POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Automatic slide generation from scientific papers based on multimodal learning

Supervisors

Prof. Luca CAGLIERO

Dott. Moreno LA QUATRA

Candidate

Enrico GIRARDI

October 2022

Table of Contents

List of Figures IV

1 Introduction 1

2 NLP framework and Related Works 5
- 2.1 Fundamentals 5
  - 2.1.1 Word Embedding 5
  - 2.1.2 Sentence embedding 6
  - 2.1.3 Transformers 7
  - 2.1.4 BERT and its variations 9
  - 2.1.5 Graphs 11
  - 2.1.6 Graph Attention Networks 12
- 2.2 Prior work on summarization 13
  - 2.2.1 Automatic summarization 13
  - 2.2.2 Slide Generation from scientific papers 17
  - 2.2.3 Multimodal Learning 20
- 2.3 Contribution 22

3 Method 23
- 3.1 Ranking Module 24
- 3.2 Extractive Module 25
- 3.3 Multimodality 27

4 Experimental Results 29
- 4.1 Hardware 29
  - 4.1.1 HPC@POLITO 30

4.1.2 CUDA 31

4.2 Software 32

4.2.1 Python Libraries 32

4.2.2 GROBID 33

4.2.3 PDFFigures 2.0 34

4.2.4 CLIP fine-tuning 34

4.3 Dataset 35

4.4 Experimental Design 36

4.4.1 ROUGE Metrics 36

4.4.2 Procedure 37

4.5 Results 38

4.5.1 Parameters 38

4.5.2 Baselines Comparison 38

4.5.3 Supervised vs. Unsupervised 39

5 Conclusion and future work 43

Bibliography 45

List of Figures

2.1 Word Embedding Vectors 6
2.2 Transformer Architecture 8
2.3 BERT 9
2.4 Self-Attention: more relevant words for the word "making" [9] 10
2.5 A graph that represents airports, airport restaurants, and flights [14] 12
2.6 Graph Attention Network as in [18] 12
2.7 Extractive and Abstractive Summarization [19] 13
2.8 LexRank [16] 15
2.9 HIBERT [24] 16
2.10 DOC2PPT summarization pipeline [1] 17
2.11 DOC2PPT model in detail [1] 19
2.12 Clip model [5] 21

3.1 The Summarization Pipeline 23
3.2 Sample from the Graph used in the Extractive Summarization Module 26

4.1 Generic HPC system Architecture [32] 29
4.2 CUDA Workflow [33] 31

Chapter 1

Introduction

Automatic Text Summarization is still today a very active research topic with several methods available and hundreds of applications, summarization from scientific papers is one of them.
In general, text Summarization can be extractive or abstractive.
This work will focus only on extractive methods, therefore the summaries will be composed of some sentences taken from the original text and re-arranged together.
The output from the automatic summarization of a scientific paper can be used to create a presentation for a scientific work i.e. slides or can be read to understand the main contents and gain more information than simply reading the Abstract and conclusions.
There are already lots of works in summarization from scientific papers and, although astonishing results, the problem is far from being solved.
Furthermore, this work will also focus on the multimodal aspect of the scientific summarization i.e. how pictures can be included in the summary in a meaningful way to make presentation slides in an automated way, in the time of writing only one paper from 2021 (DOC2PPT) [1] has done this type or research.
The model proposed by the authors of [1] uses paper/slide pairs during training and is capable of outputting a slide deck from a scientific paper during test time.

DOC2PPT [1] addresses the following tasks:

- text summarization, based on a Seq2seq model [2] (Recurrent Neural Networks used for text paraphrasing).
- figure extraction, which entails identifying the most relevant figures in the paper by using a shared embedding space for images and text. Images are embedded using a ResNet-152 [3] and text is encoded using RoBERTa [4] (a Transformer model, read below for details) , then the embeddings are projected to a shared embedding space using a two-layer multilayer perceptron (MLP).
- slide generation, which relies on a module that is able to put paraphrased objects (text and/or images) in a visually appealing manner using a two-layer MLP to predict the position and size of the object.

The idea of the project is to produce an output like the one of DOC2PPT combining simpler blocks. Therefore, this Master's Degree Thesis aims to investigate the problem of automatic slide generation based on multimodal content.
This work will use a wide range of tools from various branches of Machine Learning to address the problem, the purpose of this report is to illustrate all the tools used and the most important results and to be useful reading to anyone who wants to approach this field.
To better understand the idea of the project it can be useful to introduce two deep learning models used:

- Transformers: a family of deep learning models commonly used in many NLP tasks such as text classification, sentence similarity, etc. Transformers are the current state-of-the-art solutions for many NLP problems and they can be used also in Computer Vision tasks. For this work, Transformers are used to perform sentence embedding, i.e., make sentences processable by machine learning models.
- CLIP [5] a multimodal neural network that efficiently learns visual concepts from natural language supervision. More in particular, during training, CLIP sees an image and its caption at the same

time, this allows understanding better the relationship between visual concepts and their textual description.

In particular, the work followed this idea: a state-of-the-art solution in Extractive Summarization for Scientific Papers (no Multimodal setting) like [6] is slightly modified to try to improve its performances using Transformers, then CLIP tries to understand which pictures are more useful for a slide presentation using the text of the summary, so this work also studies how possible is to extend an only-text model like [6] to a Multimodal challenge like Automatic Presentation Slides Generation from Scientific Papers like DOC2PPT [1].
This thesis will follow a specific structure:

- Chapter 2 (Related Works) will explain the context of this work with a quick overview of Text Summarization, Multimodal Learning, and Slide generation, moreover this chapter will cover some fundamentals tools of NLP like Word/Sentence Embedding and Transformers and will give some insights about Graph and Graph Attention Networks.
- Chapter 3 will describe the model adopted and how it works.
- Chapter 4 will illustrate the experimental setting: the hardware and software used, what dataset has been adopted for training/testing, the experiments' procedure, the metrics involved, and the results.
- Chapter 5 will be dedicated to conclusions.

Chapter 2

NLP framework and Related Works

Nowadays more and more data is pushed through the internet with different modalities, one of them being text: news, articles, social media, books, scientific papers, and so on.
With this mass of data is crucial to understand how to automatically elaborate text cleverly and efficiently.
Natural Language Processing is a branch of Machine Learning that deals with word processing and text analysis performed by machines.
NLP is a very active field of research due to its many applications in various scenarios and the variety of tasks involved.

2.1 Fundamentals

2.1.1 Word Embedding

Word Embedding is a subtask of NLP, and represents the first step for a machine to understand the text of a document.
Basically, through Word Embedding, a word is represented by a vector of n-dimensions. A Word Embedding model is trained using text corpora to grasp the *context* of words, then the model associates a specific n-dimensional vector to each word.

The training allows the word embedding model to minimize the vector distance between words that are similar by semantic and syntactic patterns and maximize the distance between unrelated words.
As can be seen in figure 2.1, word embeddings try to project words in a vector space where relationships between words are preserved through distances and other geometric properties.
There exist multiple pre-trained word embedders available online, in this work GloVe [7] will be used.

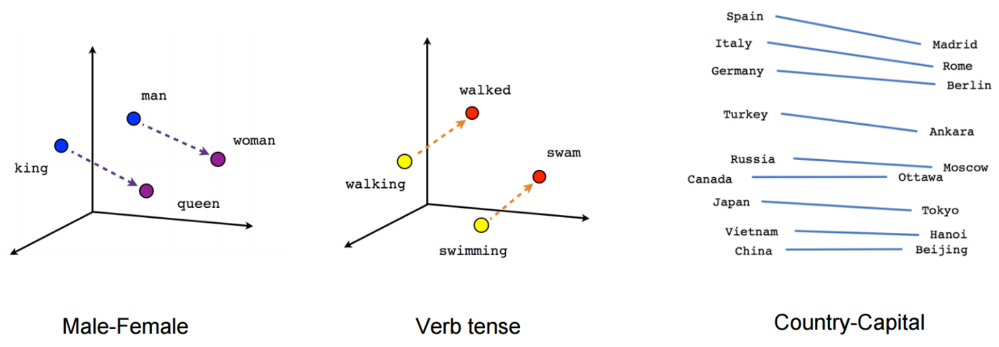


Figure 2.1: Word Embedding Vectors

There are different versions of GloVe according to the dataset used during the training phase and the number of output dimensions.
The version used for this project is the "bigger" one: trained using a dataset composed of Wikipedia articles and the English Gigaword Fifth Edition dataset (newswire text data) [8] and returns a vector of 300 dimensions.

2.1.2 Sentence embedding

For text summarization and other NLP tasks, processing only separated words is not sufficient and processing whole sentences is required, so a proper vector embedding for sentences is needed.
The task is more complex than basic Word Embedding, a good sentence embedder must be able to grasp things such as *context* of the sentence and *relationships* between words.

In other words, sentence embedders can perform *contextualized* embeddings of words: the same word can mean different things according to the context that is given by all the words of the sentence.
A sentence embedder uses all the words in a sentence to understand its context and then performs the embedding of each word, finally it combines all the word embeddings into a single vector that is the sentence embedding.
As in Word Embedding, the output of Sentence Embedding is a vector of n-dimensions capable of representing a sentence in a vector space.
Nowadays the state of the art in sentence embedding is represented by Transformers.
The model proposed will use BiLSTM for sentence embeddings in the first part and Longformer (a Transformer model) in the second part.

2.1.3 Transformers

Transformers were introduced by [9] in 2017, and are currently employed in many state-of-the-art solutions in NLP and Computer Vision. They use an encoder-decoder architecture based on self-attention.
Before Transformers, all the state-of-the-art solutions in NLP relied on recurrent neural networks (RNN) like long short-term memory (LSTM) and gated recurrent units (GRUs).
Due to the intrinsic nature of RNNs, all the text had to be processed sequentially with no opportunity for parallelization during training. The use of Attention in Transformers allows processing the text in a non-sequential way, allowing parallelization. The Attention Mechanism mimics human cognitive attention: some elements in the inputs are more important than others while others are not particularly relevant.
The type of Attention used in Transformers is Self-Attention.
With Self-Attention the input is used to value the importance of each element in the input.

