Universiteit Utrecht
Graduate School of Natural Sciences,
Department of Information and Computing Sciences

Zehao Lu
Student no. 2736888

Unsupervised Paper2Slides Generation

Master's thesis
in ARTIFICIAL INTELLIGENCE

Supervisor:
Dr. Guanyi Chen

Second Reader:
Prof. dr. Albert Gatt

Utrecht, August 2023

Abstract

Although presentations are an excellent medium for sharing academic opinions and ideas, there has been a scarcity of research into automating the "paper to slides" generation task, and a lack of publicly available datasets. In response, we propose an inventive optimization framework based on reconstruction loss, harnessing cutting-edge Large Language Models (LLMs) and unsupervised learning. This approach facilitates the creation of high-quality slide decks from scientific papers, offering heightened adaptability and flexibility. Our evaluation results provide empirical evidence of our model's superior performance in comparison to baseline models.

Contents

1. Introduction . . . 5

2. Related Works . . . 9
- 2.1. Autoencoder Networks . . . 9
- 2.2. Seq2seq Models . . . 10
  - 2.2.1. Recurrent Neural Networks . . . 10
  - 2.2.2. Attention Mechanism . . . 11
  - 2.2.3. Transformers & Large Language Model . . . 13
- 2.3. Seq2seq Decoding Methods . . . 16
  - 2.3.1. Greedy Search . . . 16
  - 2.3.2. Beam Search . . . 17
  - 2.3.3. Top-K Sampling & Top-P Sampling . . . 17
- 2.4. Unsupervised Machine Translation (UMT) . . . 18
  - 2.4.1. Unsupervised Machine Translation . . . 18
- 2.5. Paper-to-slide Generation . . . 19
  - 2.5.1. DOC2PPT . . . 20
  - 2.5.2. D2S . . . 22
  - 2.5.3. Supervised Approach Limitations . . . 23

3. Method . . . 25
- 3.1. Data Processing . . . 26
  - 3.1.1. Pre- & Post-Processing . . . 26
- 3.2. Paper-to-Slide Problem . . . 27
  - 3.2.1. Task Re-Introduction . . . 27
  - 3.2.2. Summarizer-Expander Structure . . . 28
  - 3.2.3. Reconstruction Loss . . . 30
- 3.3. Paper2Slides Model . . . 31
  - 3.3.1. Pre-training S-E Stack . . . 32
  - 3.3.2. Fine-tuning S & E . . . 33
  - 3.3.3. Greedy Optimizer . . . 35

4. Experiments . . . 39
- 4.1. Pre-trained Expander Assessment . . . 39
  - 4.1.1. Experimental Results . . . 39
  - 4.1.2. Examples . . . 41
- 4.2. Fine-tuned Models Assessment . . . 42

4.2.1. Experimental Results 42
4.2.2. Examples 43
4.3. Property of Reconstruction Loss 47
4.3.1. Hypothesis Testing 47
4.4. System Assessment 48
4.4.1. Experimental Results 49

5. Conclusion 51

A. Example Slides Outputs 59

Chapter 1

Introduction

'Researchers spend a great deal of time reading research papers. However, this skill is rarely taught, leading to much wasted effort.' as S. Keshav wrote in 'How to Read a Paper' [Kes07]. Scientific papers can be challenging to read and present, especially for non-experts who may not have a prior understanding of the topic. The widely adopted structure of research articles emphasizes *conciseness* and *precision*, yet simultaneously raises the level of complexity in understanding.

How can a *paper-to-slides* AI model help? A research paper is hard to read because the knowledge embedded in an essay must be comprehensive, objective, and detailed to reach the academic standard. However, this form limits readability. A 'paper-to-slides' model may summarize and reorganize knowledge in a clearer manner (meanwhile probably losing some details) and generate a slide for presentation so that unprofessional readers could have a much easier reading. In contrast to models created for other tasks such as summarization, a model specialized in generating slides from a paper has been fine-tuned to enhance its performance in comprehending the document as a whole, resulting in superior results. Furthermore, utilizing the 'paper-to-slides' model allows for the examination of how language models can be employed to extract information and knowledge.

Although many researchers have proposed various techniques to address the task of transforming the content of research papers into presentation slides, the 'paper-to-slides' problem has not been thoroughly investigated. Earlier attempts at this problem include rule-based approaches by using template matching or keyword queries on the source paper [SII05]. In contrast, more recent research has introduced an innovative approach to this problem by leveraging statistical learning-based methodologies. This involves employing techniques for **extractive** summarization, which aids in the generation of slide content by extracting the most essential and informative portions from the original paper [HW15]. It is noteworthy, however, that the landscape of addressing the 'paper-to-slides' predicament has undergone a significant transformation with the emergence of modern state-of-the-art language models. These cutting-edge models have substantially improved their ability to solve problems by utilizing the computational strength of large neural networks. Some recent notable contributions in this field include Fu et al. [FWMS22], Li et al. [LHML21], Sun et al. [SHW+21]. The majority of these approaches address this task by splitting it into two separate sub-tasks: document

segmentation and text summarization (or bullet point generation). In those systems, each sub-task is solved by one or more sub-models.

Although modern neural network-based models have achieved significant progress on the 'paper-to-slides' task, they are facing a few limits. First, those models are supervisely trained on paper-slides dataset, yet the open and paired data source on this task is very limited, as a result, deploying the model for papers originating from domains beyond its specific training becomes impractical. Besides, the paper-slides datasets currently accessible, lack detailed annotations and labels at the level of individual paragraphs within papers and slide pages. This dearth of granularity leads to an intricate and redundant design of model structure. Additionally, the inherent properties of these datasets introduce substantial noise, ultimately causing the sub-models to be trained using on excessively noisy data.

