permesso la nascita di sistemi automatici che producono riassunti a partite da articoli scientifici in modo da aumentare la produttività umana. Sono presenti limitate ricerche nell'ambito della generazione automatica di slide in quanto affrontano tutte il problema dello scarso materiale pubblico disponibile. Questa tesi contribuisce con un nuovo dataset, APPreD, formato da coppie di articoli e corrispondenti slide raccolti dall'antologia online di ACL. In seguito, viene presentato un nuovo sistema che affronta l'obiettivo di conversione da documento a presentazione in due passi principali: i) classificazione del dataset nel formato IMRaD, composto da quattro classi di argomenti; ii) riassumere il contenuto di ogni classe attraverso il fine-tuning di un modello di machine learning pre addestrato che sfrutta un approccio astrattivo. La valutazione automatica del sistema implementato evidenzia che quest'ultimo migliora le performance dell'attuale stato dell'arte nella generazione di slide.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The creation of presentation slides from academic scientific papers is usually a work that takes plenty of time, but, nowadays, it is increasingly used by researchers to present and discuss their projects by incorporating several useful information for publication understanding. In order to be efficient, slides need to be concise but, at the same time, they need to be able to gather all the relevant information involved in the original article. The authors must first identify the main sections of which the article is composed, they need to decide how to structure the slides according to the sections and then summarize the content of the identified parts to populate the assumed presentation structure. As the text data available continues to grow on the web, manually performing these steps has become an expensive and generally laborious task that has led to the study and emergence of automated methods in order to increase human productivity.

Slide generation task can be considered a specific field of Automatic Text Summarization in Natural Language Process (NLP) area, which is gaining more and more popularity and therefore constitutes an important field of development for academic researches. Automatic Text Summarization means producing a shorter version than the original input document by preserving the most important information. It is a very challenging task, because computers do not have the knowledge and language skills of a human, which allow him to read and understand a text and then summarize it pointing out its main points. So, numerous machine learning models have been introduced for this task. Many of these models address the problem as a classification problem by deciding whether to include a sentence in the summary or not, while other models use a topic information approach. In general, the two main approaches of summarization are the extractive and abstractive methods. The Extractive summarization is in charge of identifying the most relevant sentences of input text that are linked together in order to produce a summary. The Abstractive methods are used to generate summaries that contain different words than the

original document, while preserving the key concepts and the meaning of the input content. Generating presentation slides should be part of an abstractive approach given by the usual creative nature of human slides that present an innovative summary. Despite this, most of the existing literature proposes extractive methods because of their greater simplicity and their ability to achieve more performing results. The main steps of these works are a sentence scoring stage, where an importance score is assigned to each sentence, and a sentence selection phase, which is in charge of choosing the most salient sentences while excluding redundant information. However, Abstractive Summarization methods aim at generating summaries by interpreting the text using machine learning techniques to produce a new shorter text. This requires reformulating sentences and embedding information from the full text to generate summaries as a human usually would do. Abstractive Summarization needs the ability to generate new sentences that are syntactically correct, which represents a significant difficulty for an automatic system. Natural Language Processing also faces many language challenges that only machine learning development has been able to cope with.
In Automatic Text Summarization task, machine learning techniques can be divided into two main classes: Supervised and Unsupervised. *Supervised learning* is characterized by the use of a labeled dataset. Labeled data allows you to train and supervise algorithms in data classification or prediction of results. In this way, the model can gradually measure its accuracy and learn over time. On the other hand, *Unsupervised Learning* is used to analyze and cluster unlabeled datasets. The algorithms in unsupervised environment are in charge of discovering hidden patterns in data without the need of human intervention.

This master thesis work consists of an abstractive approach method with supervised learning with the aim of generating presentation slides more similar to human ones, based on the common IMRaD (Introduction, Method, Results and Discussions) classification of input scientific articles.

1.1 Contribution

This master thesis proposes a novel task of generating slides from academic scientific papers. To tackle this task, existing abstractive summarization models are explored and analyzed in order to identify the best available solutions.
The main elements of an automatic text summarization system with supervised learning are:

- The model in charge of performing the text summarization task.
- A text extraction strategy with the aim of "reading" the source document and transforming it into a machine-readable format.

- A model learning strategy to perform slide generation.

The thesis work presents a new dataset *APPreD* (ACL Papers' Presentations Dataset) of 466 paired academic papers-presentation slides. It also proposes a sequence of automatic processing steps to extract sentences from PDF documents and slides in order to prepare the input source model.
The implemented approach involves a model learning strategy based on the classification of the document into four main topics to fine-tune a specific pre-trained model for each topic-class in parallel. The classification exploits the IMRaD subdivision that characterizes all scientific documents by assigning each article section to the class that belongs to. So, the idea consists of generating section-aware presentation slides which follow the same IMRaD structure in order to preserve the key concepts of each topic.

The abstractive nature of generated slides is given by the choice of the model that represents a novelty in this field. Existing research works rely on extractive-based approaches which produce slides by aggregating important sentences from the document, while this thesis work aims to explore abstractive summarization models world by analyzing pros and cons and choosing the most suitable model for the input document structure. The design involves the use of Longformer, a model able to manage long-documents, which consists of an enconder-only architecture in order to encode a sentence and to produce a value with no text generation. So, Longformer Encoder-Decoder (LED) version, which has never been used in the existing literature in slide generation task, is applied to produce the summaries.

In addition, the work includes the evaluation of the implemented system quantitatively, using automated metrics and the comparison of obtained performance with the current state of the art.

The main thesis contributions include:

1. Introducing a novel dataset and techniques for text extraction for both papers and slides presentation;

2. Proposing an abstractive summarization approach to summarize a document by classifying it in IMRaD topics and to produce structured slides;

3. Evaluating the proposed approach using common metrics and compare it to baselines.

1.2 Report Structure

The remaining part of this master thesis is composed of four chapters, which are described below:

- Chapter 2 shows an overview of technology used to the automatic text summarization is explained and related works for slide-generation are introduced.
- Chapter 3 presents the implemented method by describing it step by step and motivating the choices made.
- Chapter 4 is dedicated to the experiments and tests to validate the implemented method by comparing it with existing projects.
- Chapter 5 contains the final considerations and possible future developments of the thesis work.

Chapter 2

Related Works

The continuous growth of scientific documents in digital format has led to the need for automatic synthesis methods. According to [1], Text Summarization is in charge of reducing documents' content by preserving the most relevant information in order to help researchers or normal users to handle the large amount of data available. Technological progress has allowed the birth of numerous approaches in automatic text summarization across various domains in NLP tasks which allowed text summarization to quickly become an important research field. In general, existing approaches can be divided into two main categories: extractive summarization and abstractive summarization [2]. In the following sections the studies and the models used to this task are briefly explained and existing projects are introduced.

2.1 Automatic Text Summarization

As explicated in [3], automatic summaries can be generated by recognizing the most important portions of an input text and reusing them or by rephrasing relevant information possibly using words that were not in the original text (Figure 2.1).

Extractive Summarization

The first approach represents extractive methods which consist in assigning a computed score to sentences of input source and generating the output summary by choosing the most important ones; thus, they only extract sentences from original text. In [2] extractive techiniques are resumed in three main steps:

- *Intermediate Representation of relevant topics in the input text*, where the automatic system has the purpose of rearranging text in order to make easier finding the most important portions. In [4] a topic-based representation is

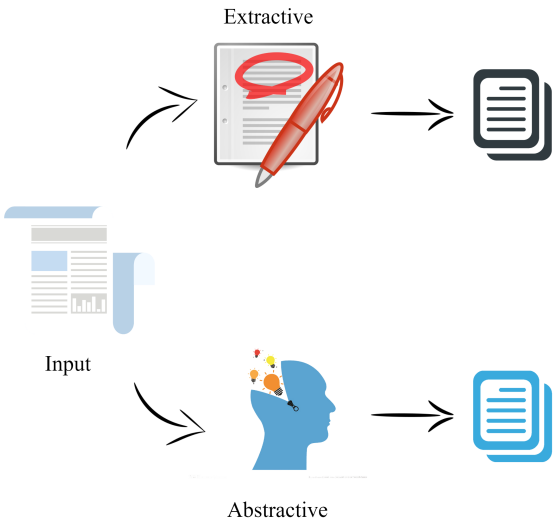

Figure 2.1: Extractive systems selects information from input source. Abstractive systems "thinks" new words.

presented where the most explicative words are identified by log-likelihood test.

- *Sentences score*, where summarizier has to assign an importance score to sentence based on its capacity to explain significant concepts. The score is computed based on two main methods: word probability and TFIDF (Term Frequency Inverse Document Frequency).

$$p(w) = \frac{f(w)}{N} \tag{2.1}$$

$$q(w) = f_d(w) * \log \frac{|D|}{f_{D(w)}} \tag{2.2}$$

Equation 2.1 shows the first method where $f(w)$ is the number of occurences of a word and N the number of all words in the input. This technique must also be able to ignore stop word list, while in [5], TFIDF is used to assign low scores to common words promoting the importance of the word rather than the frequency. It is explained in equation 2.2 where $f_d(w)$ is term frequency of a word in a document, $f_D(w)$ is the number of documents that contain the word and |D| is the number of documents in the collection.

- *Sentences selection*, where the system selects the $top - k$ most important sentences in order to generate a summary. An important method is representd

by Latent Semantic Analysis (LSA) which builds a matrix where each row corresponds to a word and each columns to a sentence, entries represent the weight of the word in the sentence. Then a singular value decomposition is applied in order to decompose the matrix into three matrices: a term-topic matrix having weights of words, a diagonal matrix where each row corresponds to the weight of a topic and a topic-sentence matrix [6].

Abstractive Summarization

On the other hand, abstractive methods produce new sentences, thus they have to "understand" the input source and generate a summary containing main concepts of document. This feature makes the abstractive summaries more similar to human ones. According to [7], approaches on abstractive summarization can be resumed in two main categories (Figure 2.2):

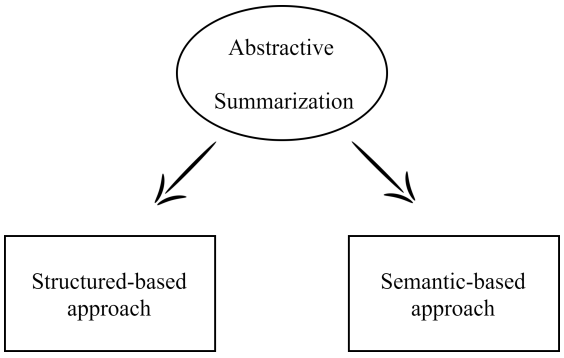

Figure 2.2: Abstractive Summarization approaches

- *Structured-based approach*, where sentences from input source are organized in a predefined structured based on specific relationships. Barzilay et al. in [8] proposes a tree based structure where text is represented by a dependency tree and first a theme intersection algorithm identifies a central theme and then a clustering algorithm sorts the sentences which are fused in order to generate summary. However, in [9] text is represented in a rule-based structure used to extract information rules that determine candidate sentences to be included in a generation pattern to produce a summary.
- *Semantic-based approach*, where documents are analyzed from a linguistic point of view by identifying semantic elements which are passed to a Natural Language Generation (NLG) system as input in order to obtain output. Gatt et al. in [10] proposes a multimodal semantic based structure where it exploits a Java library (SimpleNLG) to build a semantic model exploiting a set of lexical and phrasal types corresponding to the major grammatical categories. This

structure is passed to a lineariser which generates strings based on semantic features.

Extractive Summarization is the most explored approach at the moment, however applying abstractive summarization methods allows to generate summaries more similar to human ones which are usually not extractive.