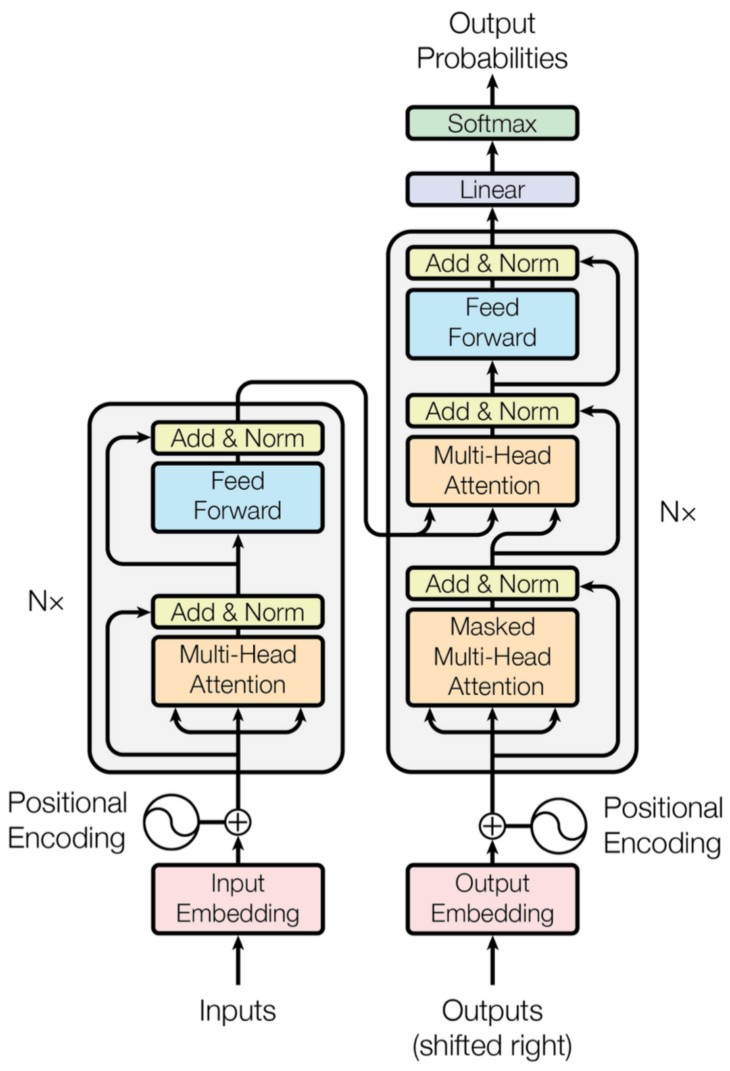


Figure 2.2: Transformer Architecture

The encoder-decoder architecture works as follows:

- The encoder consists of encoding layers that process the input iteratively one layer after another. The function of each encoder layer is to generate encodings that contain information about which parts of the inputs are relevant to each other. It passes its encodings to the next encoder layer as inputs.
- The decoder takes the output of the encoder.

- The decoder consists of decoding layers that work iteratively one layer after another. Each decoder layer does take all the encodings and uses their incorporated contextual information to generate an output sequence.

- Each encoder and decoder layer makes use of an attention mechanism. Both the encoder and decoder layers have a feed-forward neural network for additional processing of the outputs and contain residual connections and layer normalization steps.

According to the task is possible to use only one side of the encoder-decoder architecture, for example, GPT-3 [10] (state of the art in human-like text generation) only uses the decoder side, while BERT [11] (state of the art in sentence encoding) and its variants, use only the encoder-side.

2.1.4 BERT and its variations

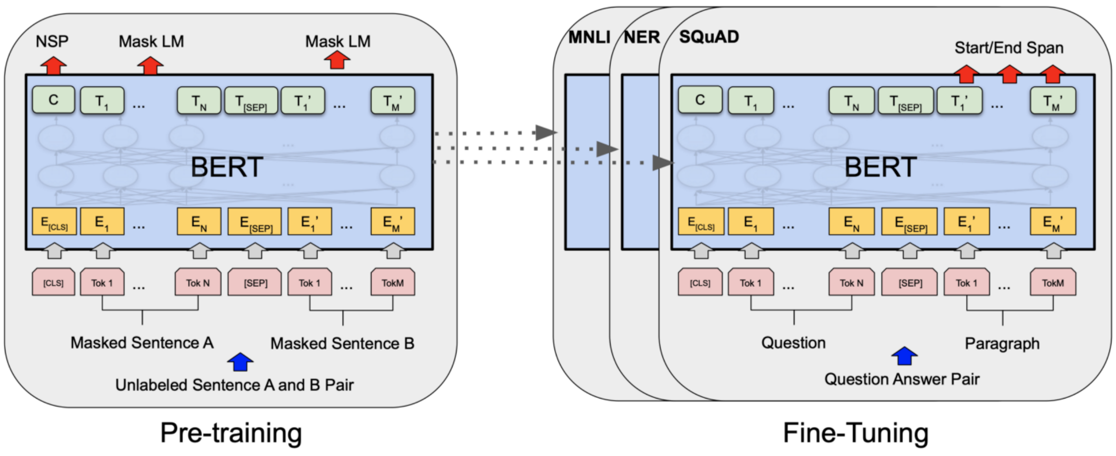


Figure 2.3: BERT

This work will use different BERT versions for sentence encoding. BERT stands for Bidirectional Encoder Representations from Transformers, it's a pre-trained bidirectional model developed by Google, and released in 2018. Since 2020, Google uses BERT in almost every English-language query.

BERT uses a bidirectional architecture (for each word using the previous and next words to understand its context) to perform deep-contextualized word embeddings, i.e. the same word has a different embedding given the context of the sentence.
BERT is available as a pre-trained model, i.e., already trained using a huge amount of generic data and time. It is possible, for the end-user, to further train the model using task-specific data to get better results, this operation is called fine-tuning.
BERT, for example, is trained on BooksCorpus (800M words) [12] and English Wikipedia (2,500M words) but is easy to fine-tune for a specific language domain or task, so there are different versions of BERT for different purposes.
More specifically, BERT uses a stack of Encoders to model a sentence, each word of a sentence is processed using all the other words of a sentence via self-attention mechanisms.
Attention allows an understanding of how words are correlated and what words are more relevant than others.

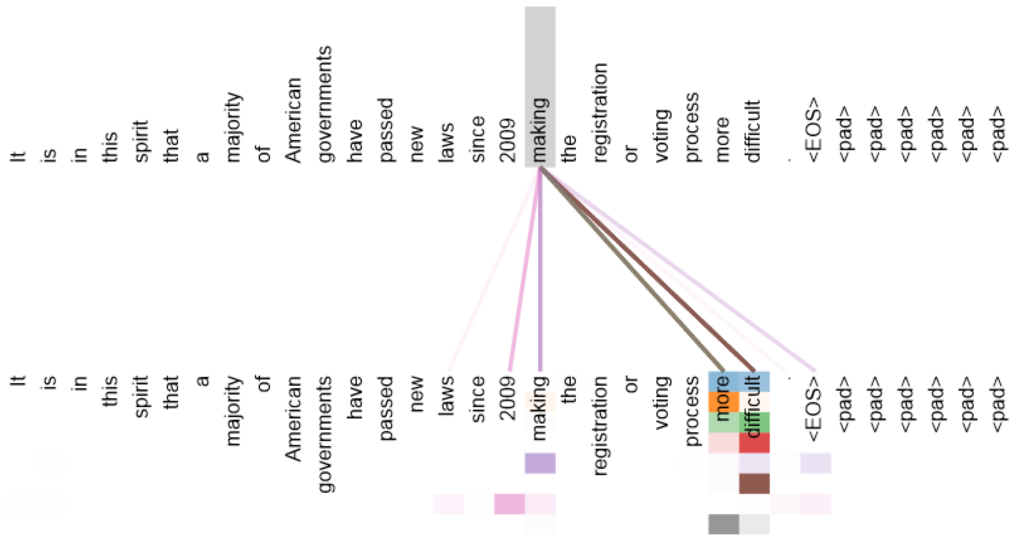


Figure 2.4: Self-Attention: more relevant words for the word "making" [9]

Given queries vectors q_i, keys vectors k_i and values v_i, to calculate the attention :

- calculate the affinity score between q_i and k_j ($e_{ij} = q_i^T k_j$)
- normalize using softmax ($a_{ij} = softmax(e_{ij})$)
- output the weighted sum of values ($output_i = \Sigma_j a_{ij} v_j$)

In a self-attention environment, q, k, and v are represented by word embeddings.
Given a sentence, the model outputs the embedding of a special CLS token (that can be used as sentence embedding) and one vector embedding for each word, these vectors can be aggregated to produce an alternative sentence embedding.

This work uses SciBERT [13] (a BERT version fine-tuned on scientific text) to divide into sentences the text of scientific papers and Longformer to get sentence embedding for the model.
Longformer is based on RoBERTa [4], a version of BERT with more robust and efficient training that use a larger dataset than base BERT. Specifically, Longformer can process sequences larger than 512 tokens (the limit for standard transformers) using sliding windows.

2.1.5 Graphs

Graphs are widely used to model relationships between entities, and thanks to their flexibility they can be used in almost every field.
The basic definition of a graph is a set of nodes (entities) connected by edges (relationships).
From the base definition can be derived various types of graphs, e.g. if edges have a direction it is the case of a *directed* graph, if the edges have weights it is a *weighted* graph if all the nodes model the same entity is an *homogeneous* graph otherwise is an *heterogeneous* graph.
There exist algorithms that use graphs to model and solve a problem, some of them can be found also in NLP.
In NLP graphs can be adopted to model single sentences or entire documents and used by text summarization algorithms like TextRank [15], LexRank [16] and GraphSum [17].