To address the above issues, we propose a novel model, the Paper2Slides model, which is designed under an optimization framework of reconstruction loss for the 'paper-to-slides' generation task. Under such framework, the proposed model has three important characteristics: First, it performs **unsupervised learning**, which is a type of machine learning method where the model is trained on large amounts of unlabeled data without any explicit supervision. This approach is especially useful in natural language processing, where obtaining labeled data can be a difficult and costly process. By training the model in an unsupervised manner, it is able to identify patterns and relationships within the data without being reliant on predefined labels. This results in a more flexible and robust model that can adapt to different tasks and domains. By leveraging the power of unsupervised learning, the model is able to learn from the vast amounts of available data, making it highly effective in paper-to-slide generation tasks and also avoiding the influence of noisy labeled data on model training.

Moreover, the **optimization framework** unifies the document segmentation and slide content generation tasks. This integration not only leads to a more streamlined and efficient model architecture but also offers significant advantages in terms of simplifying the developpment and deployment. As these tasks are unified, the complex layers and components that would have otherwise been necessary to manage them individually are minimized, resulting in a much more streamlined and easily understandable model design. This significant contribution greatly improves the efficiency of our model and also sets the stage for future researchers to build and release more impactful models based on our foundational architecture.

An additional notable characteristic of the model is its use of modern **Large Language Models** (LLMs). These models are state-of-the-art in natural language processing and have demonstrated remarkable performance on a variety of language tasks. By using an LLM, the model can take advantage of its ability to understand and generate natural language, which can be incredibly useful in many applications [GK18].

In practice, the proposed model would be integrated into a software system capable of:

- Analyze a research paper in (editable) `.pdf` form and extract meaningful content as much as possible;
- **Abstarctively** summarizes the extracted contents (text and figures) and construct slides content;

- Generate a slide explaining the paper accordingly. The slide can be used in a presentation or to help understand the original paper;

To accomplish the outlined objectives, the software is composed of three core elements: a pre-processing unit that reads and loads data from `.pdf` files, the Paper2Slides model responsible for generating slide content, and a post-processing unit in charge of arranging the layout and producing the slides. Each sub-model within the Paper2Slides model has undergone a comprehensive training process, which includes both pre-training and fine-tuning phases. Each step of these procedures is rigorously evaluated. Following this, the entire software undergoes an automated assessment, where it is compared against a state-of-art model in the field, the D2S model [SHW+21]. The proposed solution has demonstrated superior performance compared to this benchmark.

To summarize, we defined a new framework to tackle the paper-to-slides generation task, designed a model under the framework, and evaluated our approach. Besides the practical value of this study, this research will also investigate a new approach to scientific document understanding. The primary technique used in this project will be NLP/NLG theory including text summarization and text generation. No multi-model techniques will be used in this study, as we don't plan to make any use of image information in the document.

In the rest of this paper, we will proceed by first conducting a review of related research. This will involve introducing various neural network structures, ranging from the Autoencoder network to Transformer-based language models. Following this, we will detail a few widely used decoding methods employed for language generation. Furthermore, we will explore the realm of unsupervised machine translation and the pertinent training methods we adopt. Our attention will then shift to the state-of-the-art works in the domain of paper-to-slides conversion, where we will also address the limitations inherent in these approaches. Further information regarding the model will be presented in section 3. We will firstly dissect the intricacies of pre- and post-processing, followed by a formal articulation of the optimization framework of the reconstruction loss. Subsequently, we will present the implementation and architecture of the Paper2Slides model, coupled with an exposition of the training algorithms for its sub-models. The detailed experimental details can be found in section 4, wherein we showcase the evaluation outcomes of both pre-trained and fine-tuned sub-models within the Paper2Slides model. Additionally, we present an assessment of our system's performance in comparison to a state-of-the-art model.

Chapter 2

Related Works

We will give a series of literature reviews on several related topics in this section. The topics discussed in this section are either related to one of the sub-tasks mentioned previously or used techniques in this project.

Firstly, we briefly introduce the autoencoder neural network which inspired the reconstruction loss framework. Then, we'll provide an overview of the seq2seq model utilized in the language domain. Next, we will introduce several commonly used algorithms for inference and generation within the framework of the seq2seq model, followed by a subsection on unsupervised machine translation techniques that influenced the model architecture implemented in this project. Finally, we'll provide a concise assessment of the present state-of-the-art results of the paper2slide generation task.

2.1. Autoencoder Networks

An autoencoder network, a type of artificial neural network, excels in unsupervised learning by reconstructing its input, yielding efficient data representations via an encoding-decoding process through a reduced-dimensional vector space [Kra91, Ben09, JHG00]. The autoencoder model maps input data onto a lower-dimensional vector space through an encoding process, the lower-dimensional intermediate representation is then subsequently reconstructed back into the original input space through a decoding process. The network is composed of two main components: an encoder network E, responsible for transforming the input data into a compressed representation, and a decoder network D, tasked with reconstructing the original data from the encoded representation. Given the input vector $\vec{x}$, the model is trained to minimize the reconstruction loss between the input $\vec{x}$ and decoded data $\vec{y}$.

$$\begin{aligned} \vec{r} &= E(\vec{x}), \text{where } \dim(\vec{r}) < \dim(\vec{x}) \\ \vec{y} &= D(\vec{r}) \end{aligned} \tag{2.1}$$

Autoencoders have found applications in diverse fields such as feature learning, data compression, denoising, and anomaly detection. The following section will demonstrate how the integration of autoencoder principles into RNN-based Encoder-Decoder models strengthens sequence-to-sequence modeling by capitalizing on their collective advantages.

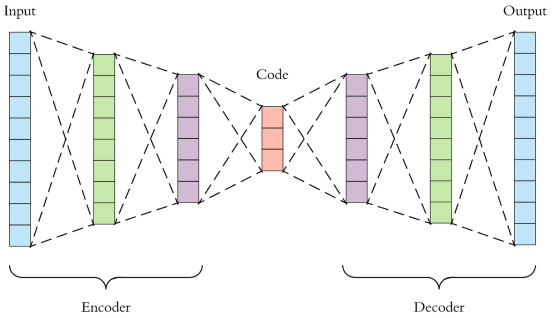


Figure 2.1: Autoencoder neural network.