2.2 Deep Learning for NLP

Natural Language Processing (NLP) can be seen as the union of two main fields: Natual Language Understanding (NLU) and Natural Language Generation (NLG). Before generating text, a system has to be able to capture the meaning of an input source, so the first important step in NLP task consists in making the words in a representation intelligible by a computer. In this sense, the first solution is presented in [11] where words are represented as vectors in a multi-dimensional semantic space. In order to find a way to set many words in the same semantic space, an important revolution is developed by Mikolov et al. in [12] (*Word2Vec*) in 2013 where the word embedding concept is introduced. Embedding means that word representations are learned from a neural network and it ensures low-dimensional vectors.

Word2Vec It is based on a shallow neural network which consists of only 1 or 2 hidden layers. Word2Vec embedding is obtained using two methods:

- *Continuous bag-of-words (CBOW)*, where the current word is predicted by using its surrounding context. During the process of predicting the target word, the model learns the vector representation of it.
- *Skip-gram*, where the words in surrounding context are predicted given the target word.

In both case, the model uses back-propagation to learn vector representations. Back-propagation is an efficient algorithm to find the optimal weights of a neural network in order to minimize the loss function. In CBOW model, for instance, the loss function to minimize is represented by Equation :

$$E = -\log(p(\vec{w_t}|\vec{W_t})) \tag{2.3}$$

where w_t is the target word and W_t is the sequence of context words. Back-propagation means applying the gradient descent algorithm, which implies finding out the derivatives of the loss function with respect to the weights. The architecture of *Word2Vec* consists of input, hidden and output layers.

GloVe However, Pennington et al. in [13] proposes another important embedding

architecture, called *Global Vectors (GloVe)*, which does not exploit neural networks. It is based on the hypothesis that the ratio of co-occurrence probabilities of two words with a third probe word is more significant than a direct co-occurrence probability. The idea consists of an optimization problem performing an objective function in order to minimize the dot product of two words vectors:

$$w_i^T w_k + b_i + b_k = \log(X_{ik}) \tag{2.4}$$

In equation 2.4 b_i and b_k are bias term of first word (i) and probe word (k), X_{ik} is the number of times i co-occurs with k.

In the first years of 90's, statistical Machine Learning approaches were the most used for many NLP tasks, in particular classification-based tasks were addressed by using feature-based techniques, for example classifier as Support Vector Machine. Deep learning represented an important revolution. It showed that a simple neural network outperforms previous models for almost all tasks.

RNN Recurrent Neural Network (RNN) have been proposed by Rumelhart et al. in [14] and quickly became predominant in NLP tasks because of its feedback loops allows the creation of a kind of network with "memory". Figure 2.3 shows an example of RNN cell where the feedback loop is extended during timesteps. It can be seen that the output at timestep t is computed by the combination of the current input and the output from the previous timestep $t-1$. An important

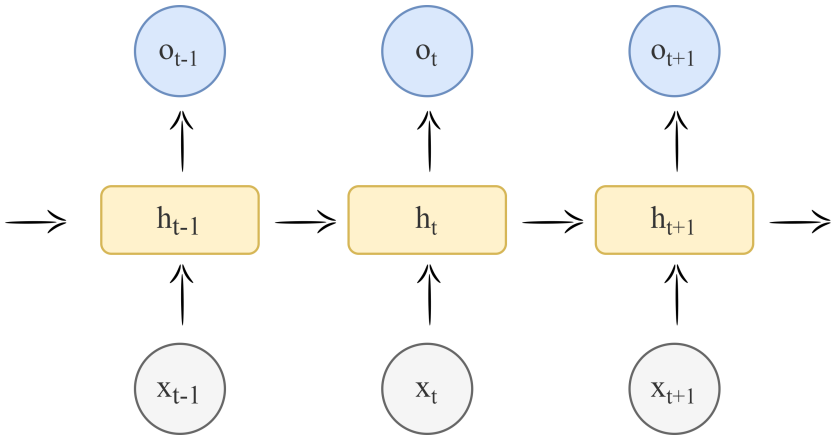

Figure 2.3: Simple Recurrent Neural Network

extension of RNN is the Bidirectional version where the text sequence is sent to the network not only from beginning to end but also from end to beginning in order to allow each step to have access to future inputs. Backpropagation is the most used algorithm for training RNNs. It computes gradients of loss function from last layer

iterating backward and so the gradient value can fastly vanish. This issue, called *vanishing gradient problem*, make it difficult to train deep neural networks limiting the memory of RNNs.

LSTM The introduction of Long Short Term Memory (LSTM), which is a variation of RNN, took deep learning approaches a step further. The architecture is similar to RNNs but it introduces the carry track which transfers cell states across timesteps. The internal structure of an LSTM cell is more complex than RNNs because it has inside three gates which decide how regulate the flow of information and which information is to be stored and which is not:

- *Forget gate*, which decides what needs to be removed from the memory. It applies an activation sigmoidal function to the current input and the previous output which returns a value between 0 and 1. This value is then multiplied with the carry track which represent the previous state. If the activation function return a value close to 0 the previous state will be "forgotten".
- *Input gate* controls how a new value should be added on memory.
- *Output gate* decides how a state should be changed in order to compute the output.

The forget gate does not involve any gradient function to the carry track, so there is at least one path where the gradient does not vanish [15].

2.3 Sequence-to-Sequence models

In the past few years, Sequence-to-Sequence models (seq2seq) have been among the most used for automatic text summarization. These models, as the name says, are built in order to convert a sequence of one type into another one. They are neural networks composed of two blocks: encoder and decoder. Encoder and Decoder are both RNNs, the first one takes as input a source article and transforms it in a vector of hidden states which is passed to the decoder and it is then used to produce a summary. Figure 2.4 shows a basic seq2seq model where encoder and decoder are bidirectional LSTM. But, as explained in [16], there are many problems with this model. Back-propagation through time does not allow a good encoder training, since the paths from encoder to the output are relatively far apart, which limits the propagation of gradient signals, as already illustraded in section 2.2. This issue causes that a seq2seq model is not able to capture long-distance dependencies between words and generates innacurate summaries and not clearly readable by humans. Furthermore, the sequential nature of this model architecture precludes parallelization, which could be critical with long sequences. In order to solve these

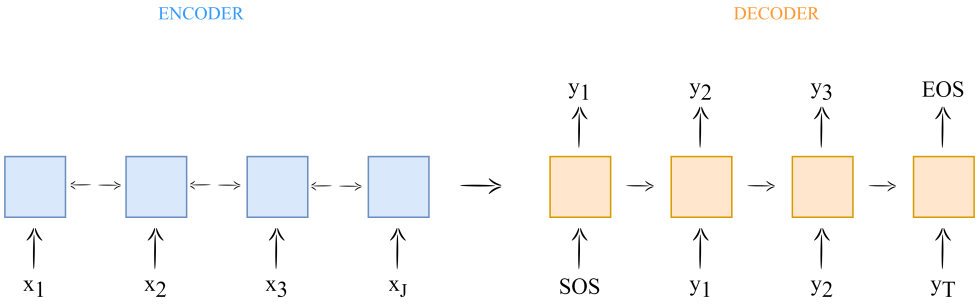

Figure 2.4: The basic seq2seq model

issues about sequence-to-sequence tasks, attention mechanism and Transformers are introduced [17].

2.4 Transformers

The Transformers, introduced by Vaswani et al. in [17], is a model architecture abandoning recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. In traditional Seq2Seq models, the decoder is initialized based solely on the final states of encoder, discarding all intermediate states, however, as the length of sequence increases it gets very difficult to produce a single vector from long sequences.
The attention-mechanism is a technique introduced to overcome this issue. For each input of encoder, the idea is to use all states to build a context vector used by the decoder to generate the output sequence. In other words, attention reads an input sequence and decides which other parts of the sequence are important by attributing different weights which are passed to the decoder with the encoded sentence. Transformers models adopt this mechanism and like the models explained above in the previous section are designed as encoder-decoder architecture, shown in Figure 2.5.

Encoder-Decoder Encoder and Decoder building blocks consist of N stacked layers that process their input iteratively. Encoder, which is on the left in Figure 2.5, is composed of two main components: a self-attention mechanism and a feed-forward neural network. The first one is for the purpose of taking inputs from the previous encoder and computing its weight, i.e. the significance in the sequence, to generate output encodings. While, the feed-forward neural network processes the outputs one by one and then they are passed to the next encoder or to the decoders as inputs. The first Encoder in the stack takes positional encoding and embedding of the input sequence in order to take the order of the sequence into account for

a good result. Decoder, on the right in the figure, has a structure similar to the encoder but has an additional attention mechanism which aims to capture relevant information from the encodings.

Scaled dot-product attention As described in [17] the attention mechanism can be explained as mapping a query and a set of key-value pairs to an output where queries, keys and values are all vectors.

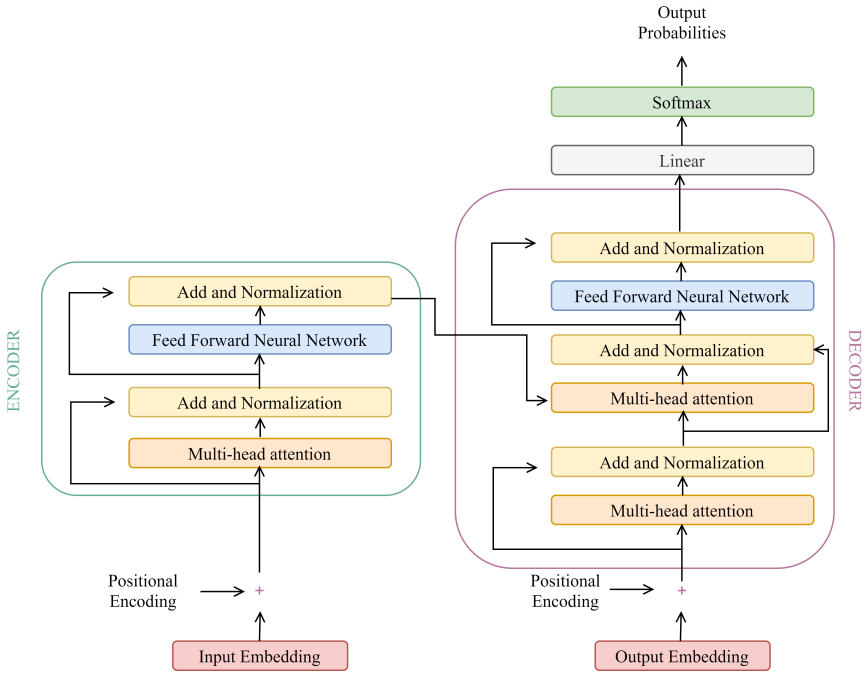

Figure 2.5: Transformer architecture

$$Attention(Q, K, V) = softmax(\frac{QK^t}{\sqrt{d_K}})V \tag{2.5}$$

$$a = softmax(\frac{QK^t}{\sqrt{d_K}}) \tag{2.6}$$

In other words, attention weights are computed by the dot product of queries and keys and normalized by a softmax function (2.5).
Q represents the query and is the vector representation of one word, K consists in the vector representation of all the words in the sequence, while V represents again the vector representation of all words but summed and multiplied with a specific attention weight (2.6) which defines how each word is influenced by all the other words in the sequence [18].

2.5 Existing Transformer-based Models

Unlike RNNs, Transformers attention-mechanism computes the context for each word of the sentence, thus transformers don't need to process data in order. This feature allows for more parallelization in training phases and this led to the emergence of many models pretrained with very large language datasets. Some of them use an encoder-only architecture, so they aim to encode a sentence and to yield a specific value with no generation of an another sequence. However, some others also add the decoder block to this architecture in order to produce a new sequence from the input source. The next two sections resumes the most used models in the state-of-the-art for both tasks.