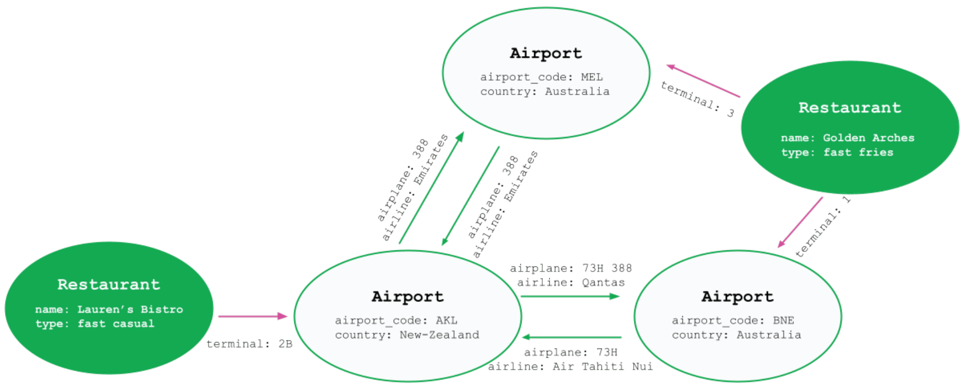


Figure 2.5: A graph that represents airports, airport restaurants, and flights [14]

2.1.6 Graph Attention Networks

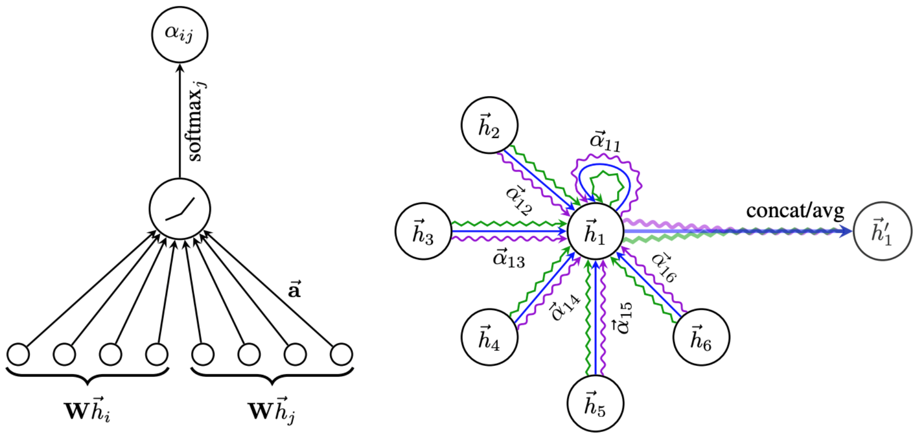


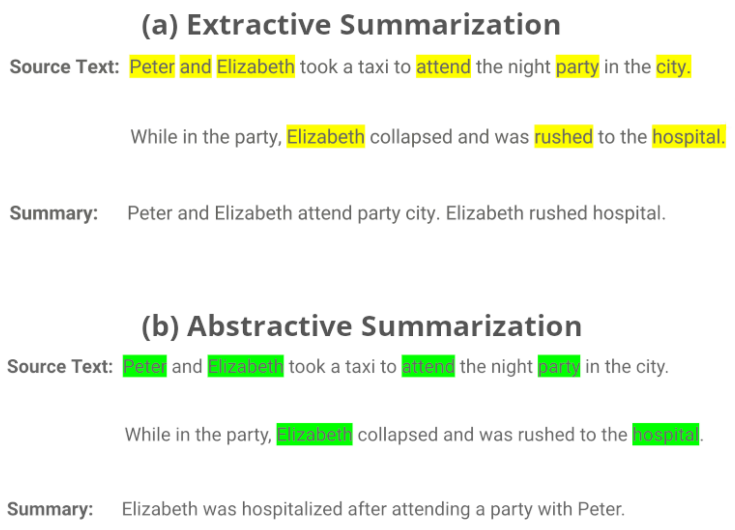
Figure 2.6: Graph Attention Network as in [18]

Graph Attention Networks (GATs) were first introduced in [18] in 2018.
GATs stack layers that use an attention mechanism to model the relationship between a node and its neighbors, this allows for a better understanding of the connection between entities and avoids the shortcomings of methods based on graph convolutions or their approximations.

In [6] (2021), one of the major references for this work, a GAT is used for scientific papers summarization.
More in detail, [6] runs a GAT over a heterogeneous graph (HGAT) where each node can represent a word, a sentence, or an entire section, and edges are used to model specific relationships between them, e.g. words in a sentence, a sentence in a section, sentences in the same sections and so on.

2.2 Prior work on summarization

2.2.1 Automatic summarization



(a) Extractive Summarization
Source Text: Peter and Elizabeth took a taxi to attend the night party in the city.
While in the party, Elizabeth collapsed and was rushed to the hospital.
Summary: Peter and Elizabeth attend party city. Elizabeth rushed hospital.
(b) Abstractive Summarization
Source Text: Peter and Elizabeth took a taxi to attend the night party in the city.
While in the party, Elizabeth collapsed and was rushed to the hospital.
Summary: Elizabeth was hospitalized after attending a party with Peter.

Figure 2.7: Extractive and Abstractive Summarization [19]

Automatic summarization is used to compute a summary of a document using Machine Learning.
The main division in Automatic summarization is between extractive summarization and abstractive summarization:

- Extractive summarization is used to extract chunks of data from the original document and use them to create a summary.

- Abstractive summarization tries to create an original summary of the document after learning a semantic representation of the original content.

The input, and the output, can include only text or text and images (multimodal input/output).
The Machine Learning involved can be shallow or deep, supervised or unsupervised according to the algorithms and models involved.
In this project the input is in the form of scientific papers (Multimodal with both text and images), the output is in the form of presentation slides (again, Multimodal with both text and images), and the summarization approach is *extractive* using Deep Learning in a supervised setting.
Automatic summarization of scientific text is still an open problem, during the bibliographic research for this project numerous works were found:

- [20] (2017) introduces a new dataset (CSPubSum), a interesting metric to evaluate sentence importance (AbstractROUGE) and some useful insight for summarization.
- [21] (2019) introduces a new dataset (ScisummNet) and proposes a summarization method that integrates the authors' original highlights (abstract) and the article's actual impacts on the community (citations), to create comprehensive, hybrid summaries.
- [22] (2020) presents a supervised approach, based on regression techniques, to extract highlights from scientific papers.
- [23] (2020) introduces a new dataset (PPSGen) and uses windows-based approach for text summarization.
- [6] (2021) introduces an interesting summarization pipeline and uses Graph Attention Networks for text summarization, and it is also one of the main references for this project.
  Being [6] the main reference for this work, the description of this project's model in Chapter 3 can be seen as a description of the model proposed by [6]

There are also other works worth mentioning regarding generic text summarization like:

- TextRank [15] and LexRank [16], very famous unsupervised models for automatic text summarization that use a graph-based approach.
- HIBERT (2019) [24] and [25] (2020) that uses a hierarchical approach to adapt transformers to the field of text summarization of large documents.

Graph-based approach: TextRank and LexRank

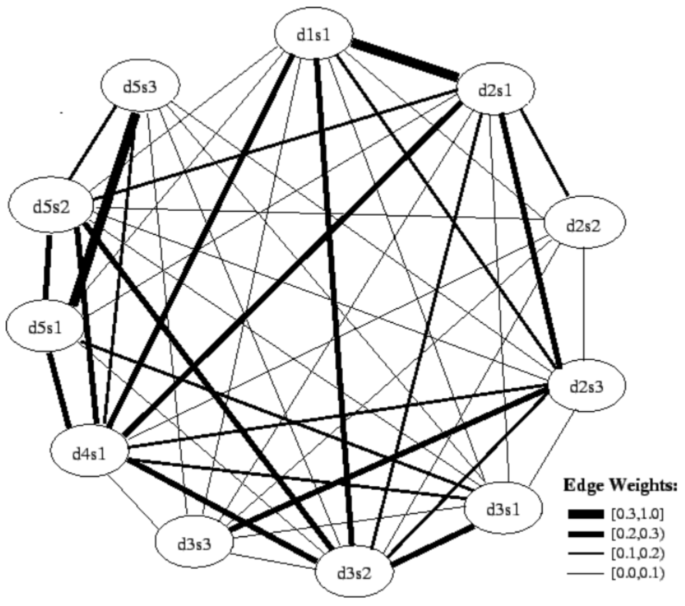


Figure 2.8: LexRank [16]

TextRank and LexRank are two of the most famous models for automatic text summarization.
They are both unsupervised and use a graph-based approach to model the structure of a document where:

- The nodes represent sentences and key phrases.
- The edges connect all the nodes and represent the similarity between sentences.

This kind of approach is similar to the one of PageRank [26], a model by Google, to model web pages and their connections, e.g., the links from one page to another.
The core difference between TextRank and LexRank is the metric used to measure sentence similarity: TextRank uses words co-occurrence while LexRank uses cosine similarity of tf-idf vectors.
Through similarity scores is possible to detect what are the most significant sentences and use them to build a summary of the document. Moreover, in the final part of the project, TextRank and LexRank will be among the models tested as baselines for text summarization.

HIBERT: Hierarchical Bidirectional Transformers for Document Summarization

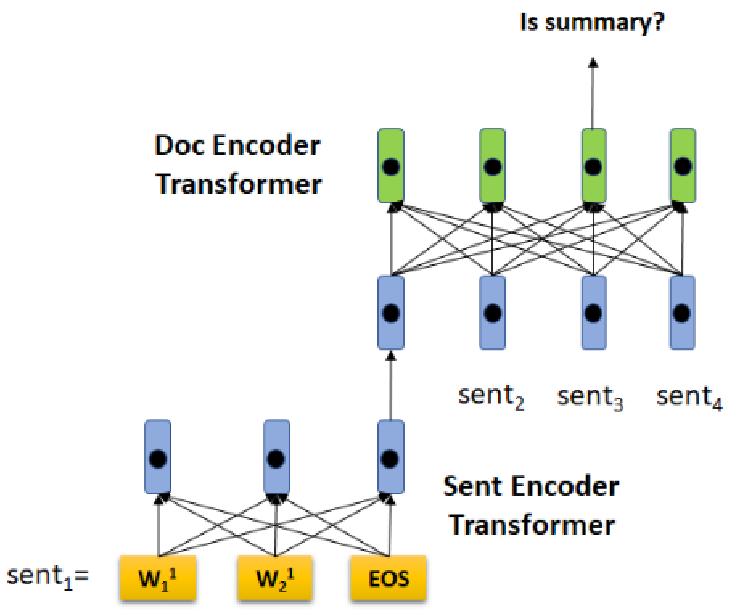


Figure 2.9: HIBERT [24]

HIBERT, and similar works, propose an extension of the standard Transformer's architecture for text summarization of documents.

Standard transformers, like BERT, are bound to a fixed sequence's length like 512 or multiples of 512 and this makes it impossible for a single transformer to process an entire document.
Works like [24] propose a hierarchical structure, like the one in figure 2.9, to allow the use of transformers and attention.
In particular, the first transformer performs sentence embedding given the words of a sentence, then the second transformer performs document embedding using all the sentence embeddings.
This procedure allows the model to learn contextualized representations of sentences and, during test time, to understand if a sentence is relevant enough to be used in a summary.