2.2. Seq2seq Models

A sequence-to-sequence (seq2seq) model is a type of model in machine learning that is used for many tasks in natural language processing and time series processing areas [SVL14]. A seq2seq Model takes sequential data as inputs and maps it to the sequential outputs. The seq2seq model has been widely used in natural language generation areas, some of the most common applications of the seq2seq model include text summarization [KKM+19], machine translation, and chatbot. The seq2seq models are usually trained on input-output pairs, wherein both the input and outputs are sequences of tokens that may have differing lengths. The typical architecture of a seq2seq model includes an Encoder and a Decoder, with the Encoder being used for processing the input sequence and the Decoder for generating the target sequence. The seq2seq model learns to predict the target output token sequenced by minimizing the loss function. The Encoders and Decoders of a seq2seq model are commonly implemented using either Recurrent Neural Networks (RNNs) or Transformers [SVL14].

2.2.1. Recurrent Neural Networks

A recurrent neural network is a neural network that can handle variable lengths of inputs. A typical vanilla RNN model has an input layer I, a hidden layer H, and an output layer O. At each step t, the output of $h_{t'.-1}$ from the previous step is recurrently fed back into the hidden layer to generate the model's output. Given the input sequence $\{x_t\}$ and $\{y_t\}$, the RNN processing function can be expressed in this form

$$\begin{aligned} i_t &= I(x_t; \cdot) \\ h_t &= H(i_t, h_{t-1}; \cdot) \\ y_t &= O(h_t; \cdot) \end{aligned} \tag{2.2}$$

Where the middle dot symbol $\cdot$ represents the linear weight, bias, and activations that might be used in those layers. By doing so, RNN gains the ability to use global information of the sequential data by memorizing the entire input sequence.

While the vanilla RNN unit is rarely used in real-world tasks, its variation like GRU [CGCB14, CvMG+14] and LSTM [HS97] units are rather popular. The gated and gap-connected units in GRU/LSTM allow RNN to separately learn short-term and long-term memories, which enables the model to figure out what should be forgotten

and what remembered and looped back into the network. Despite its prowess in seq2seq tasks, the RNN model has been found to encounter several issues such as vanishing gradient and exploding gradient, low computational efficiency, and its weak ability to memorize longer-term dependency.

RNN Encoder-Decoder. A famous seq2seq model based on RNN structure is the RNN Encoder-Decoder model, The Encoder and Decoder of the model are both RNNs, the Encoder of the model is composed of the input layer and hidden layer togetherly, the output of the hidden layer generates the weight value c,

$$\begin{aligned} h_t &= f(x_t, h_{t-1}; \cdot) \\ c &= q(h_1, \cdots, h_T, x; \cdot) \end{aligned} \tag{2.3}$$

where f and q are both non-linear functions.

The Decoder is formed by the hidden layer and the output layer, where the Decoder's hidden layer takes both the output for the last step and the Encoder's hidden layer's output as input. The Decoder is designed to maximize the overall probability of the target output sequence,

$$p(y) = \sum_t p(y_t | y_1, \cdots, y_{t-1}, c) = \sum_t g(y_{t-1}, s_t, c) \tag{2.4}$$

where function g is similar to f, and s_t is the hidden state of the Decoder RNN.

2.2.2. Attention Mechanism

To address the above issue, some researchers proposed the *alignment model* with bidirectional RNN [BCB16]. The alignment model is later called the attention mechanism to indicate that it executes the mechanism of determining which token should receive focus.

The structure of the new RNN Encoder-Decoder model with alignment can be described as follows:

$$\begin{aligned} p(y_t | y_1, \cdots, y_{t-1}, c) &= \sum_t g(y_{t-1}, s_t, c) \\ s_i &= f'(s_{i-1}, y_{i-1}, c_i) \end{aligned} \tag{2.5}$$

where s_t is the hidden state of the decoder model, same as equation 2.4. The context vector, denoted as c, is now solely dependent on the hidden state h of the Encoder model, and it encapsulates information pertaining to the complete input sequence. Specifically, the context vector is obtained as a weighted sum of h,

$$\begin{aligned} c_i &= \sum_j \alpha_{ij} h_j \\ \alpha_{ij} &= \frac{exp(e_{ij})}{\sum_k exp(e_{ik})} \\ e_{ij} &= a(s_{i-1}, h_j) \end{aligned} \tag{2.6}$$

with the cross attention weight α_{ij} is calculated through the alignment model, as a normalized correlation measure of the hidden state from the Encoder model and the Decoder model. In summary, the attention unit generates attention values for the decoder

at each time step, incorporating input from the preceding decoder step. This allows the attention unit to capture the overall dependencies necessary to predict y_t at step t.

After experiencing the power of the Attention unit, researchers soonly found that the attention unit does not have to be a component of the RNN Encoder-Decoder model. The Scaled Dot-Product Attention, a simplified attention mechanism, has demonstrated impressive efficacy in language generation and gained widespread popularity [VSP+17].

The scaled dot-product Attention unit is designed to compute the attention weight by computing the dot product between the query and the key vector, and the self-attention weight is used to determine which part of the input sequence is essential for making a more likely prediction. The computing process of the scaled dot-product Attention can be described as follows: firstly, the input vectors are added to the positional embedding vectors in order to remember the order of the input tokens. After that, the scaled dot-product Attention block takes the entire input sequence and multiplies them with three distinct matrices W_Q, W_K, and W_V. The results are denoted by Q, K, and V.

$$\begin{aligned} Q &= W_Q \cdot x^T \\ K &= W_K \cdot x^T \\ V &= W_V \cdot x^T \end{aligned} \tag{2.7}$$

Then, the attention values are computed by applying the dot-product on Q and K. The results are divided by the square root of the dimension of the key vector d_K, and multiplied by the value matrix V, as shown below,

$$Attention(Q, K, V) = softmax(\frac{Q \cdot K^T}{\sqrt{d_K}})V \tag{2.8}$$

A variation of the vanilla Attention is the multi-head Attention mechanism. With the multi-head Attention unit, the model conducts a multiple set of Q, K, and Vs in order to explore extensive "representation subspaces" within the semantic space. This enables the model to have multiple choices of the attention value when making predictions of the output sequence. Denote the Attention value of a single head as Z_i, it is computed by the same mechanism as a single head Attention,

$$\begin{aligned} Q_i &= W_Q^i \cdot x^T \\ K_i &= W_K^i \cdot x^T \\ V_i &= W_V^i \cdot x^T \\ Z_i &= softmax(\frac{Q_i \cdot K_i^T}{\sqrt{d_K}})V_i \end{aligned} \tag{2.9}$$

Once all Attention value $\{Z_i\}$ are computed, the Z_is will be concatenated and multiplied by an additional weight matrix W_O,

$$Z = W_O \cdot concate_i(Z_i) \tag{2.10}$$

where $concate_i$ denotes concatenation of all Z_i.