2.5.1 Encoder-only Transformers

BERT Bidirectional Encoder Representations from Transformers (BERT) is a standard Transformer model with a number of encoder layers and self-attention heads. It is designed to understand the meaning of corrupted language in text by using surrounding text to establish context, thus it just needs to encode the language representations so that it could be used for other tasks. For this reason it consists only of encoder parts. It is pre-trained on two different NLP tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM training is intended to mask some percentage of the input tokens at random, and then predict those masked tokens. However, NSP training aims to capture the relationship between two sentences [19]. BERT revolutionized the NLP world because of its solution for making Transformers bidirectional. Thanks to MLM, instead of predicting the next token as in the common next-word prediction objective, the model is expected to predict a masked token and this means the model is affected by both sides of the token in order to guess it.

BigBird Self-attention mechanism overcomes limits of RNNs and its sequential nature to encourage more parallelization allowing the growth of dataset but sequence length remains still a problem. In thise sense, BigBird, presented in [20], proposes a new attention mechanism where the complexity is linear in the number of tokens and not anymore quadratic as in BERT. Encoder layers of BigBird use a *generalised attention mechanism* which is easily represented by a graph whose vertex set is tokens of input sequence and edges form the set of dot products done by attention mechanism. In the case of the graph is fully connected performances are the same of BERT with its full self attention. To avoid this they introduce three windows of nodes that each token has to attend:

- *Local window*, which is based on the concept of locality of reference according to which most of the information about a token can be extracted from its

neighbouring tokens.

- *Random window*, a random number of keys which query has to attend.
- *Global window*, they define a set of global tokens that attend to all tokens in the sequence and to whom all tokens attend to.

In this way, BigBird achieves the benefits of full self-attention with a smaller number of operations.

Longformer Similarly to BigBird, Beltagy et al. in [21] propose an attention mechanism to avoid memory and computational requirements limits due to self-attention quadratically growth. Longformer use a similar local+global attention of BigBird with some differences. To capture information of local context they use the same concept of local window but, in order to improve the receptive field, it is dilated by a fixed value. This dilated window attention is not flexible enough to learn context of each token. Accordingly, they also add global attention on few pre-selected input locations. In contrast of BigBird, Longformer global attention window can change according to the downstream task the model is needed.

2.5.2 Seq2Seq Transformers

BART In [22] BART is presented as the extension of BERT for generation tasks. It is a denoising autoencoder that maps a corrupted document to the original document it was derived from. So it is implemented as a standard Transformer architecture with a bidirectional encoder, as BERT, and a left-to-right autoregressive decoder, which is the component in charge of generating the output sequence. It is considered an innovation because unlike existing denoising autoencoders, which are suited to specific noising schemes, BART allows to apply any type of document corruption. BART is trained by corrupting documents and then optimizing a reconstruction loss, the cross-entropy between the decoder's output and the original document. For summarization tasks, it evaluates using ROUGE metrics and uses two summarization datasets, CNN/DailyMail [23] and XSum [24]. For fine-tuning, an uncorrupted document is input to both the decoder and encoder, and it uses representations from the final hidden state of the decoder.

PEGASUS It proposes a new self-supervised pre-training objective for abstractive summarization, gap-sentence generation, and studies strategies for selecting those sentences. In PEGASUS pre-training task, important sentences are removedmasked from an input document and are generated together as one output sequence from the remaining sentences, this process is more similar to an extractive summary rather than in BART. It is interesting because it shows how good abstractive

summarization performance can be achieved across broad domains with very little supervision by fine-tuning the PEGASUS model. The base architecture is based on standard Transformer encoder-decoder but, in contrast to BART, which masks small continuous text spans, PEGASUS does it with multiple whole sentences [25]. For pre-training it considered two large text corpora: C4 - consists of text from 350M Web-pages, and HugeNews - a dataset of 1.5B articles.

T5 Text-to-Text-Transfer-Transformer (T5) proposes to treat every text processing problem as a text-to-text problem, taking text as input and producing new text as output. It is interesting because this approach allows to use the same model, loss function, hyperparameters, across diverse set of tasks. The model is equivalent to the original encoder-decoder Transformer (BERT's architecture) with the exception of removing the Layer Norm bias [26], placing the layer normalization outside the residual path, and using a different position embedding scheme (T5 uses relative scalar embeddings). It is trained using teacher forcing. This means that for training it always need an input sequence and a target sequence. The T5 model, pre-trained on C4, achieves state-of-the-art results on many NLP benchmarks while being flexible enough to be fine-tuned to a variety of important downstream tasks [27].

2.5.3 LED

In order to facilitate modeling long sequences for seq2seq learning, in [21] is proposed a Longformer variant consisting of both encoder and decoder as in original Transformer architecture. LED encoder uses local attention with window size 1,024 tokens and global attention in the first token, which means that it attends to all the tokens in the sequence and all tokens attend to it. Due to small number of global tokens the complexity of local+global attention remains *O(n)*, where n is the number of tokens. However, LED Decoder uses full attention to the entire encoder and to the previously decoded locations. LED is initialized with BART parameters and follows its architecture in terms of number of layers and hidden size. It is trained using arxiv summarization dataset [28] and two version are released LED-large-16384 and LED-base-16384.

Training

In particular, a word-level training is applied to LED which, as explained in [29], consists of trying to optimize the prediction of next token. In other words, from an input article x a seq2seq model generates a summary y with the probability $P_\theta(y|x)$, where θ represents model parameters, like weights and bias, that are initialized

from BART's checkpoint.

$$P_\theta(y|x) = \prod_{t=1}^{T} P_\theta(y_t|y_{<t}, x) \tag{2.7}$$

Equation 2.7 expands the probability and each multiplication factor represents the *likelihood* which is a conditional probability of the next token y_t given all previous ones denoted by $y_{<t} = (y1, y2, ..., y_{t-1})$.
The algorithm used to train LED model, known as *teacher forcing* or *Cross-Entropy Training (XENT)*, aims to learn θ maximizing the log-likelihood of observed sequences (ground-truth) $\hat{y}_t$:

$$\log P_\theta = \sum_{t=1}^{T} P_\theta(\hat{y}_t|\hat{y}_{<t}, x) \tag{2.8}$$

As shown in Figure 2.6, during training the algorithm uses observed tokens (ground-truth) as input and aims to improve the probability of the next observed token at each decoding step. Starting with a special token SOS, which stands for start-of-sequence, the model generates a token at time t with a certain probability. Then, the generated token is fed into the next decoding steps and the generation ends when the output is EOS token (end-of-sequence). The generation of a token has

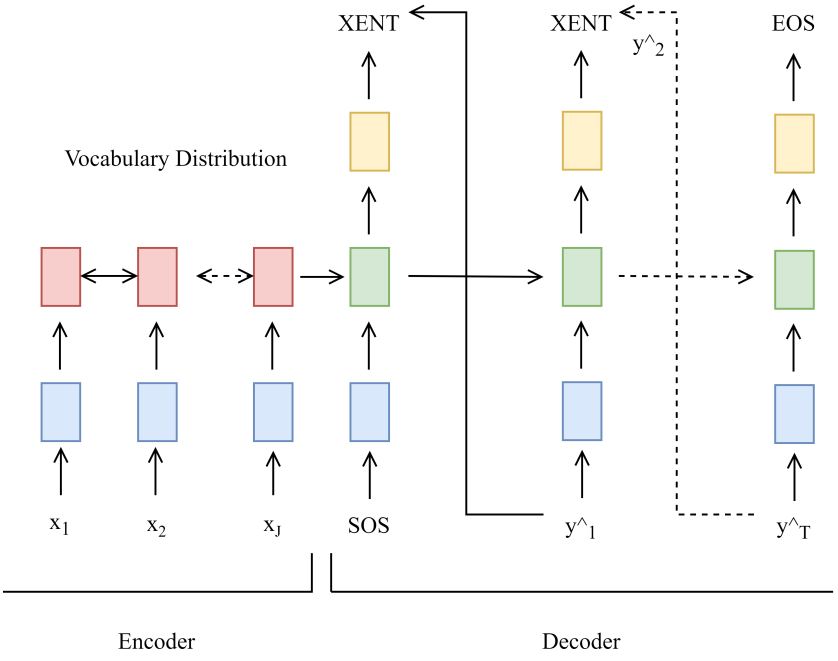

Figure 2.6: Training with teacher-forcing algorithm

the purpose of maximizing likelihood by finding the optimal sequence between all possible sequences. In LED training, *beam search* algorithm is applied to approximate the exact inference. Beam search algorithm is a graph search for a given sequence of probabilities and beam width user-selected parameter B. At each step, each candidate sequence is expanded with all possible next steps. Each candidate step is scored by multiplying the probabilities together. The B sequences with the most likely probabilities are selected and all other candidates are pruned. The process then repeats until the end of the sequence [30].

2.6 State-of-the-art in Slide Generation

Over the past few years automatic generation of presentation slides for academic papers has increasingly become a very useful task for researchers. This section briefly describes in chronological order the principal existing works which represented the state-of-the-art about slide generation from scientific paper in the last years. They take advantage of existing models, described in section 2.5, to generate summaries in slides form. Most of them use an extractive approach to generate summary, while the more recently works use a full abstractive model and for this reason could be used as competitor in this master thesis work.

PPsGen The first proposed system for slide generation is presented by Hu et al. in [31], whose work introduces a new method of automatically generating presentation slides for academic papers. PPsGen is not an abstractive method because its output is based on the selection of the most important sentences in an academic paper. The architecture leverages on two principal models. The first one is in charge of assigning an importance score for each sentence in the given paper and then a second model has the purpose of selecting and aligning key sentences to create output slides. For the first step, the importance of each sentence is learned by using the Support Vector Regression (SVR) model, which is a supervised learning algorithm used to predict discrete values and so very useful for classification tasks. In order to construct training data they apply a *similarity scoring* method based on a set of features which represent each sentence (similarity with titles, word overlap with titles, sentence position, number of noun phrases and verb phrases, stop word percentage). However, for the selection of most important sentences PPsGen relies on the Integer Linear Programming (ILP) which aims to optimize a defined objective function to maximize the overall importance score and the coverage of selected sentences. It uses 1200 paper-slide pairs obtained by crawling Arnetminer (`http://arnetminer.org`).

Afterwards, a similar PPsGen work is published in [32] where a phrase-based

approach to generate presentation slides for academic papers is introduced. It uses 175 pairs of paper and slides in the computer science field. The main difference with previous work is that they consider phrases as the basic element for content selection rather than sentences as usually. This allows to generate more concise and easy to read slides with no long sequences. Otherwise the approach is similar to [31] based on the following steps:

- Extraction of candidate phrases from each section of the given paper.
- Learns both saliency score of each phrase and the hierarchical relationship between a pair of phrases.
- Selection and alignment of salient phrases.

For the step of saliency estimation they apply a Random Forest classifier which is in charge of choosing candidate phrases for extraction relying on a specific set of features, like in PPsGen. Finally, two greedy algorithms are used to determine which candidate phrases are selected and used in output slides. The algorithms aim to satisfy two objective:

- Select as many phrases with high salience as possible.
- Align as many pairs of phrases with strong hierarchical relationships as possible.

In 2019 Sefid et al. proposed another work in [33] which describes always an extractive approach for generating presentation slides for scientific papers. The dataset contains conference proceedings in computer science. As the first two projects, the architecture consists of 2 steps. The first is to calculate scores used to identify important sentences. In contrast to previous works, it is interesting because use Deep Neural models (CNN, RNN) to encode sentences and its context as new features in sentence ranking to produce slides. For sentence selection, they compare a greedy method and the ILP. The second is to extract salient sentences under constraints. This task is done by using the Bullet Point Generation Algorithm which arranges slides in levels of bullet points.