2.2.2 Slide Generation from scientific papers

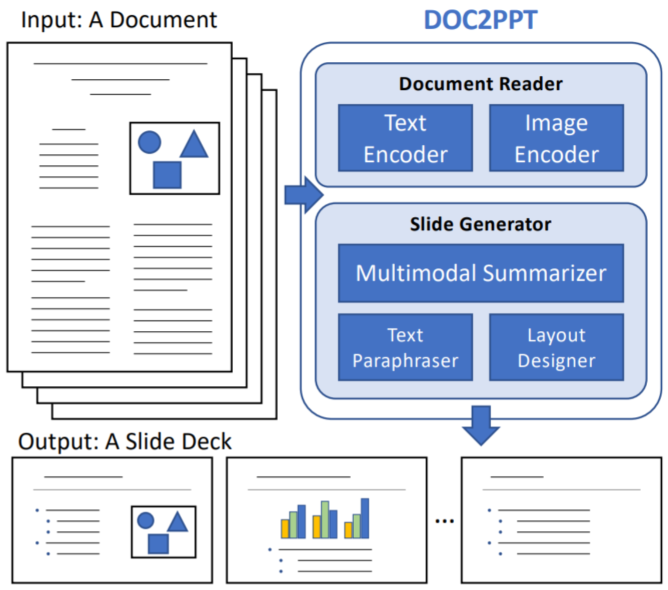


Figure 2.10: DOC2PPT summarization pipeline [1]

Slide Generation from scientific papers is an application of text summarization: from a scientific paper, a machine learning model tries to create a summary that can suit the format of slides.
In literature are present some solution for this task like [27], [28] and [1]. More in detail, [1] is the most recent and effective solution: the

authors presented a model capable of high-quality slide generation and released a huge dataset for everyone who wants to approach this task, in fact, this will be the dataset used in this work.
The authors of [1] use also a hierarchical sequence-to-sequence approach (similar to the one of [24]) and, unlike other mentioned solutions for slide generations, train a module of the model to collocate text and images in a visually pleasing manner, while other works use only text placed following hand-crafted rules.

DOC2PPT

As said before, the model proposed in [1] represents the state of the art in presentation slide generation.
Looking at figure 2.11 it is possible to grasp the complexity of the model.
The model is composed of 4 sub-modules:

- Document Reader (DR): The Document Reader reads both text and images from the source paper, embeds them separately, and then projects them in a common vector space.
- Progress Tracker (PT): The Progress Tracker keeps track of the relationship between documents and slides, i.e, a document can have multiple sections and each section can have multiple corresponding slides that can contain different objects (text or images).
- Object Placer (OP): The Object Placer chooses the object (text or image) from the current section that has the maximum compatibility score with the current state of the slide presentation and places it in a visually pleasing manner, if the object is a phrase it has to be paraphrased by the PAR first.
- Paraphraser (PAR): The Paraphraser uses Seq2Seq (Sentence Embedder based on Bi-RNNs) to make the sentence selected more concise and adequate to the slide format.

During training time every sub-module is trained at the same time using a joint loss function capable of addressing all the sub-modules.

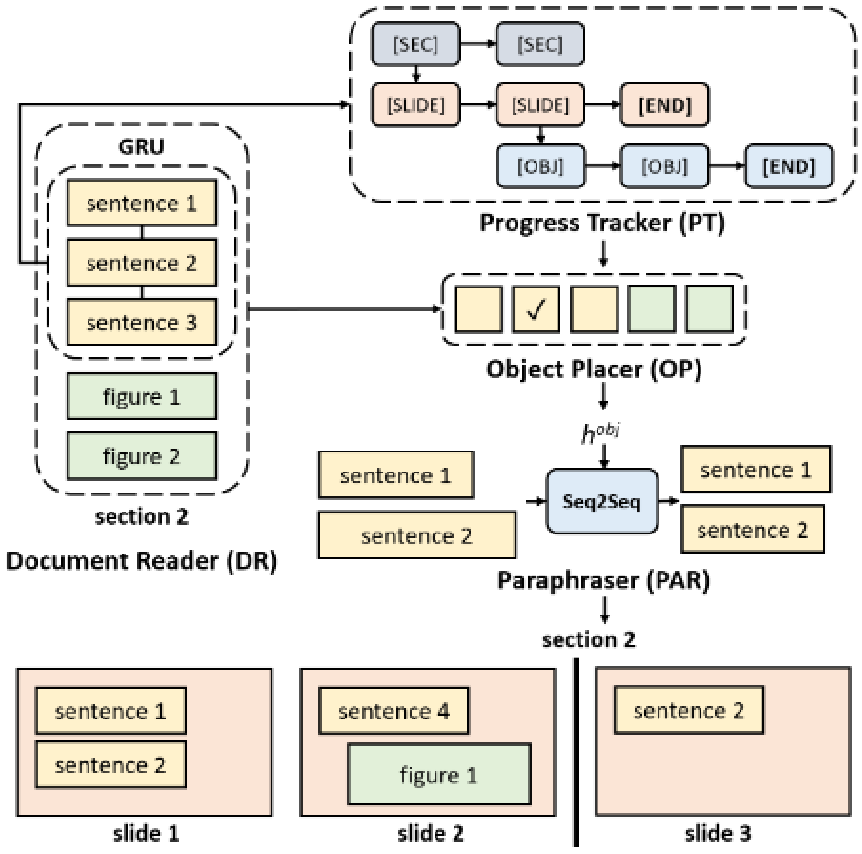


Figure 2.11: DOC2PPT model in detail [1]

It is also worth noticing that the authors of [1] perform a high-quality data processing of the original dataset and this helps to achieve better results (see pages 8-10 from [1]).

SlideGen

Slidegen [27] [28] will be used as a baseline (with other models) for the text summarization task.
The authors of SlideGen create presentation slides using only text summaries condensed in the form of bullet points.

In [28] the authors propose two different models: one extractive and one abstractive and compare them.
The extractive version uses BertSum [29] to select the most n-relevant sentences of the document and then re-arranges them in slides as bullet points.
It is worth mentioning that BertSum has a constraint on the max number of tokens that can be processed at the same time, so, to process all the documents, BertSum was used in a window-like approach (different m sentence of the paper each time).
The abstractive version uses a fine-tuned version of BART [30] (a Transformer) to produce an abstractive summary for each section to put in bullet points to create a slide.
During training time BART is fine-tuned using section/slide pairs from the training set.
The paper proposes an interesting comparison between the two methods and states that the extractive method outputs better summaries in terms of ROUGE scores (see 4.4.1 for explanation) but the abstractive method outputs summaries more similar to slide format, can generate talk overview slides and can learn the language of the slides which is different from scientific text.
SlideGen (in its extractive version) was used as a baseline because shares, in part, the same task of this work, and the availability of the code on GitHub makes it easy to modify and test for the project.

2.2.3 Multimodal Learning

The term Multimodal Learning refers to a type of machine learning method where the model can process different types of information (text, audio, images...) at the same time, the multi-modality can be in input, output, or both.
Multimodal learning is a field with many applications in different scenarios.
In the specific case of document summarization, many interesting works can be found: for example in [31] the authors use Multimodal Learning to produce a summary with text and images of news, in [1] Multimodal

Learning is used to create slide summarization for scientific papers.
In particular, [1] is one of the inspirations for this project due to his specific task.
It's worth noticing that [31] is a paper from 2020 and [1] is from January 2022, this indicates how this specific application of Multimodal Learning is in its early stages.

CLIP

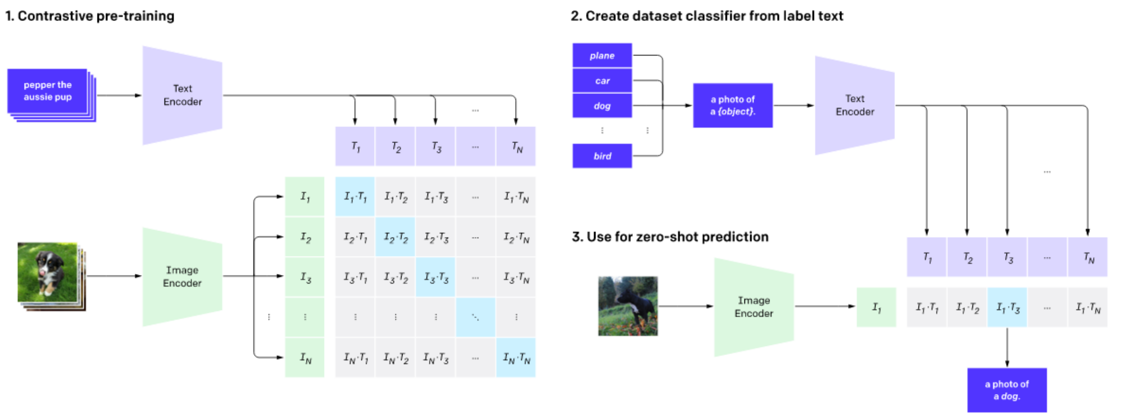


Figure 2.12: Clip model [5]

In this multimodal setting CLIP [5] will cover a crucial role.
CLIP is a pre-trained model available for download on GitHub.
Given a pair of images and captions during training, the model learns both text and image encoding.
This joint learning allows one to classify an image using a caption (e.g. "a photo of a dog" for an image where is present a dog) or retrieve a pertinent image given a specific caption.
This type of training leads to better zero-shot predictions and, more importantly, allows the model to better *understand* the content of an image.
More in detail, during training CLIP, for each image/caption pair, produces an image embedding and a text embedding for the caption, then tries to maximize the cosine similarity between text and images embeddings from the same image/caption pairs and tries to minimize

the cosine similarity between text and images embeddings from different pairs. By doing so, CLIP can understand what image is more pertinent to a caption and vice versa.
For this project CLIP was fine-tuned with images from scientific papers and their caption.
This is because, thanks to fine-tuning, a pre-trained model (like CLIP) can be able to deal with different data from the one seen during pre-training, to do so a model is pre-trained by the author with a huge amount of generic data (for example in the case of CLIP with images/captions of animals, food, random objects) then a user can continue to train to model with fewer data and time than pre-training using task-specific task (in this case images/caption from scientific papers).