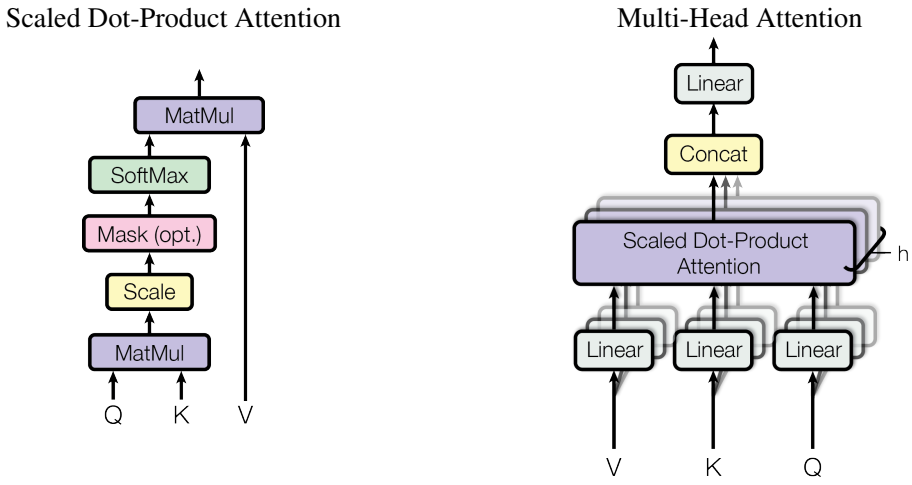


Figure 2.2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel [VSP+17].

One of the main benefits of the Attention mechanism is its computational efficiency, compared to its original RNN Encoder-Decoder version. Meanwhile, it hinders the occurrence of the vanishing gradient problem in the model. The utilization of the attention mechanism has significantly enhanced the effectiveness and precision of language generation and understanding. This technique has been incorporated into various language models such as BERT, GPT, and other Transformer-base models.

2.2.3. Transformers & Large Language Model

The Transformer model/block is a neural network structure that is purely based on the Attention unit and residual connection. The Transformer block is first proposed in the paper *Attention is all you need* [VSP+17]. As it is shown in Figure 2.3, the Transformer model has two components: the Encoder and Decoder. The Transformer Encoder consists of a stack of identical Transformer blocks, with each Transformer block having two components including the Multi-head self Attention unit and a feed-forward unit. Both of the layers are paralleled with a residual connection and are followed by a normalization layer. Before the input representation (usually the embedding vectors) is fed into the Encoder, the representation is added to the positional embedding in order to give the model some knowledge about the position of the input vectors.

The Decoder is composed of several Decoder Transformer blocks, the number of blocks is usually equal to the number of Encoder Transformers. Each Decoder transformer block has two Attention units and a feed-forward unit, both of the Attention units are connected paralleled with a residual connection. The output of the Encoder is transformed into a set of Attention vectors K and V and iteratively used by the second Attention unit of these Transformer blocks. The procedure can be described by the following formulas. Given the input sequence $X = \{x_i\}$, output sequence $Y = \{y_i\}$, and positional encoding tensor $\vec{pos}$, a Encoder model with N Transformer blocks performs

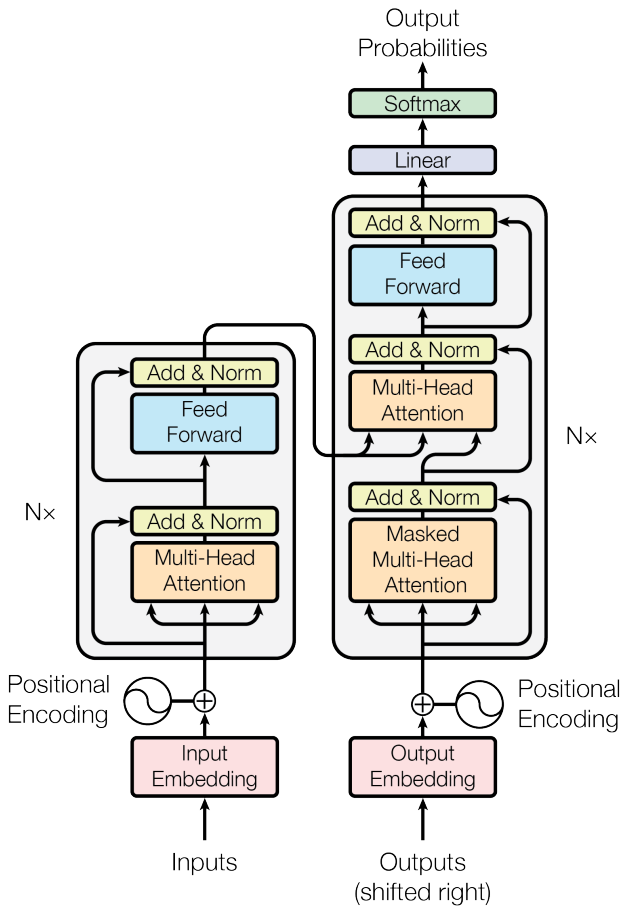


Figure 2.3: The Transformer Model structure [VSP+17].