DOC2PPT The first interesting work which proposes an abstractive approach of generating a slide deck from a document is presented in [34] and introduces a novel task of creating presentation slides from a multimodal document with text and figures. It uses a dataset about 6K paired documents and slide decks. It is interesting because it solves several challenges in the vision and language domain, e.g., visual-semantic embedding and multimodal summarization.The architecture leverages on a hierarchical recurrent sequence-to-sequence architecture. DOC2PPT network is made up of four main modules that read a document by dividing it into sections containing sentences and figures containing images and then produce a set of slides in a hierarchically way:

- *Document Reader (DR):* It encodes sentences and figures in a document, using a version of BERT, section 2.5.1, and projects them to a shared embedding space using a deep learning model, called MLP, which is a specific class of feedforward artificial neural network.
- *Progress Tracker (PT):* It is formed by some pointers that indicate which section is currently being processed and consequently which slide is currently being generated.
- *Object Placer (OP):* It is in charge of deciding which object, between sentence and figure, to put on the generating slide.
- *Paraphraser (PAR):* It is the component that summarizes the selected sentences in a concise form. It is implemented as an attention based Seq2Seq model with the add of copy mechanism, which consists in generating target tokens by directly copying them from inpute sequences based on their attention weights [35].

The DOC2PPT main goal is to propose a new way to do Multimodal Summarization, encoding different types of input in the same visual-semantic embedding space. All of which makes it out of the scope of this thesis work and so it can not be used as a real competitor.

D2S However, the project that will be used as competitor in the following sections is presented in [36] by Sun et al. because this paper proposes a new variant on document-to-slide generation process in an abstractive text summarization approach. It is interesting because in addition to the contribution of the novel model architecture, it also constributes a high-quality dataset (SciDuet), which contains 1,088 papers and 10,034 slides. It considers the task as a Query-Based Single-Document Text Summarization. D2S is built as an interactive model where in the first step users have to input a short text as the slide title and it is used by a Dense Vector IR module to select the most appropriate sentences from the corresponding paper. In a second time, a QA model is used to generate the original summary of the extracted text. The architecture is illustrated in Figure 2.7 and consist of the following three modules:

- *Keyword Module:* This block is in charge of building relationship trees between section titles and subsection headings in order to facilitate next steps.
- *Dense IR Module:* The idea here is to rank paper sentences in terms of relevance to a given slide title. They design a weighted ranking function with vector representations of titles, passage texts, and the leaf node keywords of a dense vector IR system based on a distilled BERT miniature. The top ten candidates are choosed as input's context to the next module.

- *QA Module:* This module uses slide titles and keywords as query and the concatenation of the output of IR module as the context. The QA module is a fine tuned BART, section 2.5.2, where the query and the context are encoded in the format of "*title[SEP1]keywords[SEP2]context.*"

It evaluates using IDF-recall, which computes the proportion of words in the original slide text in the retrieved context weighted by their inverse document frequency. For slide text content generation, it uses ROUGE metrics and human evaluation. A distilled uncased Bert miniature with 8-layers, 768 hidden units, and 12 attention heads was trained and used to perform IR. QA model was fine-tuned over BART-Large-CNN.

The D2S system is used in the following sections to compare the obtained results in this thesis work.

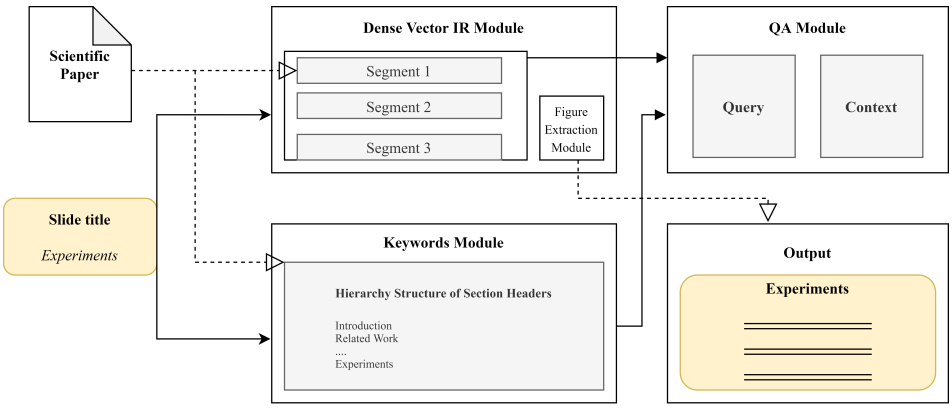

Figure 2.7: D2S Architecture [36]

At last, there are also works that propose approaches about slide generation from academic papers in unsupervised environment, thus with the absence of training data.

The first one introduces a topic-aware paper to slide generation approach based on sentence selection. It uses ACL Anthology Reference Corpus as unlabeled corpus of papers for unsupervised learning. It has gained popularity because it adapt mutual learning in the unsupervised setting, where it provides a flexible framework for integrating prior knowledge to initiate the training. It evaluates using three metrics: 1) accuracy; 2) sentence-level classification precision/recall; 3) BLEU metrics. The architecture consists of 2 models: Neural Sentence Selection Model which captures sentence semantics and Log-Linear Classifier where the prior knowledge is encoded as features. The approach is based on mutual learning with two models updating

collaboratively [37].

Recently work is published in [38] where previous tasks encountered, like sentence importance evaluation, sentence selection and generation, are addressed by applying unsupervised methods.

Chapter 3

Implemented methods

The final goal of this thesis work is to generate a set of slides from academic papers following the standard format of academic journals, called IMRAD, with an abstractive approach in supervised environment. Figure 3.1 shows an overview of the workflow, the task involves reading a document by extracting sentences of each sections and its titles, then classifying sections in Imrad topics and summarizing each topic in order to create new concise sentences that cover most important information of input document. After a first phase of study and evaluation of existing models, illustrated in 2.5, LED was chosen to summarize input source due to its capacity to manage long sequences and to outperforms state-of-the-art summarization baselines. The workflow of implemented method can be summarized in three main steps:

- The Crawling and Parsing dataset phase, where the dataset is collected and then sections with sentences are extracted from PDFs and rearranged into json files.
- The Classification phase, where sections are assigned to macro-topics (introduction, method, result and conclusion) in order to obtain a common organizational structure to future generated slides.
- The Generation phase, where sequence-to-sequence model is in charge of producing a summary for each topic.

The following sections will introduce in details each of these steps and the approach used to implement it.

3.1 Crawling and Dataset Construction

This master thesis work proposes a new dataset composed of scientific papers and their corresponding slide presentations. The Dataset is crawled from ACL online

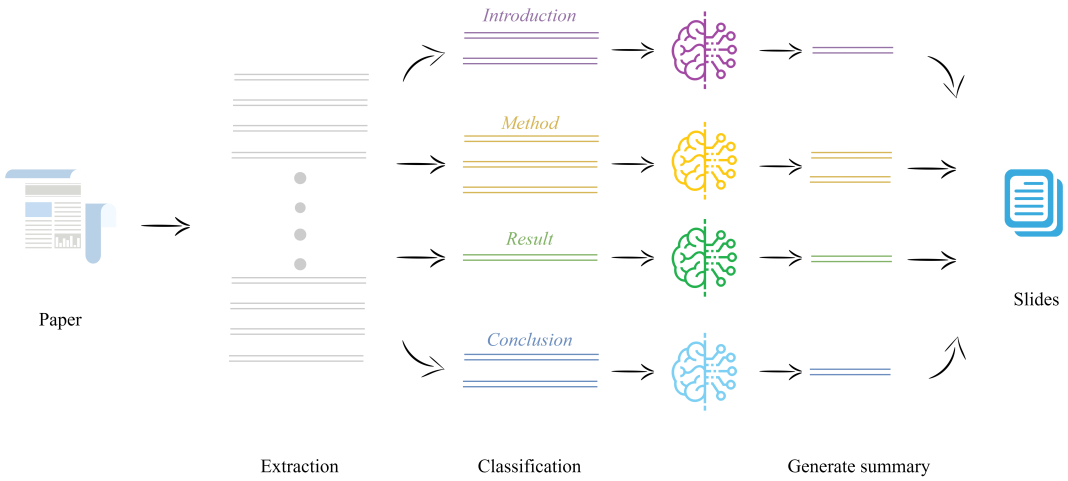

Figure 3.1: General overview of implemented method workflow

anthology (`https://aclanthology.org/`) and contains pairs of papers and slides from recent years' computer linguistic and machine learning venues.
The dataset consists of 466 paper-slides pairs splitted in the Train-Dev-Test format. Table 3.1 and Table 3.2 show some dataset statistics. In particular, *SC-len* parameter shows the average token length in the dataset that helps the model to define output summary length.

	#papers	#slides	SC-len
train	401	6,952	138.61
dev	20	78	71.86
test	45	717	76.25

Table 3.1: Dataset statistics: 'SC-len' represents the average token length for slide contents

3.1.1 Dataset Parsing

The first task of the proposed model aims to read a PDF document in order to convert it into a sequence of sentences which can be sent to a model as input. According to supervised nature of work and consequentially to the necessity to have a ground-truth to apply in the fine-tuning phase, the parsing stage is applied to both papers and its corresponding slides.

	SC-len
Introduction	69.0
Method	368.32
Result	77.59
Conclusion	76.87

Table 3.2: The average token length for slide contents for each IMRaD topic

Papers

For parsing papers, PDF documents were converted in a txt format through GROBID [39], which is a machine learning library in charge of re-structuring raw documents by transforming them into machine-friendly, structured, and predictable representations. GROBID exploits a cascade of sequence labeling models for parsing a document. Figure 3.2 shows how it works, where increasingly specific models are used to identify the areas of the paper. The first one, for instance, *Segmentation model* is in charge of detecting main sections of document,e.g. the title page, the header, the body, the head and foot notes, the bibliographical sections. The models at the lower level, in turn, identify the sections of detected areas. This approach allows to produce a very detailed result preserving original document layout and not compromising its sectional division. Afterwards, the GROBID output is again

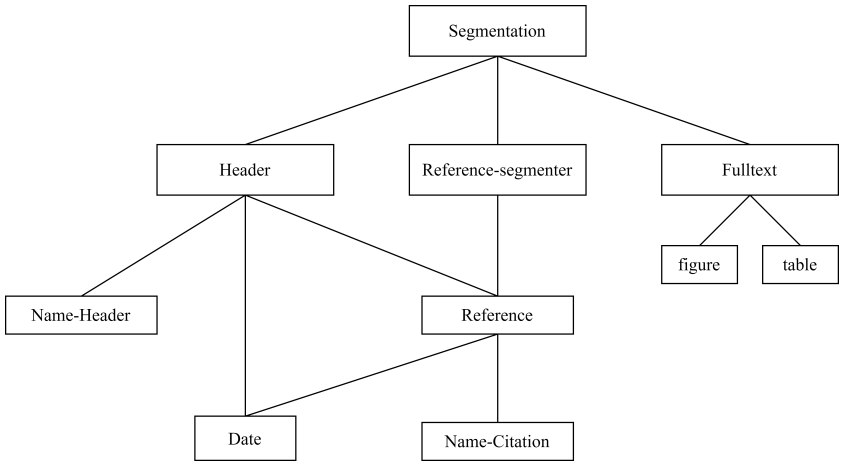

Figure 3.2: GROBID cascade of sequence labeling models [39]

analyzed to build a more comprehensive format, so a json file is created where each paper of dataset is represented by a dictionary consisting of the keys illustrated in Figure 3.3.

This dataset design facilitates the following paper sections classification and

```
{
    "title": ....,
    "abstract: ....,
    "sections": [
        { "heading": ...., "text": .... },
        { "heading": ...., "text": .... },
        .... ],
}
```

Figure 3.3: Dataset in json format after parsing stage

makes the document accessible to the summarization model.

Slides

In order to build a ground-truth to use during model fine-tuning, original slides have to be parsed and rearranged. Due their less structured nature than papers, GROBID can not be applied because it is not able to recognize a common design and extract correct text. Human slides are full of images and tables surrounded by text without a specific order and for these reasons extracting meaningful text is very difficult. In this work, the used approach consists of applying python library *pdfplumber*[1], a fork of the original *PDFMiner*[2], which scans a PDF to retrieve detailed information about each text character. In particular, for each slide of deck, *extract_words* method from pdfplumber class is applied, which retrieves each individually recognized word as a dictionary along with some extra attributes that help to rebuilt sensible text:

- Fontname
- Size

[1] `https://pypi.org/project/pdfplumber/`

[2] `https://pypi.org/project/pdfminer/`

- Doctop, that is the distance of top of character from top of document.