2.3 Contribution

As already mentioned, this project draws inspiration mainly from two papers: "DOC2PPT: Automatic Presentation Slides Generation from Scientific Documents" [1] and "Summarizing Long-Form Document with Rich Discourse Information" [6].
This works uses the dataset released with [1] and shares the same task, while [6] is followed (with some modifications) for the summarization phase of the paper.
The main innovation presented by this work can be seen in the use of the summarization pipeline by [6] in the context of slide generation with more extensive use of transformers than [1] and [6].
Both the reference papers rely on GRUs or Convolutional neural networks (CNNS) paired with transformers like BERT or RoBERTa, while this work uses directly Longformer for sentence and section embedding.
It's worth noticing that this work tries also to extend the approach in [6] to a multimodal setting, introducing images in the summarization and slide-generation step.

Chapter 3

Method

Following the structure of the model proposed by [6], the Summarization Model proposed in this work is composed of two main blocks: the *Ranking Module* and the *Extractive Module*.
The Ranking Module selects for each section the most important sentences and the Extractive Module performs an extra step of extractive summarization from these sentences, the output of the Extractive Module is the summary of the paper.

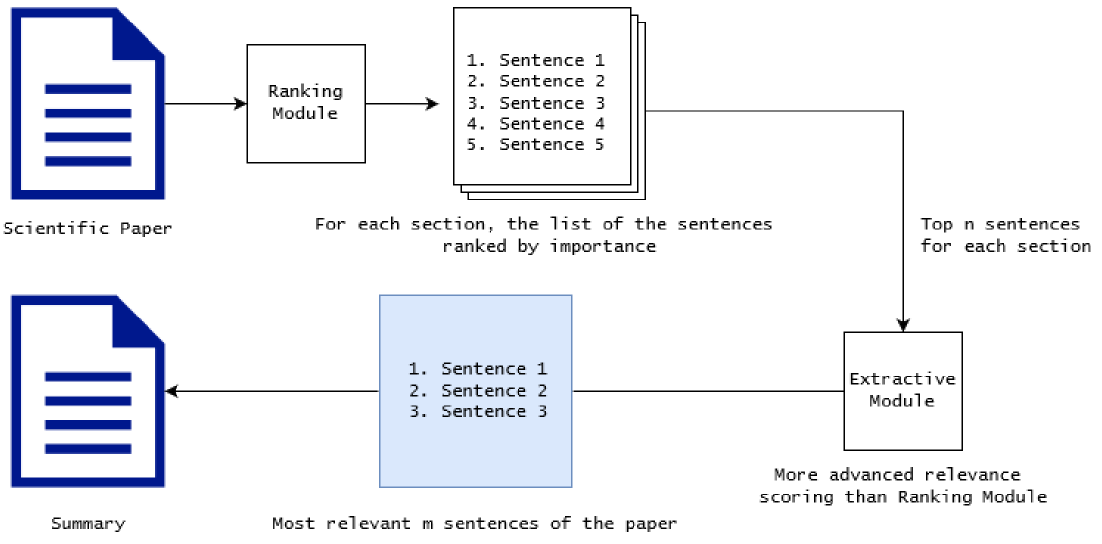

Ranking Module
1. Sentence 1
2. Sentence 2
3. Sentence 3
4. Sentence 4
5. Sentence 5
Scientific Paper
For each section, the list of the sentences ranked by importance
Top n sentences for each section
1. Sentence 1
2. Sentence 2
3. Sentence 3
Extractive Module
More advanced relevance scoring than Ranking Module
Summary
Most relevant m sentences of the paper

Figure 3.1: The Summarization Pipeline

3.1 Ranking Module

The purpose of the Ranking Module is to perform a sort of *filter* for the sentences from the paper: due to the complexity of the Extractive Summarization Module is infeasible to process **all** the sentences from the paper, so the Ranking Module selects the most important sentences for each section. The top-n sentences selected for each section are used as input for the Extractive Module.
The Ranking Module is composed of different sub-modules:

- Title Representation module: this module uses a BiLSTM to obtain contextualized word representation from the title of a paper, then the attention mechanism is used to detect the most important words.

- Content Representation module: a word-level BiLSTM encoder is used to encode words and a word-level Attention layer is used to aggregate word representations into sentence embedding.
  For the context representation (the embedding of a section) the same approach is used starting from the sentence embeddings, a BiLSTM is used to encode the sentence embeddings and an Attention Layer is used to aggregate them into a content representation (one for each sentence).

- Sentence and Section importance: for the importance of the sentences a feed-forward network activation is used. For section importance, first, an Attention Layer fuses the Content Representation and the Title Rconcerningthe output of this Attention Layer is sent through a feed-forward network with sigmoid activation.

To train this module the ground-truth importance of each sentence and section is the average of ROUGE-1/2/L F1 against the golden summary (for ROUGE description see 4.4.1).
The model tries to minimize the Binary Cross Entropy loss between the sentences and sections' importance estimated by the model and the importance from the ground truth.

3.2 Extractive Module

The Extractive Module is a Heterogeneous Graph Attention Network (HGAT for short).
Is in this module that we can see one of the main differences concerning [6]: the authors of [6] use a mixture of CNN and BiLSTM for the initial representation of the nodes of the HGAT (in a similar way to the Ranking Module), in this work instead, two different Longformer versions are used.
For sentence embedding a version with *max_length*=512 is used, for sentence embedding is used a version with *max_length*=4096 (the highest value supported by Longformer), while for word-embedding is still used GloVe, the version with the 300 dimensions output.
A *Projection Layer* is used to project words, sentences, and section embeddings in common feature space with 300 dimensions.
After the projection, all the embeddings are allocated as nodes in an HGAT, for the edges of this HGAT the model suggested by [6] is followed:

- *word-to-sentence edges*: edges from each word to the sentence containing that word.
- *sentence-to-word edges*: the reverse version of word-to-sentence edges.
- *intra-section sentence-to-sentence edges*: edges from a sentence to every other sentence in the same section.
- *cross-section section-to-sentence edges*: for each sentence in a section, edges from every section node to that sentence.
- *intra-section sentence-to-section edges*: edges from a sentence to the section it belongs to.
- *section-to-section edges*: edges from a section to other sections.

The Extractive Summarization Module's complex architecture is capable of better understanding the concepts from the paper and thus performs a better sentence relevance estimation than the Ranking Module. Finally, the most relevant sentences from the paper are selected, put in order as they appear in the paper, and concatenated to form the summary.

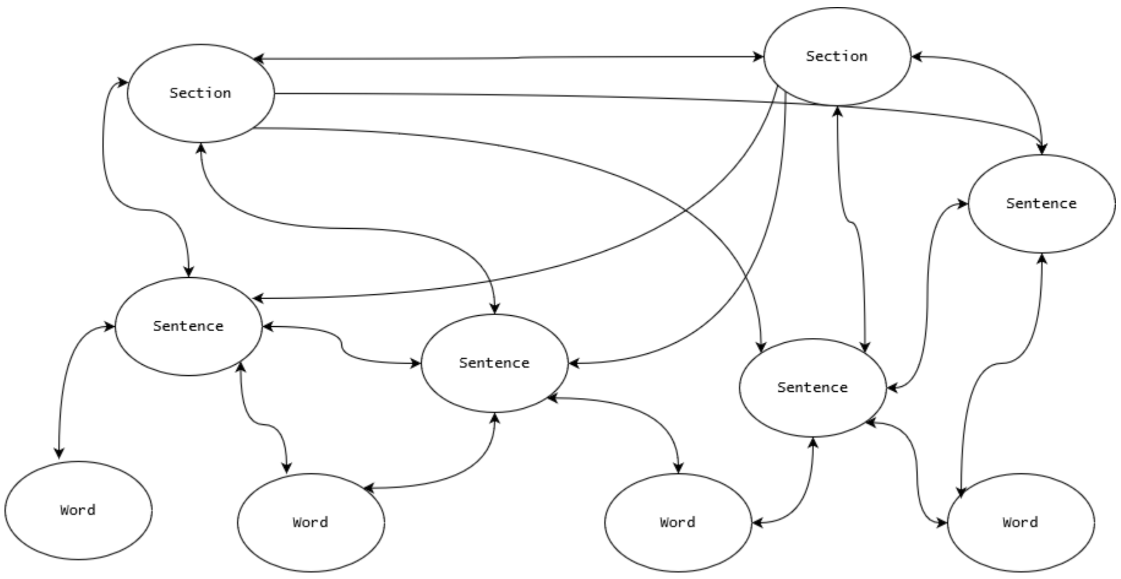


Figure 3.2: Sample from the Graph used in the Extractive Summarization Module

As stated before, two versions of the model were trained for comparisons purposes: one version uses the abstract text as ground truth (like in [6]), so this part of the paper can not be used to build the summary, instead, another version of the model uses the text parsed from the slides as ground truth (like in [1]) and this makes the abstract text usable in the summarization process.

3.3 Multimodality

To extend the Text Summarization pipeline to a Multimodal context, a model like CLIP can be used.
The main idea is to avoid training from scratch a high-complexity model to perform both Text Summarization and Image understanding but instead concatenate two independent state-of-the-art models.
More in detail, in the case of CLIP, is possible to obtain, given in input Text/Image pairs a similarity score of their embeddings, this can be used to understand, from a pool of images from the original paper, which can be more pertinent with a specific sentence from the summary. This is only possible thanks to the fine-tuning done using images/captions from the papers.
Of course, this approach is simpler than the one proposed by DOC2PPT but its high modularity allows for simpler training and easier modifications and updates: for example can be used as a different approach for Text Summarization without changing the multimodal module or, vice-versa, change CLIP in favor of another model without having to modify the Text Summarization pipeline.

Chapter 4

Experimental Results

4.1 Hardware

Train Machine Learning models can be heavy computational work unfeasible in a reasonable time by normal computers.
To solve this problem there exist solutions like cloud computing where high computing power is usable via the internet. For this project was used HPC@POLITO (HPC for sake of brevity).
HPC is a service of *Academic Computing* provided by Politecnico di Torino, upon request, for thesis work, didactic purposes, or research.
The following section will describe the HPC architecture.