the following algorithm,

$$
\begin{aligned}
& X = X + p\vec{o}s \\
& \hat{X}_0 = \mathrm{norm}(\mathrm{Att}(X) + X) && \text{Attention Unit} \\
& X_0 = \mathrm{norm}(\mathrm{FF}(\hat{X}_0) + \hat{X}_0) && \text{Feed Forward Unit} \\
N-2 \ \Bigg\{ & \quad \vdots && \text{Repeat} \\
& \hat{X}_N = \mathrm{norm}(\mathrm{Att}(X_{N-1}) + X_{N-1}) \\
& X_N = \mathrm{norm}(\mathrm{FF}(\hat{X}_N) + \hat{X}_N)
\end{aligned}
\tag{2.11}
$$

where $\mathrm{Att}(\cdot)$ and $\mathrm{FF}(\cdot)$ denote the Attention Unit and the Feed Forward Unit respectively. Similarly, a Transformer Decoder step can be represented as follows.

$$
\begin{aligned}
& Q, K = \mathrm{transform}(X) \\
& Y = \mathrm{mask}(Y) + p\vec{o}s \\
& \hat{Y}_0 = \mathrm{norm}(\mathrm{Att}(Y) + Y) && \text{Self Attention Unit} \\
& \hat{Y}_0 = \mathrm{norm}(\mathrm{Att}(Q, K, \hat{Y}_0) + \hat{Y}_0) && \text{Encoder-Decoder Attention Unit} \\
& Y_0 = \mathrm{norm}(\mathrm{FF}(\hat{Y}_0) + \hat{Y}_0) && \text{Feed Forward Unit} \\
N-2 \ \Bigg\{ & \quad \vdots && \text{Repeat} \\
& \hat{Y}_N = \mathrm{norm}(\mathrm{Att}(Y_{N-1}) + Y_{N-1}) \\
& \hat{X}_N = \mathrm{norm}(\mathrm{Att}(Q, K, \hat{Y}_N) + \hat{Y}_N) \\
& X_N = \mathrm{norm}(\mathrm{FF}(\hat{X}_N) + \hat{X}_N)
\end{aligned}
\tag{2.12}
$$

Transformer-based models have gained immense popularity and have quickly become the dominant force in the field of NLP. Many cutting-edge models have been developed to address specific tasks such as sentiment classification and text summarization, using supervised learning techniques. However, supervised models usually require a large amount of annotated datasets and have limited generalized ability. Therefore, large pre-trained language models (LLM) are proposed to solve these two issues. One of the most well-known and popular LLMs is the Generative Pre-trained Transformer (GPT) family model [BMR+20]. As a Decoder-only model, the GPT model is composed of a stack of Decoder Transformers.

The GPT models are pre-trained on vast amounts of text data and can generate coherent responses based on the prompts. To mitigate the potential of the model replicating input sequences, data masking techniques are employed during the training process.

In addition to the GPT models (GPT1 - GPT4) [BMR+20], the BERT model developed by Google has gained significant attention [DCLT19]. Due to its Encoder-only architecture, the BERT model is limited to natural language understanding tasks such as sentiment analysis and named entity recognition, and cannot be applied to NLG tasks. BERT excels at extracting information and insights from text, whereas GPT's strength

lies in generating new and creative text. Other notable recent language models include T5 (Encoder-Decoder) [RSR+20] by Google, LLaMa (Decoder-only) by Meta [TLI+23], and LaMDA (Decoder-only) [TFH+22] also developed by Google.

In this study, we will use the Bidirectional and Auto-Regressive Transformer [LLG+19], short for BART, as the pre-trained model for summarization. One of the key features of BART is its ability to generate text in both forward and backward directions, which is achieved by training the model on both auto-regressive and denoising tasks. This means that BART is capable of both predicting the next word in a sequence and reconstructing the original sequence from a corrupted version. Similar to GPT, BART is pre-trained on a large corpus of text using a combination of masked language modeling and denoising auto-encoding tasks.

2.3. Seq2seq Decoding Methods

With the popularity of the seq2seq model, various decoding strategies have also received more attention. In this section, we will give a brief discussion of the most widely used generation methods of the seq2seq models. We start by refreshing the notation and definitions of the auto-regressive language generation. The auto-regressive language generation is a language model that works by predicting the probability of the next token based on the previous tokens in the input sequence. At each step, the predicted tokens (probabilities) will be iteratively used to predict the next token.

$$P(w_{1:T}|W_0) = \prod_T P(w_t|w_{1:t-1}, W_0), \text{with } w_{1:0} = \emptyset \tag{2.13}$$

where W_0 is the entire input sequence.

We will introduce the most popular decoding methods at present, mainly Greedy search, Beam search, Top-K sampling, and Top-p sampling [HBFC19].

2.3.1. Greedy Search

The Greedy search is the simplest method among the decoding methods. Similar to the Maximum Likelihood Method (MLE), the greedy search algorithm selects the most likely token as the next word at each step.

$$w_T = argmax_w(P(w|w_{1:T-1}) \tag{2.14}$$

Some of the drawbacks of this method are:

- It may not be able to find the global optima, since it does not consider other options that could result in better outcomes;
- It can be prone to errors because one wrong choice can influence the rest of the generation and lower the output quality;

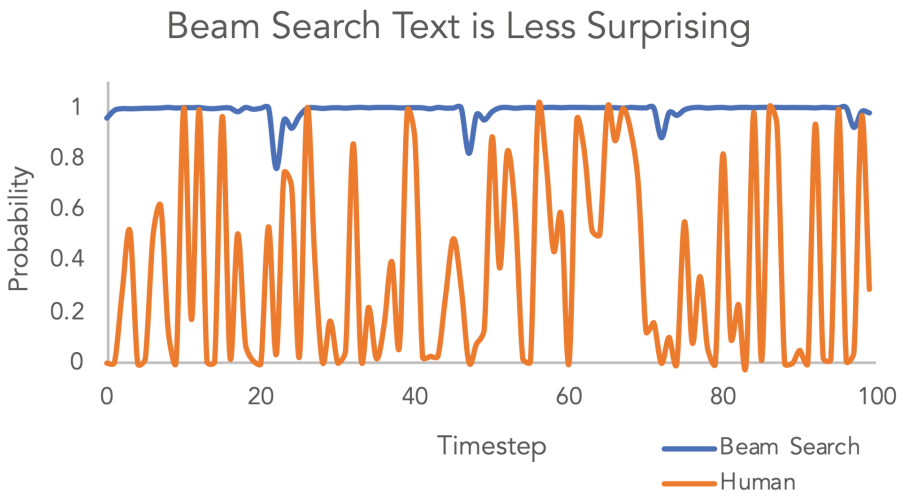


Figure 2.4: The probability assigned to the next token by beam search versus by humans. The human-selected tokens have much higher variance in predicted probability and are thus more surprising [HBD+20].