These three features allow to analyze for each word its position and style in order to compose sentences with words on the same line, thus if the difference between doctop feature of two words is not greater than a chosen *y_tolerance* they are considered to belong to the same sentence and therefore linked together. However, fontname and size along with position are used to extract slide titles in order to rebuild slide in the same json style like for papers, as already illustrated above.

Data Cleaning

Once text is extracted and rearranged from papers and corresponding slides, another important step is a pre-processing phase in order to have available data as comprehensible as possible. Usually, a scientific paper is full of references or notes

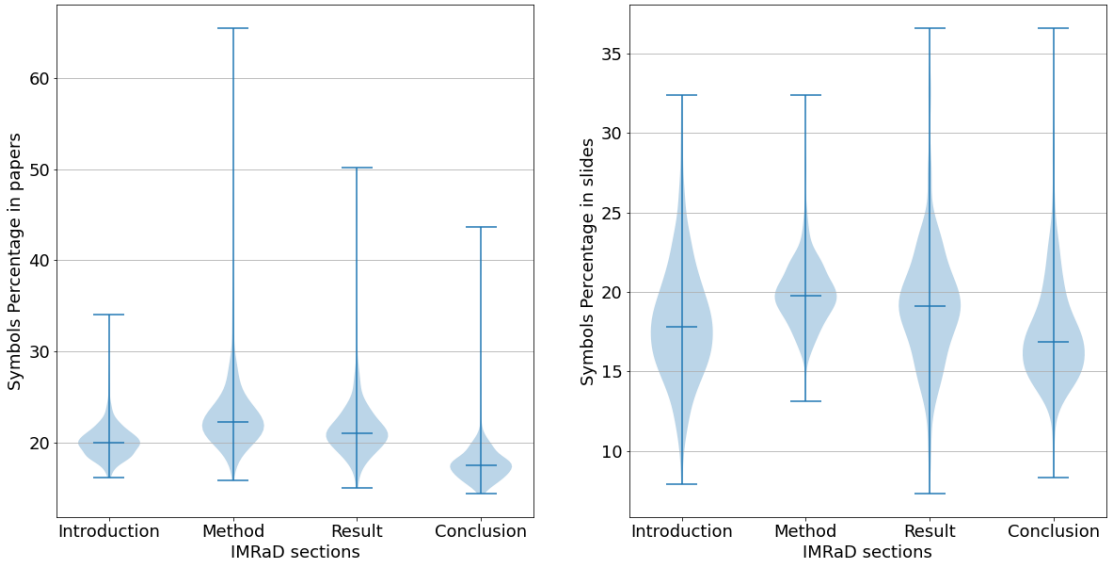

Figure 3.4: The two plots shows the trend of the percentage of symbols within sections of a paper (on the left) and its corresponding slides (on the right).

which are not considered sensible parts of text that will be summarized. Moreover, the presence of particular mathematical formula could affect text comprehension because of its symbols which can contain non-ascii characters that later lead to generate unicode error. Even more in slides, extracted text could include sentences hard to understand because they can belong to caption images or could be parts of a table. In order to try to overcome these issues, the average of the percentage of symbols in sentences is computed for each section of papers and slides, Figure 3.4 shows the symbols percentage of each paper and corresponding slides in IMRaD

sections. In papers, Method class contains more symbols because it describes techniques and procedure of the purpose of the study, while, in presentation slides, Result and Conclusion reach high values because statistical results (tables and figures) are reported. The trend has been analyzed for each section and sentences that has a percentage of symbols greater than a selected threshold have been pruned from dataset text.

3.2 IMRaD Classification

The second step of thesis work consists of assigning extracted sections of papers and slides to four main topics in order to create a concise and professional format for generated slides which covers all relevant information of documents. The four topics,

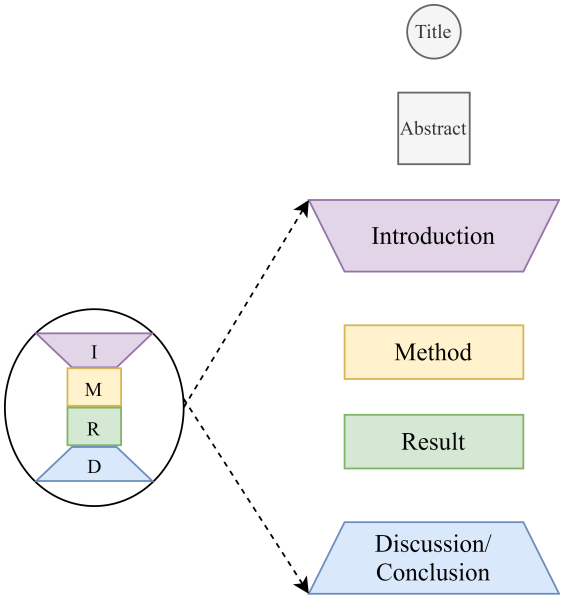

Figure 3.5: Imrad Structure

illustrated in Figure 3.5, establish the most common format used in publication, called "IMRaD".

IMRaD stands for Introduction, Materials and Methods, Results, and Discussion/Conclusion and according to Wu in [40] it began to be adopted by scientific journals around the 1940s, and quickly became the dominant format for research papers in a majority of leading scientific journals. IMRaD includes:

- **Introduction**, where relevant background information and descriptions of the purpose of the study are provided.

- **Materials and Methods**, which describes the work procedure and techniques. It is the most detailed section.
- **Result**, where study results and their statistical importance are described.
- **Discussion or Conclusion**, which explains the meaning of study and if objectives are achieved.

An important element of scientific papers is the Abstract, which does not belong to IMRaD format but contains a condensed version of it. Organizing document sections following IMRaD structure has a wide range of advantages because it facilitates reading and knowledge acquisition not only for humans but also for a learning model that can obtain better performances. Moreover, generating slides individually for each topic ensures a full coverage of input source information.
So, the idea here is to rearrange the extracted dataset in a new dictionary made up of the four IMRaD topics. A classification script is applied to each section in order to assign its sentences to the topic it belongs to. So for each topic the classification is implemented by using helpers containing keywords that refer to that topic and then the section is assigned according to its title.

```
TITLE_HELPER = ["title"]
ABSTRACT_HELPER = ["abstract"]
INTRODUCTION_HELPER = ["intro", "motivation"]
LITERATURE_HELPER = ["related", "literature", "review", "background"]
METHOD_HELPER = []
EXPERIMENTS_HELPER = ["exp", "eval", "result", "statistic"]
CONCLUSION_HELPER = ["concl", "discuss", "future"]
```

If section title is composed of one of the helper words then its sentences are assigned to the topic to which the helper refers. If section is found to belong to *Literature* or *Experiments* its sentences are pruned from dataset because they are not included in IMRaD structure, while if section does not match any of the helpers then the sentences are assigned to *Method* topic.

3.3 Slide Generation

The idea in this thesis work is to use classified dataset in order to fine-tune four sequence-to-sequence models one for each IMRaD topic. Due to the long sequences nature of scientific papers, as explained in 2.4, Transformers is currently the best choice to manage them and produce sensible summaries. In particular, Longformer allows to overcome memory and computational limits because of its

modified architecture which employs a linearly scalable self-attention mechanism. These considerations led to the choice of Longformer-Encoder-Decoder (LED) as sequence-to-sequence model to generate slides.

3.3.1 IMRaD LED Fine-tuning

The Longformer model, explained in section 2.5.1, has been extended in a generative model with an Encoder-Decoder architecture in order to support sequence-to-sequence tasks. LED Encoder uses the attention pattern of Longformer, local+global attention, while the decoder applies full self attention mechanism to encoded tokens. It is initialized following BART's architecture and parameters but position embedding is extended from 1K tokens to 16K in order to be able to proccess longer inputs. Training steps, described in section 2.5.3, produced two pre-trained LED version, base and large, that can be fine-tuned in order to make small adjustments to achieve desired output. Because of limits of available computational resources, in this thesis work the LED-base version is used to be fine-tuned as checkpoint.
The proposed idea for fine-tuning is to setup four LED models one for each IMRaD class in order to create more specialized models that capture relevant information from all topics of an input document and generates presentation slides that follows IMRaD structure. Using pre-trained models with similar dataset for fine-tuning is a time-efficient process because a huge amount of data is imported from previous models and the new dataset makes the new model much more reliable. As shown in [41], large-scale model first pre-trained with massive text-corpora with self-supervised objectives and then fine-tuned on downstream tasks achieved state-of-the-art performance on a variety of summarization benchmark dataset. As in training phase, fine-tuning goal is to minimize the cross-entropy loss, equation 3.1, where each example (x, y) represents a pair of document_imrad_section and presentation_imrad_section, T is the target sequence length and p is the model's predicted probability for the next word [42].

$$\mathcal{L}_{Data} = -\sum_{t=1}^{T} \log p(y_{t+1}|y_{1:t}, x) \tag{3.1}$$

Figure 3.6 shows an overview of the proposed fine-tune where dataset built as illustrated in section 3.2 is applied in order to learn the four different LED models.

Tokenization

Before the model feeds on the extracted text, a pre-processing phase is required in order to encode sentences in vector representations. According to [43], the first step in all NLP tasks is to recognize the basic units of input text which must not

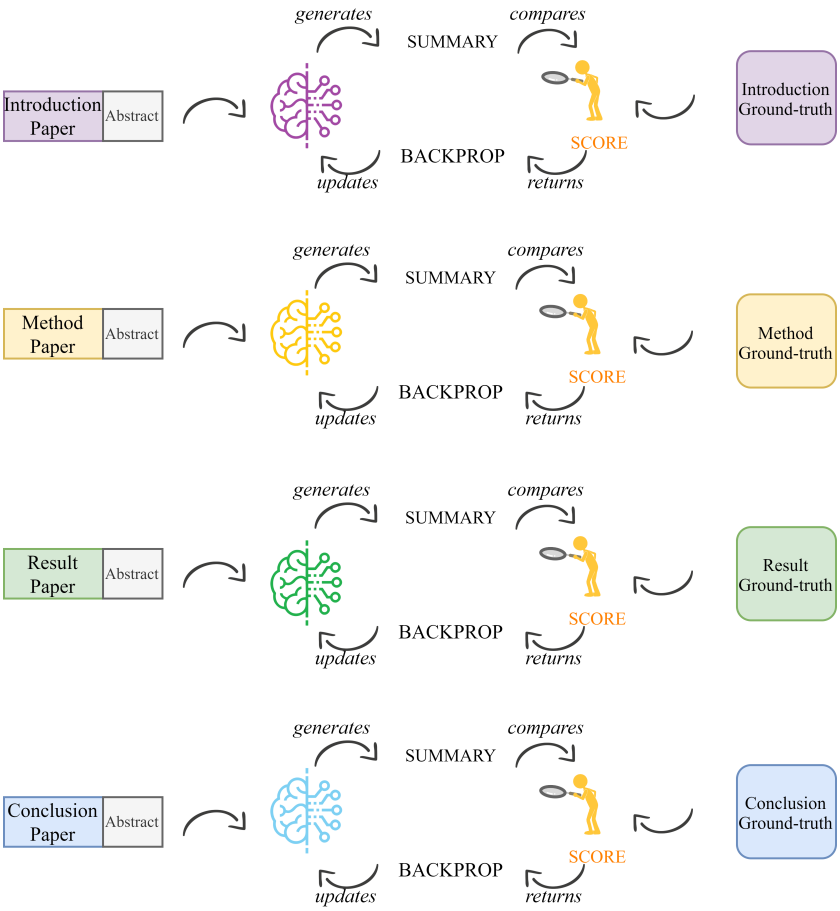

Figure 3.6: Fine-tuning illustration

be decomposed during following stages. Tokenization represents the process to identify words or subwords which are to be considered as basic units, thus it is in charge of splitting input text and converting units to vectors of input ids through a look-up table. Tokenization can be splitted in three different approaches:

- *Character-based*, where tokenizer splits text into single characters in order to avoid out-of-vocabulary words issue because it can create the representation of unknown words from individual character representation. On the other hand, this approach causes a significant increase in the size of the vocabulary.
- *Word-based*, which is the most common used approach. It consists of splitting text into words based on a specific delimiter, usually space. This tokenization represents similar words with different IDs which is an important limit.
- *Subword-based*, where rare words are splitted in subwords but frequently used

words are no decomposed by tokenizer. This allows to reduce vocabulary size and avoid different meaning to similar words issues.