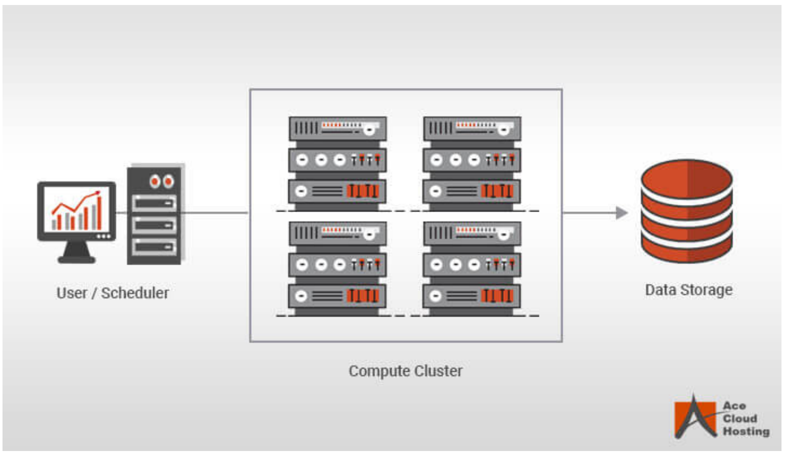


Figure 4.1: Generic HPC system Architecture [32]

4.1.1 HPC@POLITO

The HPC center is composed of 3 cluster that shares both login system and storage: CASPER, HACTAR, and LEGION.
To submit and manage jobs by the users, all 3 clusters use SLURM scheduler.
For this project LEGION, the most recent and powerful one, was used. LEGION is an InfiniBand cluster with a sustained performance of 21.094 TFLOPS (with 14 nodes) and capable of 110.865 TFLOPS at peak performance (57 nodes). For what concerns hardware, LEGION is equipped with:

- CPU: 2x Intel Xeon Scalable Processors Gold 6130 2.10 GHz 16 cores.
- GPU: 4x Nvidia Tesla V100 SXM2 - 32 GB - 5120 Cuda cores (on 6 nodes).
- RAM: 21.888 TB DDR4 REGISTERED ECC.
- Node Interconession: Infiniband EDR 100 Gb/s.
- Service Network: Gigabit Ethernet 1 Gb/s.
- OS: Centos 7.6 - OpenHPC 1.3.8.1.

Each authorized user has their own space that can be used for Python, Bash scripts, Matlab, Blender, and more.
After uploading the necessary files (data sets, scripts, and more), a job can be submitted to the job queue, waiting to be executed according to resource availability and scheduler policies. For each job, a text output file is created for logging.

4.1.2 CUDA

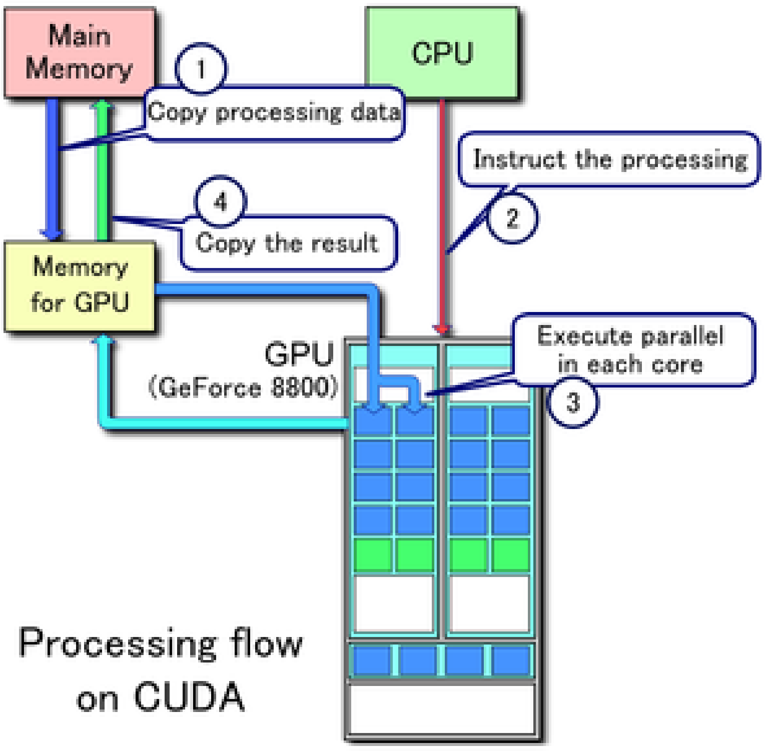


Figure 4.2: CUDA Workflow [33]

CUDA [34] is an API created by Nvidia for general-purpose processing on GPUs.
CUDA, and CUDA-capable GPUs, are nowadays the standard for training huge machine learning models and are also used in various fields like rendering of 3D graphics, encryption, decryption and compression, physical simulations (in particular in fluid dynamics), and much more.
CUDA, and other GPU APIs, became the standard in Machine Learning thanks to the high parallelization introduced by GPU that allows drastically reduced training time for models.
CUDA is designed to work with programming languages such as C and C++ and, thanks to third-party wrappers, it can be used in Python, MATLAB, Java, and others. It's worth noticing that CUDA, unlike other APIs, doesn't require advanced skills in programming.

4.2 Software

4.2.1 Python Libraries

For this project, Python language [35] was used.
Of course, there exist other languages and frameworks for Machine Learning but Python, thanks to its simple syntax and variety of libraries, seems the easiest and most flexible to use.
Moreover, creating a python virtual environment in HPC it's quite easy and the CUDA version of the PyTorch [36] library for Python makes CUDA instantly available.
For this work various libraries are used, this section will describe them and explain how they are used.
One of the most important libraries for this project is PyTorch and its ecosystem.
PyTorch is an open-source End-to-end Machine Learning Framework.
PyTorch comes with APIs to download and use famous datasets, deploy already trained models, create and train custom models, create a dataset from scratch, and much more.
PyTorch is available for Windows, Mac, and Linux systems, for Java/C++ or Python, for CPU-only usage, or with CUDA support.
Thanks to its easy-to-use, variety of tools integrated and active community PyTorch can make testing and development simpler and faster.
Moreover, Pytorch is a building block for another library used in this project: PyTorch Geometric.
PyTorch Geometric [37] is a library built upon PyTorch to easily write and train Graph Neural Networks (GNNs) for a wide range of applications related to structured data. In this project, PyTorch Geometric is necessary to work with Graphs and GANs.
Hugging Face's Transformers [38] is another core library for this work: it provides APIs to easily download and train state-of-the-art pre-trained models.
Hugging Face's Transformers supports seamless integration between three of the most popular deep learning libraries: PyTorch, TensorFlow, and JAX. The Transformers library was used to download the

Longformer model used for sentence embedding.
Spacy [39] and Gensim [40] are the two core libraries more oriented towards words and sentence processing for this work.
Gensim is a library used to train large-scale semantic NLP models and for semantic vector representation of text. Also, Gensim comes with implemented datasets and models, for this project Gensim was used for its GloVe word embedding.
Spacy is a library created for efficient and easy-to-use NLP, it comes with 60+ supported languages, pretrained pipelines, transformers, and word vectors. Spacy alone can be used for NLP tasks such as Named Entity Recognition (NER), text classification, part-of-speech tagging, and more.
Upon Spacy is built scispaCy [41], that contains spaCy models for processing biomedical, scientific or clinical text. For this work scispaCy was deployed only for its implementation of SciBERT.
Concerning the CLIP model, an ad-hoc GitHub library from the authors of the paper was used [42]. Thanks to this library a pre-trained version of the model and a preprocess block are available to download.
For this work, some preprocessing work was done on the dataset provided by the authors of [1], to do so, others python libraries were used: Pandas [43], Numpy [44] and PIL [45] (for image processing), Beautiful Soup [46] and EasyOCR [47].
Pandas and Numpy are two famous standard Python libraries, Beautiful Soup was used to parse XML output from GROBID (more on this subject later) and EasyOCR is a library for optical character recognition (OCR) and was used to parse text from slides.

4.2.2 GROBID

GROBID [48] stands for GeneRation Of BIbliographic Data.
GROBID is a machine learning library for extracting, parsing, and re-structuring raw documents such as PDF into structured XML/TEI encoded documents with a particular focus on technical and scientific publications. In the context of this work GROBID was used to parse the text from PDF files of papers, the output of GROBID comes in the

form of an XML document and to manage this kind of output Beautiful Soup was used.

4.2.3 PDFFigures 2.0

To fine-tune CLIP, an image-caption dataset from scientific papers was necessary.
To create this dataset, PDFFigures [49] was used. Thanks to this software it was possible to parse images from the scientific papers provided by the dataset of [1] and pair them with their caption. The original dataset from [1] has approximately 5,000 papers, parsing them with PDFFigures results in 54,603 figure-caption pairs.
It is worth noticing that also tables in the PDFs are processed like images.
Furthermore, the figures from some papers of the original dataset are absent due to conversion errors.

4.2.4 CLIP fine-tuning

For this work, as mentioned earlier, a new dataset was created and used for CLIP fine-tuning.
CLIP comes as a pre-trained model and, thanks to its architecture can achieve competitive zero-shot performance on a great variety of image classification datasets.
For the fine-tuning process were used 10 epochs with learning rate=1e-5 (learning rate decreasing by a factor of 10 after each epoch), all the hyper-parameters were equal to the ones used for train CLIP: AdamW optimizer with 0.2 weight decay (except for bias layers where no weight decay occurs), $\beta_1 = 0.9$ and $\beta_2 = 0.98$ and $\epsilon = 1e - 6$.

4.3 Dataset

The core dataset for this project is the dataset introduced by [1], it will be referred to as DOC2PPT dataset.
The DOC2PPT dataset includes pairs of papers and their corresponding slide decks from academic proceedings: the papers are in PDF format and the slide decks are JPG files, one for each slide. There are a total of 6,581 scientific papers with their slides.
From the DOC2PPT dataset papers, using PDFFigures, the figures and their caption were extracted, and these pairs were used to create a separate dataset to fine-tune CLIP.
This dataset is composed of 54,603 figure-caption pairs: one JPEG for each figure and a CSV file that contains entries in the form of the name of the file, and caption.
To train the summarization model, all the text from PDF files was extracted using GROBID, the output of GROBID is in the form of XML, to parse XML files Beautiful Soup library for Python was used.
All the parsed papers were divided into 3 separate datasets: train (80%-5,265 papers), validation (10%-657 papers), and test(10%-659 papers).
For each portion of the dataset, a JSON file contains a list of papers with their text divided into a list of sections, each section is then divided into sentences, for each section also the title is reported, and the abstract of a paper is treated as a special stand-alone section (it is separated from the list of sections).

Listing 4.1: The JSON format for each document

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

The text from the slides of the DOC2PPT dataset was used as ground truth for one version of the summarization model, to obtain this text, EasyOCR was used.