2.3.2. Beam Search

Beam search addresses the above issue by involving the memory mechanism in the generation procedure. Given the beam number n_{beam}, The Beam search would save the most likely n_{beam} next tokens in memory, and append them onto the current sequence candidates. The beam search will use the stored memory to build a search tree and then produce the best possible sequence as the output. A common issue with the beam search result is that it might repeat itself and run into an endless loop. And using the n-gram penalty in beam search is a way to avoid that issue [KKD+17] [PXS17]. An n-gram is a sequence of n words or symbols, like "apple tree" (2-gram) or "wake up early" (3-gram). An n-gram penalty is a parameter that penalizes the score of a candidate sequence if it contains repeated n-grams. The purpose of using the n-gram penalty in beam search is to reduce the likelihood of generating redundant or unnatural sequences. A drawback of the Beam search technique is that the Beam search output is too regular and does not resemble natural language [HBD+20]. A clear evidence is shown in Figure 2.4 from [HBD+20].

2.3.3. Top-K Sampling & Top-P Sampling

The Top-K or Top-P Sampling means introduces more randomness into the generation procedure. Unlike the search strategies, the sampling methods would arbitrarily select from a set of most likely next tokens. The Top-K Sampling sort the possible next tokens by their probability and performs random selection. By doing this, the words with low probabilities in the distribution's tail are discarded and have no possibility of being picked. On the other hand, unlike Top-K sampling, Top-p sampling (or nucleus sampling) does not constrain the number of tokens to consider but picks from the smallest

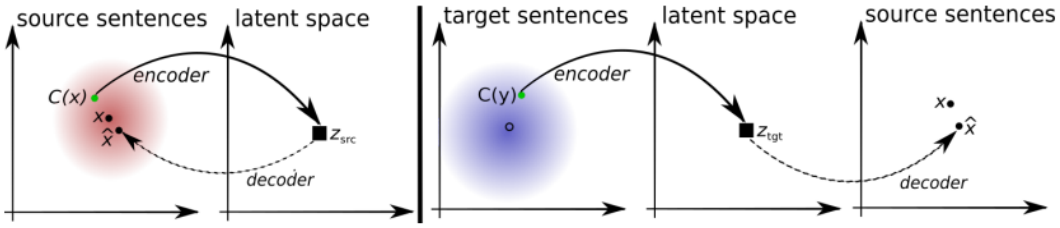


Figure 2.5: Unsupervised machine translation.

set with a cumulative probability higher than p.

2.4. Unsupervised Machine Translation (UMT)

We briefly introduce some recent progress in the field of unsupervised machine translation, in order to provide a thorough explanation of our inspirations. As an old and classic research topic in the NLG area, machine translation has drawn much attention and has been put under thorough investigation. Many models require a substantial quantity of parallel data, particularly in the form of parallel sentences, to achieve their best performance. As a result, several researchers concentrate on unsupervised techniques, due to the lack of paralleled data. The two papers published by the Meta (formerly Facebook) AI team [LCDR18, LOC+18] are among the most impressive unsupervised learning works, and their model structure and ideas have profoundly inspired me in this project.

2.4.1. Unsupervised Machine Translation

In their first work [LCDR18] on this topic, they proposed the Neural Unsupervised Machine Translation Model. The model leverages the seq2seq language models and the back translation mechanism. As shown in Figure 2.5, the translation model consists of two components, the Encoder and the Decoder, the Encoder maps the input sequence (from the source language) to the latent space and the Decoder maps from the latent space to the target language, the back translation model works vice versa. The translation models are trained to translate by reconstructing both languages while matching the latent space.

In their second paper [LOC+18], the authors improved the model by proposing an unsupervised strategy and also highlighting the three essential elements that a successful unsupervised translation model should have:

- Initial bilingual dictionary or phrase-based translator;
- Leveraging strong language models;
- Turning the unsupervised problem into a supervised problem by automatically performing the back-translation procedure;

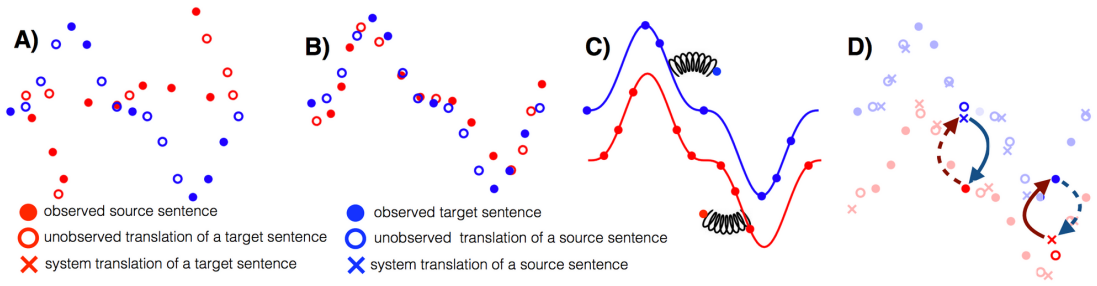


Figure 2.6: The three principles of the UMT.

The authors followed the above three principles and proposed their second research. Their model consists of a $< source - target >$ translator and a $< target - source >$ back-translator, as shown in Figure 2.6.

The MT system first aligns the unparalleled monolingual datasets by applying the word level or phrase-based translation. The dictionary or phrase-based translation is considered the initial translation model. Then a language model for each language is learned independently on its corpus, the language model is used later for reformatting and denoise the translated data. The third step is to iteratively perform the translation and back translation based on the generated $< source - target >$ pairs, and use the language model on monolingual data to improve the quality of generation. The translation model is a seq2seq model that has an entire encoder-decoder structure. In order to match the semantic space of both languages, the encoder/decoder representation is shared across the source and target languages. The loss function for each translation model is defined as the reconstruction of the whole procedure: $source \rightarrow target \rightarrow source$ and $target \rightarrow source \rightarrow target$. Specifically, the loss function is defined as the expectation of the probability of successfully reconstructing the *natural* sentences from their translated result conducted by the translation models.