LED Tokenizer uses a subword-based approach, in particular, adapts Byte-Pair Encoding (BPE) algorithm, introduced in [44], where frequent words are merged into a single symbol in order to represent the entire dataset with the least amount of tokens. In pseudo-code Algorithm 1 an example of BPE algorithm is explained.

Algorithm 1 BPE Tokenization pseudo-code

Input: string C, number of merges k **Output:** Vocabulary V

Initialize V with all unique characters in C
for $i = 1, k$ **do**
    Choose most frequent pair of adjacent tokens in C (t_{left}, t_{right});
    Create new tokens by concatenating it $t_{NEW} = (t_{left}, t_{right})$;
    Update vocabulary V;
    Replace each occurence of (t_{left}, t_{right}) with t_{NEW} in C;
end for

Encoding

Tokenizer has then to represent basic units in numerical indices that will be used as input by the model. In this sense, LED Tokenizer provides encoding methods that return input ids of a sequence or pair of sequences. The idea, as illustrated in 3.6, is to append to each IMRaD topic the Abstract by exploiting *encode_plus* method provided by PreTrainedTokenizer class which allows to encode a pair of sequences together. The Abstract represents a condensed version of all IMRaD topics, so the hypothesis is that integrating it with each single topic can help model to pay attention towards important information across all topics when generating summary. IMRaD topic and Abstract are encoded in the format of *"<s>Topic<\s><\s>Abstract<s>"* where *\s* is the separator token used to build a sequence from multiple sequences. Encoding method provides, also, local attention mask for each input id to which global attention have to be added that defines which token are attended globally and which not. According to [21], for summarization tasks, global attention mask is set only for the first token in the sequence.

During fine-tuning, an automatic scores computes the metrics from predictions compared to ground-truth. Back-propagation is used in order to exploit the metrics to updates model parameters improving following predictions.

The proposed model is fine-tuned on crawled dataset with a batch size of 2 and with half-precision floating-point format.

The maximum input token length was set to 8192 and according to Table 3.2,min and max output token lengths are set to 100 and 512 for Method topic and to 50 and 128 for the other topics.

3.4 Implementation tools

The main programming language used in methods implementation is *python*. This language allows the development of deep learning models by providing several useful libraries. In particular, during the project implementation, the following libraries and tools have been used:

- *GROBID* [39], machine learning library used to extract text from PDF papers.
- *PdfPlumber* [45], python library used to extract text from PDF slides.
- *Transformers* and *datasets* by Huggingface [46], open-source community which provides Transformers library. It allows to use all available models with a set of pre-trained weights. Huggingface provides Datasets library in order to access and share datasets for NLP tasks.
- *Pytorch* [47], is the principal library used to implement the model. It is an optimized tensor library for deep learning using GPUs and CPUs.
- *Numpy* [48], is an open-source project for scientific computing in Python. It is used to manage inputs as vectors.
- *spaCy* [49], is a free, open-source library for NLP in Python. It is used to recognize and split sentences from extracted text.
- *MatPlotlib* [50], is a python library used to support the visualization of charts.

Chapter 4

Experimental Results

Only some of the existing projects concerning slides generation, introduced in chapter 2, have publicly released their own dataset and code to be used as a comparison during the evaluation phase. In particular, *DOC2PPT* [34] and *D2S* [36] represent the works most similar to the implemented model. *DOC2PPT* proposes a multi-modal summarization for the purpose of resolving several tasks in vision and language domain. For this reason, it is considered out-of-the-scope of this thesis work which relies only on textual domain. In order to compare the results obtained by the implemented method with state-of-the-art, *D2S* results have been considered as competitors.
D2S has released its own dataset (*SciDuet*) and code. However, due to computational resource limits and errors in the released code, *D2S* model was unable to be fine-tuned and tested on proposed dataset.

So, the implemented model has been tested with both datasets, proposed one and *SciDuet*, and the results for *SciDuet* dataset have been compared with *D2S* published results. In addition, two baseline models, *BART* [22] and *PEGASUS* [25], have been also fine-tuned and tested with both datasets.

4.1 Computational resources

The implemented procedure have been developed by using *Colaboratory* by Google[1] which allows to execute Python code in the browser. The virtual machines made available in Google Colab host a configured environment where numerous Python libraries are present. The model fine-tuning and the tests presented in the next sections have been conducted using provided resources by Google:

[1] `https://colab.research.google.com/`

- Dual Intel® Xeon ® CPU @ 2.20GHz.
- A single 12GB NVIDIA® Tesla® K80 GPU

The free version has limited the development of the proposed model because GPU can be used up to 12 hours continuosly.

4.2 Datasets Description

The model evaluation is made on the two different available datasets in order to test the implemented model with different inputs. The section describes characteristics and differences of the two datasets.

SciDuet

According to [36], the SciDuet (Scientific Document Slide Match) dataset consists of paper-slide pairs scraped from some online anthologies:

- International Conference on Machine Learning (ICML'19),
- Neural Information Processing Systems (NeurIPS'18& '19),
- Association for Computational Linguistics (since ACL'79).

Text from dataset was extracted through the combination of Grobid, IBM Watson Discovery package[2] and OCR by *pytesseract*[3]. Further dataset cleaning was performed by pruning equation and floating caption and by removing duplicate lines. SciDuet consists of 1,088 paper-slide pairs decomposed on 952-55-81 in the Train-Dev-Test split. However, released SciDuet dataset contains the full Dev and Test sets and a portion of the Train dataset (136 paper-slide pairs).

	#papers	#slides	SC-len
train	952	8,123	55.1
dev	55	733	63.4
test	81	1,178	52.3

Table 4.1: SciDuet Dataset statistics: 'SC-len' represents the average token length for slide contents

[2] `https://www.ibm.com/cloud/watson-discovery`

[3] `https://pypi.org/project/pytesseract`

APPreD

The APPreD (ACL Papers' PREsentations Dataset) dataset, proposed in thesis work, was crawled from ACL online anthology and consists of 466 paper-slide pairs from recent years' computer linguistic conferences. The dataset has 401-20-45 paper-slide pairs in the Train-Dev-Test split. Text on papers was extracted through GROBID [39], while text on slides was extracted through *pdfplumber* from *PDFMiner*. As explained in section 3.1.1, Dataset was then pre-processed by analyzing the trend of the symbols percentage present in each paper and slide sections and pruning the sentences that contained a percentage greater than a selected threshold based on the output trend. Dataset symbols cleaning is intended to make the automatically extracted text more comprehensible to the machine learning model.

Dataset	#pair paper-slides (released)	avg sections per paper	avg sentences per section	avg token length per section
SciDuet	1,088(272)	12.82	14.56	195.53
APPreD	466	14.70	11.57	171.16

Table 4.2: Datasets statistics: comparison of the two datasets about paper features: number of sections for paper, number of sentences per section and the average token length per sections content.

Dataset	#pair paper-slides (released)	avg slides per paper	avg sentences per slide	avg token length per slide
SciDuet	1,088(272)	15.48	4.28	43.79
APPreD	466	16.63	7.82	69.68

Table 4.3: Datasets statistics: comparison of the two datasets about slides features: number of slides per paper, number of sentences per slide and the average token length per slides content.

Table 4.2 shows the statistics of the two different datasets about papers content. Table 4.3 compares slides content and shows that APPreD dataset has a number of slides and corresponding sentences length greater than Sciduet.
In addition, Table 4.4 shows the average token length of paper sections after IMRaD classification. The number of tokens grows because sentences of same topic are

IMRaD topic	avg token length Sciduet papers	avg token length APPreD papers
Abstract	162.82	163.66
Introduction	755.02	697.58
Method	3,203.09	3,403.51
Result	731.88	678.20
Conclusion	311.78	271.34

Table 4.4: Datasets statistics: the average token length per IMRaD topic and Abstract for both datasets.

linked together creating larger sections than original one. This value is useful to understand how much tokens the model has to receive as input.

4.3 Evaluation Metrics

In order to measure the quality of text in the generated slides, automatic metrics specifically designed to evaluate slide generation models have been used. In particular, ROUGE, introduced in [51] by Lin et al., which stand for Recall-Oriented Understudy for Gisting Evaluation represents the most common measures used in state-of-the-art to determine a quality of generated summary by comparing it to ground-truth summaries created by humans.
ROUGE is composed of a set of metrics and the most used in summarization tasks are:

- *ROUGE-N*, which measures how many *n-grams* match between proposed model generated text and human reference.
  An *n-gram* represents n words or tokens grouped together. A unigram (1-gram) consists of a single word groups, while a bigram (2-gram) is composed of two consecutive words. In this work, ROUGE-1 and ROUGE-2 have been evaluated in order to measure the match rate of unigrams and bigrams between model output and ground-truth.

- *ROUGE-L*, which is a measure based on longest common subsequences (LCS) between the two compared summaries. It means that ROUGE-L metric counts the longest sequence of tokens/words which is shared. A longer shared sequence indicates more similarity between two summaries.

For all used metrics (ROUGE-1, ROUGE-2, ROUGE-L), recall, precision and F-score have been calculated.

Recall measures the number of matched n-grams between model output and reference divided by the total number of n-grams in reference summary (Equation 4.1).

$$ROUGE - N_{recall} = \frac{\sum_{S \in Reference} \sum_{gram_n \in S} count_{match}(gram_n)}{\sum_{S \in Reference} \sum_{gram_n \in S} count(gram_n)} \tag{4.1}$$

It is in charge of evaluating how much the proposed model captures of all the information contained in ground-truth.

Precision metric is calculated by dividing shared n-grams by the total number of n-grams in model summary, as shown in Equation 4.2.

$$ROUGE - N_{precision} = \frac{\sum_{S \in Reference} \sum_{gram_n \in S} count_{match}(gram_n)}{\sum_{S \in ModelSummaries} \sum_{gram_n \in S} count(gram_n)} \tag{4.2}$$

Precision is fundamental in summaries because it measures how much of the model output summary is relevant and needed. In fact, a generated summary could be extremely long capturing all words contained in the reference summary obtaining an high recall score but it could contain words that are not necessary for summary task.

F-score is computed in order to balance the scores of *precision* and *recall* and to give a measure of model performances.

$$F = 2 * \frac{precision * recall}{precision + recall} \tag{4.3}$$

In this work, ROUGE metrics are applied in three different ways to evaluate generated slides:

- *ROUGE Overall*, where the output of each IMRaD topic is merged and the metrics are computed in order to measure the accuracy of entire produced text regardless of classification. It is calculated for both datasets.
- *ROUGE InterClass*, where the idea is to evaluate how much information of each single topic the model has been able to capture. So, the entire output text has been compared to reference topic text one by one. It is calculated on *SciDuet* dataset in order to compare the implemented method with *D2S* of which classification of prediction is not available.
- *ROUGE IntraClass*, is applied on *APPreD* dataset and it evaluates the quality of generated text for each IMRaD class.

In addition, *ROUGE Overall* is computed, also, to evaluate the impact of data pre-processing on fine-tuning, explained in section 3.1.