For each paper's slide deck, a single txt file contains all the parsed text. Another version of the model uses the abstract text as ground truth instead.
For both the text from papers and text from slides, SciBERT was used to divide all the corpus into sentences.

4.4 Experimental Design

4.4.1 ROUGE Metrics

ROUGE [50] stands for Recall-Oriented Understudy for Gisting Evaluation, and is a set of metrics used for evaluating automatic summarization and machine translation software in natural language processing.
The average ROUGE-1/2/L F1 against the golden summary is used as a metric for the ground truth of a sentence, as a measurement of its importance. Here is a brief explanation of these metrics:

- ROUGE-1: refers to the overlap of unigrams (each word) between a sentence and reference summary.

- ROUGE-2: refers to the overlap of bigrams (two words) between a sentence and reference summary.

- ROUGE-L: Longest Common Subsequence (LCS).

More precisely, ROGUE-n precision is given by the ratio of the number of n-grams in a sentence that appear also in the reference summary over the number of n-grams of the sentence, while ROUGE-n recall is the opposite: the ratio of the number of n-grams in reference that appear also in a sentence over the number of n-grams in the reference. Given ROUGE-n precision and ROUGE-n recall, the ROUGE-n F1 is given by their harmonic mean.

4.4.2 Procedure

In terms of results, the only quantitative measure exploitable for this work is the ROUGE metric to evaluate the quality of the Text Summarization Module (also referred to as the Extractive Model).
To understand how well the Extractive Model performs, the ROUGE scores between the summary produced by the model and the text parsed from slides will be used.
Moreover, the ROUGE scores obtained by the model proposed will be compared with the ROUGE scores obtained by summaries produced by state-of-the-art unsupervised models (TextRank, LexRank, LSA...), by SlideGen [28] and by BertSum [29].
As said before, there exist two versions of the model proposed in this paper: one trained on the abstract text of the paper (like [6]) and one trained on text parsed from slides (like [1]).
Unsupervised models do not require training ad are used in the version provided by the Python library Sumy [51].
For SlideGen two versions were used: one trained with the dataset proposed by the authors of the paper and one trained with the DOC2PPT dataset, using the text parsed from the slides as ground truth.
BertSum was trained using the dataset provided by the author (CNN/-Daily Mail dataset[52], based on summaries from CNN and Daily Mail websites).
All the summaries are truncated to 620 words (620 is the average number of words in the presentation slides from the DOC2PPT dataset).

4.5 Results

4.5.1 Parameters

All the unsupervised methods tested are used in the version proposed by the Sumy library [51], this library uses this configuration of parameters for the models:

- TextRank: epsilon = 1e-4 and damping = 0.85.
- LexRank: threshold = 0.1 and epsilon = 0.1.
- Luhn Summarizer: window size = 4 and significant percentage = 100.
- LSA summarizer: min dimensions = 3 and reduction ratio = 1.

The other unsupervised methods tested (Random, Reduction, and KL Divergence) do not require parameters.
BertSum and SlideGen are trained and tested following the instructions by the authors of the paper from the GitHub repositories, the procedures do not indicate parameters to set during test time so all the parameters used are set in the code by the authors.
BertSum, following the instructions of the authors, was trained using the version BertSum+Classifier setting the dropout = 0.1 and the learning rate = 2e-3.
The training of SlideGen did not require any specific setting.

4.5.2 Baselines Comparison

In general, with exceptions from a few cases, all the scores from the ROUGE metrics are close to each other.
From a strictly numerical point of view, the better model in terms of ROUGE precision is BertSum, while for ROUGE recall is the Luhn's Heuristic Method for text summarization (Luhn for the sake of brevity) [53], for ROUGE-F instead, LexRank [16] is the best for ROUGE-1 F

and ROUGE-L F and Luhn is the best for ROUGE-2 F.
For what concerns our model, the version trained on the slide ground truth performs better than the one trained on the abstract ground truth.
The version of SlideGen that scores better is the version trained on the DOC2PPT dataset, maybe this happens because the data used to train this version is more similar to the data used during test time than the data proposed by the original authors of the model.
Is worth mentioning that, trained both on the DOC2PPT dataset, our model performs better than SlideGen (except for ROUGE-1 P and ROUGE-L P).
Considering two of the better scoring models, LexRank and Luhn, they share some similarities with the model proposed:

- Lexrank uses a graph structure to model the relationship/similarity between sentences.
- Luhn uses tf-idf frequencies to model the importance of a word and the sentence that contains it while the model proposed uses word nodes in the graph structure to model a word and its connections.

This can be an indicator that the graph approach of the model can be the right solution but still needs improvements in architecture and training.

4.5.3 Supervised vs. Unsupervised

The fact that unsupervised models score better than supervised models can be caused by the quality/quantity of the dataset used for training: 6000 papers (and slides) can be too few to train a complex model like the one proposed, moreover, the text parsed from pdf papers and a jpeg of slides could have been of better quality (free solutions were used for parsing).
Another cause can be the absence of an intensive grind search activity, surely more fine-tuned hyperparameters can boost the performances of the model proposed.
As proof of what are the problems faced by the model proposed during

training, BertSum is the only supervised model tested that can score better than LexRank and TextRank in one metric (ROUGE-P), maybe this happens for 2 crucial factors:

- The quality of the Dataset used for training by the authors (CNN/Daily Mail dataset is well known in literature).
- The quality of the training procedure and architecture of the model in terms of hyperparameters.

The fact that is ROUGE-P the metric where BertSum shines can be caused by the type of data used during training: BertSum must provide significant summaries of news, so the output of this model has to be very short and still retain all the crucial information.
Is worth mentioning that, due to the architectural limitation of the model (the intensive use of BERT), BertSum can process only a limited number of tokens, this implies that the summaries produced by BertSum are shorter than the summaries proposed by other models (the average length is of 282 words against 620 by other models).

Table 4.1: ROUGE-1 evaluations

Method	ROUGE-1 R	ROUGE-1 P	ROUGE-1 F
Ours (trained on slide)	39.64	38.90	35.52
Ours (trained on abstract)	38.00	36.48	33.64
SlideGen	36.41	37.88	32.92
SlideGen (trained on doc2ppt slides)	37.50	39.04	33.60
TextRank	38.12	39.80	34.77
Reduction	37.74	39.61	34.49
KL Divergence	32.47	37.33	30.18
Luhn	**40.45**	39.37	35.96
LSA	37.78	37.44	32.91
LexRank	39.43	41.29	**35.97**
Random	36.20	38.40	32.73
BertSum	23.89	**46.66**	28.27

Table 4.2: ROUGE-2 evaluations

Method	ROUGE-2 R	ROUGE-2 P	ROUGE-2 F
Ours (trained on slide)	09.46	09.50	08.56
Ours (trained on abstract)	08.37	08.35	07.56
SlideGen	08.19	08.74	07.46
SlideGen (trained on doc2ppt slides)	08.62	09.25	07.79
TextRank	09.42	10.03	08.66
Reduction	09.33	09.94	08.58
KL Divergence	06.86	08.07	06.44
Luhn	**10.09**	10.03	**09.04**
LSA	08.65	08.88	07.63
LexRank	09.75	10.47	08.99
Random	07.58	08.30	06.97
BertSum	05.47	**11.39**	06.62

Table 4.3: ROUGE-L evaluations

Method	ROUGE-L R	ROUGE-L P	ROUGE-L F
Ours (trained on slide)	15.32	14.36	13.25
Ours (trained on abstract)	14.72	13.49	12.57
SlideGen	14.11	14.34	12.34
SlideGen (trained on doc2ppt slides)	14.45	14.87	12.51
TextRank	15.44	15.50	13.56
Reduction	15.37	15.53	13.53
KL Divergence	13.20	15.36	11.91
Luhn	**15.64**	14.52	13.38
LSA	14.43	14.24	12.14
LexRank	15.63	15.71	**13.71**
Random	13.74	14.57	12.05
BertSum	09.82	**19.47**	11.51

Chapter 5

Conclusion and future work

From the results, is clear that the model proposed has to be improved to become a competitive result in text summarization, this can be done with a better dataset, a proper grid search for training, and architectural changes where required.
Moreover, the model is realized by aggregating specific blocs for each task, this modularity of the model allows its update according to the progress in state-of-the-art solutions.
It is to keep in mind also that the main focus of this work was not to realize a state-of-the-art model but to represent a useful *starting point* for anyone that approaches the problem of Automatic Slide Generation from Scientific Papers and NLP in general, giving an insight on the task and illustrating what can be used to tackle this problem in terms of related works, models, approach, software, and hardware. In the spirit of research and help, moreover, all the code written for each step of this work will be made publicly available on GitHub.
Future research for this project could explore a detailed grid search to optimize the hyperparameters of the model, extend the model with more advanced state-of-the-art solutions in word embedding or sentence embedding, use a more sophisticated HGAT for the Extractive Module, use a different model than CLIP (maybe a multimodal model more suited for scientific papers), and train a module capable of put images

and text in a visually pleasing manner like the one of [1].
In conclusion, this thesis work was beneficial to grasp the problems and difficulties related to NLP and Text Summarization and to understanding that there is still a lot of work to do and plenty of room for improvement in terms of models and solutions proposed, in fact, was the low numbers of publications on Automatic Slide Generation from Scientific Papers to inspire this thesis.
In the hope that all this work will be useful for future research, the author wishes good luck and all the best to anyone that wants to approach the same challenges of this project.

Bibliography

[1] Tsu-Jui Fu, William Yang Wang, Daniel McDuff, and Yale Song. *DOC2PPT: Automatic Presentation Slides Generation from Scientific Documents*. 2022. arXiv: `2101.11796 [cs.CV]` (cit. on pp. 1–3, 17–22, 26, 33–35, 37, 44).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio. «Neural Machine Translation by Jointly Learning to Align and Translate». In: *ArXiv* 1409 (Sept. 2014) (cit. on p. 2).

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep Residual Learning for Image Recognition». In: *CoRR* abs/1512.03385 (2015). arXiv: `1512.03385`. URL: `http://arxiv.org/abs/1512.03385` (cit. on p. 2).

[4] Yinhan Liu et al. «RoBERTa: A Robustly Optimized BERT Pretraining Approach». In: *CoRR* abs/1907.11692 (2019). arXiv: `1907.11692`. URL: `http://arxiv.org/abs/1907.11692` (cit. on pp. 2, 11).