2.5. Paper-to-slide Generation

The challenge of producing slides to accompany academic papers has not been thoroughly investigated and poses a significant difficulty. Earlier researcher mainly focuses on rule-based methods, their works focus on selecting the most important sentences and re-organize them. However, this approach highly relies on a set of predefined rules or heuristics that are based on human expertise or domain knowledge. To address that issue, some statistical learning-based methods are proposed. The majority of those statistical learning algorithms first compute the importance of sentences and create a set of heuristics that can be used to group these sentences together as the output slides. The drawbacks of statistical learning approaches are that they could only produce extractive summarization instead of abstractive summarization [HW15]. More recent approaches such as the D2S model [SHW+21] and the DOC2PPT model [FWMS22] are using neural networks on both sentence selection and summarization tasks. The DOC2PPT model used a gated recurrent neural network (GRU) to group the sentence and a sentence-level

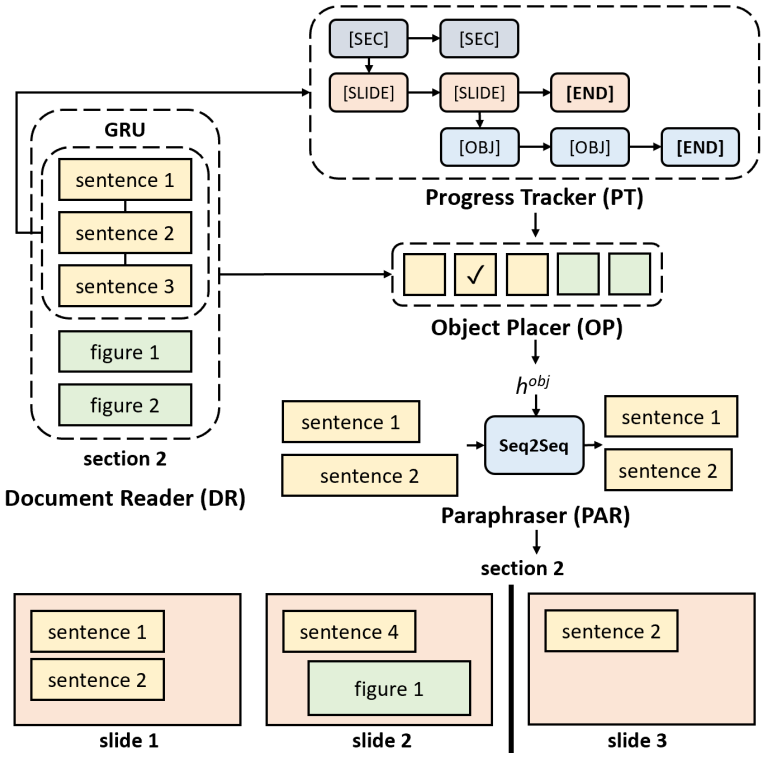


Figure 2.7: An overview of the DOC2PPT architecture.

summarizer to generate the bullet points. On the other hand, D2S uses a *keywords module* to select information and a *Question Answer module* to generate the slide contents. [SHW+21] [FWMS22]

2.5.1. DOC2PPT

We have drawn inspiration from the DOC2PPT model in our research, and as such, I will provide a brief overview of its model structure. The DOC2PPT model is composed of multiple blocks, including the document reader, progress tracker, object placer, and paraphraser. The progress tracker and paraphraser blocks are particularly essential in facilitating document comprehension. The document reader block utilizes a GRU seq2seq model to accommodate any input length, each input sentence is passed to the GRU model to encode the tokens into vectors. The progress tracker block track and evaluates each sentence in the input document to determine its suitability for inclusion on a new slide. The object placer block assigns labels to each sentence to determine its appropriate position on a slide. Finally, the paraphraser block takes the selected sentence and condenses it into a more concise form before placing it on the slide. The author of the DOC2PPT model evaluates the model's performance in two aspects: structural loss and content loss. The structure loss is computed using cross-entropy loss of content assignment actions (page turning and bullet point assignment). The content loss is

the sum of selection loss (whether the keywords or important sentences are selected for slide generation), Paraphrasing loss (cross-entropy loss of the word-word comparison of the generated language and the ground truth), Text-Figure matching loss, and Layout loss. The latter two terms are evaluating multi-model performance across the vision and language. The final loss is the sum of the content loss and the structural loss.

$$L_{DOC2PPT} = L_{content} + L_{structure} \tag{2.15}$$

One of the major drawbacks of the DOC2PPT model is that the paragrapher block can only rephrase sentences but not paragraph-level summarization, consequently, it fails to capture the hidden global dependencies necessary for bullet point generation. Besides, DOC2PPT employed a complex system that designed the loss function to consider both structural and content-wise differences between predictions and labels. This involved combining the two aspects into a single training loss, but the challenge arose in determining the appropriate weights for each part. This resulted in significantly unequal contributions to the gradient during training, leading to the model's failure to fulfill the author's promise of excelling in both segmentation and summarization tasks.

Figure 2.8: Example of paragraphs in monthly report and in history text.

The DOC2PPT framework has another notable drawback related to its use of the RNN-based network structure as a *Document Reader* module for segmentation. Unfortunately, the RNN-based structure has shown limitations in handling long-range dependencies and, as a result, struggles to capture the necessary context information. Considering that global context plays a crucial role in proper segmentation, this becomes a significant drawback. In different contexts, the appropriate segmentation of a given piece of text can vary significantly. For instance, a text describing a man's activities on Day 0, Day 10, Day 15, and Day 20 may need to be divided into four segments in a monthly report. However, the same text should be considered as a cohesive whole in a biography. Figure 2.8 illustrates an instance where a selected text recounts the tale of

an ancient nobleman named Meng Chang Jun, forming friendships with his followers. This excerpt is taken from the biography of Meng Chang Jun within "Shiji", a collection of historical records encompassing a vast 2,500-year period. Notably, it is trivial that any segmentation of Meng Chang Jun's story should refrain from dividing the provided text into four separate parts.