4.4 Experimental Setup

All fine-tuning has been done on a single 12GB K80 GPU in parallel on PyTorch. The code adapts the transformers models from Huggingface [46]. Hyperparameters model are fine-tuned on the dev set. The proposed model has been fine-tuned over *LED-base-16384*, a small LED version pre-trained on the arxiv dataset. ROUGE evaluation is applied to obtained prediction with test set.

BART A BART summarization model pre-trained with xsum dataset [24] has been fine-tune on the train set of both dataset (*SciDuet* and Thesis Propose). Due to computational resources limits a distilled BART version has been used. Batch size is set to 2 and maximum input token length to 1024. Min and max output token length are set to 100 and 512 for Method topic and to 50 and 128 for the rest of IMRaD topics.

PEGASUS Similarly to BART, a pre-trained PEGASUS summarization model has been fine-tuned and tested with both datasets. Fine-tuning parameters follows BART setup.

D2S Unfortunately, due to computational GPU limits and to errors contained in the released code it has not been possible to fine-tune D2S model with the proposed dataset. Therefore, the published results obtained with *SciDuet* dataset have been considered for evaluation. According to [36], D2S model was fine-tuned with a batch size of 4 and a maximum input token length of 1024. Min and max output token lengths were set to 64 and 128.

Implemented method (LED-IMRaD) As described in section 3.3, the proposed slide generation model has been fine-tuned and tested with both datasets with a batch size of 2. The maximum input token length has been set to 8192, while min and max token lengths depend on IMRaD topic. For Method class they have been set to 100 and 512, for the other topic to 50 and 128. In addition, a LED-IMRaD version with no abstract as context in fine-tuning has been tested.

4.5 Results and Discussion

ROUGE Overall

As explained in section 4.3, in order to evaluate the quality of the entire text of generated slides, ROUGE metrics are computed for all text with no IMRaD division. T-TEST is used for each metric to compare models' performances and it evidences whether improvements are statistically relevant.

Model	ROUGE-1			ROUGE-2			ROUGE-L		
	P	R	F	P	R	F	P	R	F
BART	16.62*	26.10*	18.15	3.35	5.16*	3.63	13.97*	23.29*	15.48*
Pegasus	27.74*	19.07*	22.15*	6.91	4.96*	5.34*	23.55	16.46*	19.11*
D2S	18.30	**30.31**	20.47*	4.73*	**7.79**	5.26*	16.86	**27.21**	19.08*
LED BASE imrad (no Context)	27.81*	23.82*	23.93	7.16*	7.26*	6.67	24.54*	21.22*	21.24
LED BASE imrad (thesis model)	**28.30**	23.90*	**24.29**	**7.27**	7.29*	**6.72**	**25.42**	21.56*	**21.86**

Table 4.5: ROUGE Overall *SciDuet*, statistical significance improvements are starred

Table 4.5 shows the results for ROUGE overall on *SciDuet* dataset. D2S model reaches higher recall values because, according to min and max output token lengths setup, illustrated in section 4.4, it produces longer summaries than LED-IMRaD model output, but in terms of F-score, which gives a balanced score between precision and recall, the results show that the proposed method outperforms D2S and baseline models across all metrics. Moreover, the scores highlight the fact that Abstract as context during fine-tuning helps the model to reach high performances.

Results in Table 4.6 confirm the best performances obtained by proposed model which outperforms,also, baselines and no context version across all metrics with thesis proposed dataset *APPreD*.

Model	ROUGE-1			ROUGE-2			ROUGE-L		
	P	R	F	P	R	F	P	R	F
BART	25.82	23.73*	22.40	6.65	7.01	5.99*	22.93	21.05*	19.83
Pegasus	24.56*	18.40*	21.03*	6.46	4.80*	5.51*	22.44*	15.90*	18.61*
LED BASE imrad (no Context)	25.98	26.23	23.75	6.75	8.23	6.58	23.05	23.30	21.06
LED BASE imrad (thesis model)	**26.43**	**26.31**	**24.09**	**6.85**	**8.24**	**6.62**	**23.87**	**23.67**	**21.67**

Table 4.6: ROUGE Overall *APPreD*, statistical significance improvements are starred

ROUGE InterClass

The results of *ROUGE InterClass* help to evaluate the relevance of generated summaries for each IMRaD topic. In order to have D2S scores, its *SciDuet* dataset has been classified by the same method explained in section 3.2 and entire predictions from all examined models have been compared to it.

Table 4.7 shows the obtained results and the highest F-score values are highlighted for each ROUGE metric. LED-BASE-Imrad proposed version outperforms across almost all topics. D2S has a greater recall value which enables it to has a better performance in Method topic for ROUGE-L, because of the greater length of its summaries in the rest of sections where LED-BASE-Imrad min output tokens is set to 50 while in D2S is set to 64.

ROUGE IntraClass

Due to the impossibility of classifying the *D2S* predictions the obtained scores for each IMRaD section could not be compared with implemented method results.

So, *ROUGE IntraClass* has been computed with *APPreD* in order to compare the thesis model with baselines and with the no context version. Table 4.8 shows the achieved results where the implemented method reaches the best performances in all topics in terms of F-score. In Conclusion topic the no context version has similar values because abstract does not add relevant information that may help the model understanding in this section.

Dataset Cleaning Impact

In addition, a version equal to the proposed model has been fine-tuned and tested with proposed Dataset with any cleaning operations. Table 4.9 shows the obtained results where it proved that symbols pruning from dataset helps the model to understand data and to obtain better performances across all metrics.

Finally, the results computed by ROUGE show that implemented method outperforms state-of-the-art for slide generation task with both available datasets. This is due to the capacity of LED to receive in input a greater number of tokens (up to 16384), which represents an important advantage in long document summarization. Moreover, the results confirm the hypothesis that integrating abstract into encoding phase during fine-tuning can improve the model to capture all relevant information through paper sections.
The results show, also, that dataset cleaning represent a fundamental stage for model learning, in particular for slide generation due to highly creative nature of human reference slides that make ground-truth slides difficult to understand by an automatic system during training.

Table 4.7: ROUGE InterClass *SciDuet*

Rouge IntraClass **SciDuet**		ROUGE-1			ROUGE-2			ROUGE-L		
Imrad	**Model**	P	R	F	P	R	F	P	R	F
Intro	LED-BASE-imrad (thesis propose)	5.35	35.30	**8.87**	0.85	8.63	**1.54**	4.98	33.29	**8.25**
	LED-BASE-imrad (noContext)	4.59	34.09	7.83	0.62	8.77	1.13	4.16	31.28	7.10
	D2S	4.80	43.41	8.10	0.85	7.55	1.52	4.34	40.52	7.40
	Pegasus	4.58	31.11	7.98	0.90	6.22	1.15	4.10	28.33	7.16
	BART	4.76	38.04	4.76	0.81	8.83	1.45	4.51	36.43	7.71
Method	LED-BASE-imrad (thesis propose)	22.81	24.96	**23.84**	5.11	6.80	**5.83**	20.33	22.48	21.35
	LED-BASE-imrad (noContext)	19.04	27.62	20.72	4.09	7.42	4.88	16.88	24.68	18.36
	D2S	19.72	27.16	22.84	4.60	7.99	5.83	19.36	31.42	**23.01**
	Pegasus	25.40	20.52	20.91	5.00	4.74	4.44	21.67	17.63	17.84
	BART	20.50	28.05	22.12	5.08	8.03	5.76	18.36	25.25	19.82
Result	LED-BASE-imrad (thesis propose)	8.50	31.60	**12.45**	1.53	6.39	**2.46**	7.76	29.07	**12.24**
	LED-BASE-imrad (noContext)	7.56	31.62	11.44	1.12	6.69	1.83	6.73	28.71	10.21
	D2S	7.83	42.54	12.27	1.28	10.67	2.29	7.13	39.59	11.23
	Pegasus	9.51	25.51	12.43	1.43	4.91	2.06	8.44	23.15	11.23
	BART	7.05	32.83	10.74	1.15	7.62	1.87	6.46	30.35	9.84
Conclusion	LED-BASE-imrad (thesis propose)	8.67	45.33	**14.55**	1.87	15.10	**3.26**	8.02	42.11	**13.57**
	LED-BASE-imrad (noContext)	6.96	44.75	11.75	1.38	14.05	2.49	6.41	41.58	10.84
	D2S	6.53	50.68	11.09	1.47	16.35	2.52	5.80	46.13	9.92
	Pegasus	7.75	38.12	12.88	1.78	11.11	2.98	9.74	34.61	13.55
	BART	7.04	47.38	12.00	1.47	15.83	2.65	6.51	44.09	11.10

Table 4.8: ROUGE IntraClass *APPreD*

Rouge IntraClass **APPreD**		ROUGE-1			ROUGE-2			ROUGE-L		
Imrad	**Model**	P	R	F	P	R	F	P	R	F
Intro	LED-BASE-imrad (thesis model)	12.66	23.55	**14.61**	2.98	5.12	**3.30**	11.05	14.46	**12.52**
	LED-BASE-imrad (noContext)	11.14	18.80	11.68	1.87	2.99	1.80	10.04	16.43	10.40
	Pegasus	13.03	22.79	13.62	1.89	3.12	1.94	10.64	19.31	11.28
	BART	10.49	18.80	10.62	1.41	3.08	1.23	8.88	15.21	8.72
Method	LED-BASE-imrad (thesis model)	26.73	18.08	**19.61**	6.11	4.66	**4.72**	23.73	16.02	**17.36**
	LED-BASE-imrad (noContext)	25.26	17.45	18.67	5.44	4.26	4.27	22.02	15.12	16.19
	Pegasus	34.64	7.75	11.82	7.89	1.49	2.36	30.36	6.71	10.22
	BART	25.63	15.06	17.36	5.86	3.80	4.14	22.37	13.06	15.03
Result	LED-BASE-imrad (thesis model)	13.23	24.13	**14.63**	3.28	7.27	**3.88**	10.54	11.16	**10.84**
	LED-BASE-imrad (noContext)	11.54	12.80	10.67	1.55	2.36	1.61	10.31	11.75	9.65
	Pegasus	13.39	12.86	11.15	1.28	0.94	0.98	11.36	11.22	9.47
	BART	9.16	13.27	8.98	1.71	2.55	1.66	8.19	12.02	8.04
Conclusion	LED-BASE-imrad (thesis model)	15.82	32.09	**21.19**	5.90	9.57	**6.49**	15.18	20.14	17.31
	LED-BASE-imrad (noContext)	17.30	26.98	21.08	5.61	9.52	6.47	15.95	24.85	**17.73**
	Pegasus	16.10	18.62	15.80	3.18	4.08	3.27	13.98	16.16	13.63
	BART	16.61	22.13	17.39	4.32	5.99	4.40	14.89	19.75	15.43

Model	ROUGE-1			ROUGE-2			ROUGE-L		
	P	R	F	P	R	F	P	R	F
LED BASE imrad (no Cleaning)	25.30	25.80	23.51	6.79	7.97	6.56	22.69	23.02	20.98
LED BASE imrad (thesis model)	**26.43**	**26.31**	**24.09**	**6.85**	**8.24**	**6.62**	**23.87**	**23.67**	**21.67**

Table 4.9: Effect of *APPreD* cleaning on ROUGE overall

Chapter 5

Conclusion

This thesis project aims to generate automatic presentation slides from scientific paper documents. The task is addressed using an abstractive summarization approach in order to reproduce slides more similar to human presentations. For this reason, the objective is to generate slides that follow the common structure of corresponding scientific papers by summarizing paper content section by section. The work exploits the similar structure of each paper made by Introduction, Method, Results and Conclusion (IMRaD) main topics assigning paper sections to the belonging common topic and then generate different abstractive summaries for each argument in parallel. Inspired by Transformers by Huggingface work [17], a pre-trained summarization model is fine-tuned in four different versions belonging to IMRaD topics to tackle this challenge.The project proposes, also, a new dataset (*APPreD*) collected from ACL online anthology. The dataset will be publicly released to foster future research on this domain.