[5] Alec Radford et al. «Learning Transferable Visual Models From Natural Language Supervision». In: *CoRR* abs/2103.00020 (2021). arXiv: `2103.00020`. URL: `https://arxiv.org/abs/2103.00020` (cit. on pp. 2, 21).

[6] Tianyu Zhu, Wen Hua, Jianfeng Qu, and Xiaofang Zhou. «Summarizing Long-Form Document with Rich Discourse Information». In: *Proceedings of the 30th ACM International Conference on Information Knowledge Management*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 2770–2779. ISBN: 9781450384469.

URL: https://doi.org/10.1145/3459637.3482396 (cit. on pp. 3, 13, 14, 22, 23, 25, 26, 37).

[7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. «GloVe: Global Vectors for Word Representation». In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162 (cit. on p. 6).

[8] Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. *English Gigaword Fifth Edition LDC2011T07*. 2011. URL: https://catalog.ldc.upenn.edu/LDC2011T07 (cit. on p. 6).

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. «Attention Is All You Need». In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762 (cit. on pp. 7, 10).

[10] Tom B. Brown et al. «Language Models are Few-Shot Learners». In: *CoRR* abs/2005.14165 (2020). arXiv: 2005.14165. URL: https://arxiv.org/abs/2005.14165 (cit. on p. 9).

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805 (cit. on p. 9).

[12] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. «Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books». In: *CoRR* abs/1506.06724 (2015). arXiv: 1506.06724. URL: http://arxiv.org/abs/1506.06724 (cit. on p. 10).

[13] Iz Beltagy, Arman Cohan, and Kyle Lo. «SciBERT: Pretrained Contextualized Embeddings for Scientific Text». In: *CoRR* abs/1903.10676 (2019). arXiv: 1903.10676. URL: http://arxiv.org/abs/1903.10676 (cit. on p. 11).

[14] MongoDB. *MongoDB as a Graph Database.* URL: `https://www.mongodb.com/databases/mongodb-graph-database` (cit. on p. 12).

[15] Rada Mihalcea and Paul Tarau. «TextRank: Bringing Order into Text». In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 404–411. URL: `https://aclanthology.org/W04-3252` (cit. on pp. 11, 15).

[16] Günes Erkan and Dragomir R. Radev. «LexRank: Graph-based Lexical Centrality as Salience in Text Summarization». In: *CoRR* abs/1109.2128 (2011). arXiv: `1109.2128`. URL: `http://arxiv.org/abs/1109.2128` (cit. on pp. 11, 15, 38).

[17] Elena Baralis, Luca Cagliero, Naeem Mahoto, and Alessandro Fiori. «GraphSum: Discovering correlations among multiple terms for graph-based summarization». In: *Information Sciences* 249 (2013), pp. 96–109. ISSN: 0020-0255. DOI: `https://doi.org/10.1016/j.ins.2013.06.046`. URL: `https://www.sciencedirect.com/science/article/pii/S0020025513004830` (cit. on p. 11).

[18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. *Graph Attention Networks.* 2017. DOI: `10.48550/ARXIV.1710.10903`. URL: `https://arxiv.org/abs/1710.10903` (cit. on p. 12).

[19] Opidi. *A Gentle Introduction to Text Summarization in Machine Learning.* 2019. URL: `https://blog.floydhub.com/gentle-introduction-to-text-summarization-in-machine-learning/` (cit. on p. 13).

[20] Ed Collins, Isabelle Augenstein, and Sebastian Riedel. *A Supervised Approach to Extractive Summarisation of Scientific Papers.* 2017. arXiv: `1706.03946 [cs.CL]` (cit. on p. 14).

[21] Michihiro Yasunaga, Jungo Kasai, Rui Zhang, Alexander R. Fabbri, Irene Li, Dan Friedman, and Dragomir R. Radev. *ScisummNet: A Large Annotated Corpus and Content-Impact Models for Scientific*

Paper Summarization with Citation Networks. 2019. arXiv: `1909.01716 [cs.CL]` (cit. on p. 14).

[22] Luca Cagliero and Moreno La Quatra. «Extracting highlights of scientific articles: A supervised summarization approach». In: *Expert Systems with Applications* 160 (2020), p. 113659. ISSN: 0957-4174. DOI: `https://doi.org/10.1016/j.eswa.2020.113659`. URL: `http://www.sciencedirect.com/science/article/pii/S0957417420304838` (cit. on p. 14).

[23] Athar Sefid, Clyde Lee Giles, and Prasenjit Mitra. *Extractive Summarizer for Scholarly Articles.* 2020. arXiv: `2008.11290 [cs.CL]` (cit. on p. 14).

[24] Xingxing Zhang, Furu Wei, and Ming Zhou. *HIBERT: Document Level Pre-training of Hierarchical Bidirectional Transformers for Document Summarization.* 2019. arXiv: `1905.06566 [cs.CL]` (cit. on pp. 15–18).

[25] Shusheng Xu, Xingxing Zhang, Yi Wu, Furu Wei, and Ming Zhou. *Unsupervised Extractive Summarization by Pre-training Hierarchical Transformers.* 2020. arXiv: `2010.08242 [cs.CL]` (cit. on p. 15).

[26] David F. Gleich. «PageRank beyond the Web». In: *CoRR* abs/1407.5107 (2014). arXiv: `1407.5107`. URL: `http://arxiv.org/abs/1407.5107` (cit. on p. 16).

[27] Athar Sefid, Jian Wu, Prasenjit Mitra, and C. Lee Giles. «Automatic Slide Generation for Scientific Papers». In: *SciKnow@K-CAP*. 2019 (cit. on pp. 17, 19).

[28] Athar Sefid, Prasenjit Mitra, and C. Lee Giles. «SlideGen: an abstractive section-based slide generator for scholarly documents». In: *Proceedings of the 21st ACM Symposium on Document Engineering* (2021) (cit. on pp. 17, 19, 20, 37).

[29] Yang Liu. «Fine-tune BERT for Extractive Summarization». In: *CoRR* abs/1903.10318 (2019). arXiv: `1903.10318`. URL: `http://arxiv.org/abs/1903.10318` (cit. on pp. 20, 37).

[30] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. «BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension». In: *CoRR* abs/1910.13461 (2019). arXiv: `1910.13461`. URL: `http://arxiv.org/abs/1910.13461` (cit. on p. 20).

[31] Junnan Zhu, Yu Zhou, Jiajun Zhang, Haoran Li, Chengqing Zong, and Changliang Li. «Multimodal Summarization with Guidance of Multimodal Reference». In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.05 (Apr. 2020), pp. 9749–9756. DOI: `10.1609/aaai.v34i05.6525`. URL: `https://ojs.aaai.org/index.php/AAAI/article/view/6525` (cit. on pp. 20, 21).

[32] Ace Cloud Hosting Editor. *High-Performance Computing (HPC): All You Need to Know.* 2019. URL: `https://www.acecloudhosting.com/blog/high-performance-computing/` (cit. on p. 29).

[33] Tosaka. *CUDA processing flow.* 2008. URL: `https://commons.wikimedia.org/wiki/File:CUDA_processing_flow_(En).PNG` (cit. on p. 31).

[34] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. *CUDA, release: 10.2.89.* 2020. URL: `https://developer.nvidia.com/cuda-toolkit` (cit. on p. 31).

[35] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual.* Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697 (cit. on p. 32).

[36] Adam Paszke et al. «PyTorch: An Imperative Style, High-Performance Deep Learning Library». In: *Advances in Neural Information Processing Systems 32.* Curran Associates, Inc., 2019, pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf` (cit. on p. 32).

[37] Matthias Fey and Jan E. Lenssen. «Fast Graph Representation Learning with PyTorch Geometric». In: *ICLR Workshop on Representation Learning on Graphs and Manifolds.* 2019 (cit. on p. 32).

[38] Thomas Wolf et al. «HuggingFace's Transformers: State-of-the-art Natural Language Processing». In: *CoRR* abs/1910.03771 (2019). arXiv: `1910.03771`. URL: `http://arxiv.org/abs/1910.03771` (cit. on p. 32).

[39] Matthew Honnibal and Ines Montani. «spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing». To appear. 2017 (cit. on p. 33).

[40] Radim Rehurek and Petr Sojka. «Gensim–python framework for vector space modelling». In: *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic* 3.2 (2011) (cit. on p. 33).

[41] Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. «ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing». In: *Proceedings of the 18th BioNLP Workshop and Shared Task*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 319–327. DOI: `10.18653/v1/W19-5034`. eprint: `arXiv:1902.07669`. URL: `https://www.aclweb.org/anthology/W19-5034` (cit. on p. 33).

[42] OpenAI. «CLIP». In: (). URL: `https://github.com/openai/CLIP` (cit. on p. 33).

[43] Wes McKinney et al. «Data structures for statistical computing in python». In: *Proceedings of the 9th Python in Science Conference*. Vol. 445. Austin, TX. 2010, pp. 51–56 (cit. on p. 33).

[44] Charles R. Harris et al. «Array programming with NumPy». In: *Nature* 585 (2020), pp. 357–362. DOI: `10.1038/s41586-020-2649-2` (cit. on p. 33).

[45] P Umesh. «Image Processing in Python». In: *CSI Communications* 23 (2012) (cit. on p. 33).

[46] Leonard Richardson. «Beautiful soup documentation». In: *April* (2007) (cit. on p. 33).

[47] Jaided AI. «EasyOCR». In: (2020). URL: `https://github.com/JaidedAI/EasyOCR` (cit. on p. 33).

[48] *GROBID*. https://github.com/kermitt2/grobid. 2008–2021. swh: 1:dir:dab86b296e3c3216e2241968f0d63b68e8209d3c (cit. on p. 33).

[49] Christopher Clark and Santosh Divvala. «PDFFigures 2.0: Mining Figures from Research Papers». In: (2016) (cit. on p. 34).

[50] Chin-Yew Lin. «ROUGE: A Package for Automatic Evaluation of Summaries». In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013 (cit. on p. 36).

[51] Miso Belica. *Sumy*. Version 0.9.0. 2021. URL: https://github.com/miso-belica/sumy (cit. on pp. 37, 38).

[52] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. «Teaching Machines to Read and Comprehend». In: *NIPS*. 2015, pp. 1693–1701. URL: http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend (cit. on p. 37).

[53] H. P. Luhn. «The Automatic Creation of Literature Abstracts». In: *IBM Journal of Research and Development* 2.2 (1958), pp. 159–165. DOI: 10.1147/rd.22.0159 (cit. on p. 38).