Unfortunately, due to computational available resources limits, state-of-the-art represented by *D2S* work is unable to be fine-tuned on released dataset in order to compare its results on *APPreD*. This issue and the lack of material released from other existing work have limited the experimentation of implemented method which can be considered one of the few works to use a full abstractive approach for slide generation task. Some baseline summarization models are fine-tuned and tested with the proposed dataset. However, in order to compare the obtained results with state-of-the-art results the implemented method is, also, fine-tuned with *D2S* dataset (*SciDuet*). Automated evaluations suggest that the implemented system outperforms some examined baselines and *D2S* for the document-to-slide task.
The evaluation also confirms the assumptions made during the method implementation that a pre-processing step is necessary to facilitate the understanding of the dataset by the automatic system and that adding the Abstract section as context

during fine-tuning helps the model to reach better performance because it allows to capture more relevant information. This is possible because of the choice of Longformer Encoder-Decoder (LED) as summarization model which is able to receive as input a high number of tokens compared to the other studied existing models.
However, the possibility of having more computational available resources can improve the proposed system with the aim of being able to impose itself as state of the art in slide-generation task.

Considering the rapid growth of deep learning techniques in this task, this project can be further developed in order to be able to address more domains. Moreover, the results obtained in textual domain can be improved making the system adaptable to multimodal summarization that deals with summarizing documents including text and figures together. Recent studies in cross-modal retrieval can help to find a multimodal joint embedding space with images and text [52].
Furthermore, due to the lack of labeling data, a possible investigation can be done for unsupervised learning which allow the model to perform without needing any human intervention. This would allow the creation of paper presentations before they are created by humans. In this sense, a possible development of this thesis project is exploring the self-supervised learning technique, e.g. Contrastive Learning, in section-aware slide generation domain. It consists of learning general features of a dataset without labels for the purpose of recognizing to which class each section of an article belongs.

Bibliography

[1] Elena Lloret and Manuel Palomar. «Text summarisation in progress: a literature review». In: *Artificial Intelligence Review* 37.1 (2012), pp. 1–41 (cit. on p. 5).

[2] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. *Text Summarization Techniques: A Brief Survey*. 2017. arXiv: `1707.02268 [cs.CL]` (cit. on p. 5).

[3] Dragomir R Radev, Eduard Hovy, and Kathleen McKeown. «Introduction to the special issue on summarization». In: *Computational linguistics* 28.4 (2002), pp. 399–408 (cit. on p. 5).

[4] Ted Dunning. «Accurate Methods for the Statistics of Surprise and Coincidence». In: *Comput. Linguist.* 19.1 (Mar. 1993), pp. 61–74. ISSN: 0891-2017 (cit. on p. 5).

[5] Günes Erkan and Dragomir R Radev. «Lexrank: Graph-based lexical centrality as salience in text summarization». In: *Journal of artificial intelligence research* 22 (2004), pp. 457–479 (cit. on p. 6).

[6] Yihong Gong and Xin Liu. «Generic text summarization using relevance measure and latent semantic analysis». In: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. 2001, pp. 19–25 (cit. on p. 7).

[7] N Moratanch and S Chitrakala. «A survey on abstractive text summarization». In: *2016 International Conference on Circuit, power and computing technologies (ICCPCT)*. IEEE. 2016, pp. 1–7 (cit. on p. 7).

[8] Regina Barzilay, Kathleen McKeown, and Michael Elhadad. «Information fusion in the context of multi-document summarization». In: *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*. 1999, pp. 550–557 (cit. on p. 7).

[9] Pierre-Etienne Genest and Guy Lapalme. «Fully Abstractive Approach to Guided Summarization». In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Jeju Island, Korea: Association for Computational Linguistics, July 2012, pp. 354–358. URL: https://aclanthology.org/P12-2069 (cit. on p. 7).

[10] Albert Gatt and Ehud Reiter. «SimpleNLG: A realisation engine for practical applications». In: *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)*. 2009, pp. 90–93 (cit. on p. 7).

[11] Gerard Salton, A Wong, and CS Yang. «Avector space model for automatic indexing». In: *Communications* (1975) (cit. on p. 8).

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL] (cit. on p. 8).

[13] Jeffrey Pennington, Richard Socher, and Christopher D Manning. «Glove: Global vectors for word representation». In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543 (cit. on p. 8).

[14] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. «Learning representations by back-propagating errors». In: *nature* 323.6088 (1986), pp. 533–536 (cit. on p. 9).

[15] Ralf C. Staudemeyer and Eric Rothstein Morris. *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*. 2019. arXiv: 1909.09586 [cs.NE] (cit. on p. 10).

[16] Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K. Reddy. «Neural Abstractive Text Summarization with Sequence-to-Sequence Models». In: *ACM/IMS Trans. Data Sci.* 2.1 (Jan. 2021). ISSN: 2691-1922. DOI: 10.1145/3419106. URL: https://doi.org/10.1145/3419106 (cit. on p. 10).

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL] (cit. on pp. 11, 12, 45).

[18] Maxime. *What is a Transformer?* https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04. [Online]. 2019 (cit. on p. 12).

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL] (cit. on p. 13).

[20] Manzil Zaheer et al. *Big Bird: Transformers for Longer Sequences*. 2021. arXiv: 2007.14062 [cs.LG] (cit. on p. 13).

[21] Iz Beltagy, Matthew E. Peters, and Arman Cohan. *Longformer: The Long-Document Transformer*. 2020. arXiv: 2004.05150 [cs.CL] (cit. on pp. 14, 15, 31).

[22] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. arXiv: 1910.13461 [cs.CL] (cit. on pp. 14, 33).

[23] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. «Abstractive text summarization using sequence-to-sequence rnns and beyond». In: *arXiv preprint arXiv:1602.06023* (2016) (cit. on p. 14).

[24] Shashi Narayan, Shay B Cohen, and Mirella Lapata. «Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization». In: *arXiv preprint arXiv:1808.08745* (2018) (cit. on pp. 14, 38).

[25] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. «PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization». In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 11328–11339. URL: http://proceedings.mlr.press/v119/zhang20ae.html (cit. on pp. 15, 33).

[26] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. *Layer Normalization*. 2016. arXiv: 1607.06450 [stat.ML] (cit. on p. 15).

[27] Adam Roberts, Colin Raffel, Katherine Lee, Michael Matena, Noam Shazeer, Peter J. Liu, Sharan Narang, Wei Li, and Yanqi Zhou. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. Tech. rep. Google, 2019 (cit. on p. 15).

[28] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. *A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents*. 2018. arXiv: 1804.05685 [cs.CL] (cit. on p. 15).

[29] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. *Sequence Level Training with Recurrent Neural Networks*. 2016. arXiv: 1511.06732 [cs.LG] (cit. on p. 15).

[30] Alexander M. Rush, Sumit Chopra, and Jason Weston. *A Neural Attention Model for Abstractive Sentence Summarization*. 2015. arXiv: 1509.00685 [cs.CL] (cit. on p. 17).

[31] Y. Hu and X. Wan. «PPSGen: Learning-Based Presentation Slides Generation for Academic Papers». In: *IEEE Transactions on Knowledge and Data Engineering* 27.4 (2015), pp. 1085–1097. DOI: 10.1109/TKDE.2014.2359652 (cit. on pp. 17, 18).

[32] Sida Wang, Xiaojun Wan, and Shikang Du. «Phrase-Based Presentation Slides Generation for Academic Papers». In: *Proceedings of the AAAI Conference on Artificial Intelligence* 31.1 (Feb. 2017). URL: https://ojs.aaai.org/index.php/AAAI/article/view/10481 (cit. on p. 17).

[33] Jian Wu Athar Sefid. «Automatic Slide Generation for Scientific Papers». In: *Third International Workshop on Capturing Scientific Knowledge co-located with the 10th International Conference on Knowledge Capture (K-CAP 2019),SciKnow@K-CAP 2019* (). URL: https://par.nsf.gov/biblio/10173903 (cit. on p. 18).

[34] Tsu-Jui Fu, William Yang Wang, Daniel McDuff, and Yale Song. *DOC2PPT: Automatic Presentation Slides Generation from Scientific Documents*. 2021. arXiv: 2101.11796 [cs.CV] (cit. on pp. 18, 33).

[35] Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. *Efficient Summarization with Read-Again and Copy Mechanism*. 2016. arXiv: 1611.03382 [cs.CL] (cit. on p. 19).

[36] Edward Sun, Yufang Hou, Dakuo Wang, Yunfeng Zhang, and Nancy X. R. Wang. *D2S: Document-to-Slide Generation Via Query-Based Text Summarization*. 2021. arXiv: 2105.03664 [cs.CL] (cit. on pp. 19, 20, 33, 34, 38).

[37] Da-Wei Li, Danqing Huang, Tingting Ma, and Chin-Yew Lin. «Towards Topic-Aware Slide Generation For Academic Papers With Unsupervised Mutual Learning». In: *35th AAAI Conference on Artificial Intelligence (AAAI-21)*. Association for the Advancement of Artificial Intelligence. Association for the Advancement of Artificial Intelligence, Feb. 2021. URL: https://www.microsoft.com/en-us/research/publication/towards-topic-aware-slide-generation-for-academic-papers-with-unsupervised-mutual-learning/ (cit. on p. 21).

[38] Cagliero Luca and La Quatra Moreno. *Automatic slides generation in the absence of training data*. http://hdl.handle.net/11583/2919520. 2021 (cit. on p. 21).

[39] *GROBID*. https://github.com/kermitt2/grobid. 2008–2021. swh: 1:dir:dab86b296e3c3216e2241968f0d63b68e8209d3c (cit. on pp. 24, 32, 35).

[40] Jianguo Wu. «Improving the writing of research papers: IMRAD and beyond». In: *Landscape Ecology* 26.10 (Dec. 2011), pp. 1345–1349. ISSN: 1572-9761. DOI: 10.1007/s10980-011-9674-3. URL: https://doi.org/10.1007/s10980-011-9674-3 (cit. on p. 27).

[41] Yang Liu and Mirella Lapata. *Text Summarization with Pretrained Encoders*. 2019. arXiv: 1908.08345 [cs.CL] (cit. on p. 29).

[42] Sam Shleifer and Alexander M. Rush. *Pre-trained Summarization Distillation*. 2020. arXiv: 2010.13002 [cs.CL] (cit. on p. 29).

[43] Jonathan J Webster and Chunyu Kit. «Tokenization as the initial phase in NLP». In: *COLING 1992 Volume 4: The 14th International Conference on Computational Linguistics*. 1992 (cit. on p. 29).

[44] Rico Sennrich, Barry Haddow, and Alexandra Birch. *Neural Machine Translation of Rare Words with Subword Units*. 2016. arXiv: 1508.07909 [cs.CL] (cit. on p. 31).

[45] *Pdfplumber reference website*. URL: https://pypi.org/project/pdfplumber/0.1.2/ (cit. on p. 32).

[46] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL] (cit. on pp. 32, 38).

[47] *PyTorch reference website*. URL: https://pytorch.org/ (cit. on p. 32).

[48] *NumPy reference website*. URL: https://numpy.org/ (cit. on p. 32).

[49] *spaCy reference website*. URL: https://spacy.io/ (cit. on p. 32).

[50] *MatPlotlib reference website*. URL: https://matplotlib.org/ (cit. on p. 32).

[51] Chin-Yew Lin. «Rouge: A package for automatic evaluation of summaries». In: *Text summarization branches out*. 2004, pp. 74–81 (cit. on p. 36).

[52] Ryan Kiros, Ruslan Salakhutdinov, and Richard S. Zemel. *Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models*. 2014. arXiv: 1411.2539 [cs.LG] (cit. on p. 46).