2.5.2. D2S

For the purpose of our comparative experiment with the D2S model, it is essential to provide an overview of the D2S model's structural components. Diverging from the DOC2PPT model, the D2S model operates within a framework centered on closed-domain question answering. The D2S system is composed of three key modules: a keyword module responsible for capturing the hierarchy of article sections through their titles and contents, an information retrieval (IR) module designed to detect relevant information, and a question-answer module focused on generating slide content. In the slide generation process, users input their desired slide titles, prompting the IR module to identify relevant keywords by comparing word embeddings based similarity and Levenshtein distance. These keywords are then matched with section titles using the keyword module, which subsequently extracts content from corresponding sections. Finally, the QA module is applied to generate the actual slide content.

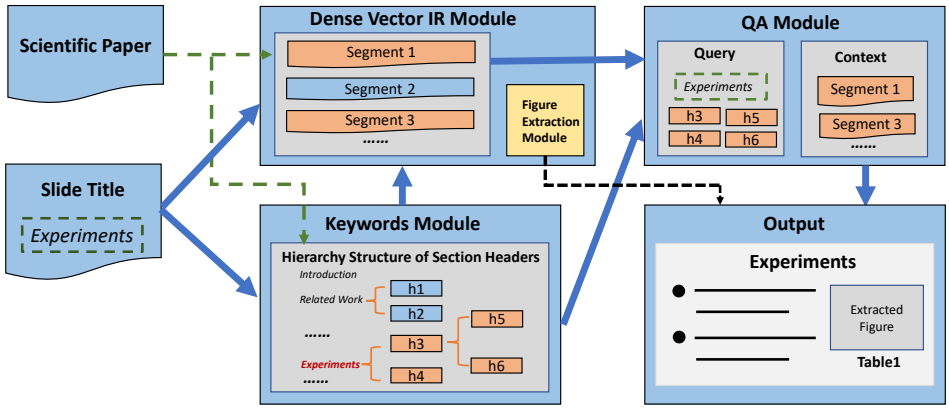


Figure 2.9: System architecture of D2S.

The IR module and the QA module undergo distinct training procedures, both requiring substantial data preparation and relying on tricky assumptions. For instance, during training, the IR module aims to minimize cross-entropy loss between slide titles and their corresponding contents on the same page. The underlying assumption here is that slide contents should closely resemble the article's contents, therefore aligning with the intended purpose of the IR module. However, the validity of this assumption remains unverified and introduces a potential for misguiding the model's training process.

Another major drawback of the D2S model is that users had to input slide titles to enable the system to extract the most relevant information from the paper and further generate the slides. This interactive design sacrificed efficiency in production and also

posed a limitation as users who were not well-acquainted with the paper might not obtain the desired output.

In conclusion, both the D2S and DOC2PPT frameworks from previous studies exhibit redundant system designs, which substantially elevate the complexity of their respective frameworks. Consequently, this results in considerable difficulties during the training and deployment phases. However, the optimization framework of reconstruction loss proposed in our study offers a notable solution to this issue and, remarkably, achieves superior performance compared to the aforementioned approaches. In section 3 and section 4, we will introduce the implementation details and report the comprehensive test results, drawing insightful comparisons with existing approaches.

2.5.3. Supervised Approach Limitations

An important decision we made during our model's implementation process was to utilize unsupervised learning, even though there are a few accessible datasets for the paper-to-slide generation task. Within this section, we perform a comparative analysis of various training frameworks used in prior research and contrast them with our approach, highlighting the advantages inherent in our chosen methodology.

The first issue stems from the limited size of the publicly available dataset, which poses a challenge for training modern large language model (LLM) powered software from scratch. Notably, modern LLMs commonly encompass more than 300 million parameters, while all existing paper-to-slide datasets consist of merely 1000 to 10000 training data pairs. This quantity is insufficient to initiate comprehensive pre-training of a model or even to adequately support fine-tuning efforts.

Furthermore, there is an additional challenge due to the limitations on input length for affordable language models. For instance, the GPT-3 model has a maximum input length of 2048 tokens, while the utilized Bart model's limit is 1024 tokens, equivalent to about 700-800 words. Given that typical research papers span between 4000 to 6000 words, the previous approach in the field involves breaking down the article into segments before applying the language model for generating slides. Building on this notion, the language model necessitates training on a dataset where each data point comprises matched slide content and its corresponding article content. In this context, the DOC2PPT model establishes this dataset by leveraging a fine-tuned BERT model, RoBERTa [LOG+19], to extract word embeddings from sentences within slides and documents. Subsequently, these sentences are matched based on cosine similarity. Conversely, the D2S model formulates a ranking function for mapping slide titles to corresponding paper content. This function, known as the Information Retrieval (IR) module, is trained on slide titles and their associated contents from the same page. However, these approaches are intricate and rely on a simplistic word embedding similarity approach. This similarily based approach is based on the idea that the content of the slides should closely resemble the content of the article, which is in line with the main goal of the IR module. Consequently, they can lead to the generation of a dataset with substantial noisy labels and exert significant influence on the performance of models trained on such data.

In our proposed framework, we employ unsupervised learning techniques as a means to address these challenge. This approach circumvents the requirement of meticulously

aligning parallel data to construct the dataset, streamlining the data preparation process and significantly simplifying the software architecture. As a result, we are able to achieve enhanced performance without the burden of explicitly paired but also noisy training examples.

Chapter 3

Method

Within this section, we present the implementation of our proposed software. We will commence by providing an introduction to the key data processing components within the software, encompassing both the pre-processing and post-processing units. Following this, we will present an innovative and actionable redefinition of the paper-to-slides problem. This new perspective enables us to address the challenge through a systematic optimization framework. Lastly, we will give an overview of the Paper2Slides model, which serves as a core component of the proposed software. This involves a detailed presentation of the implementation and training details for each of the sub-models integrated into the software.

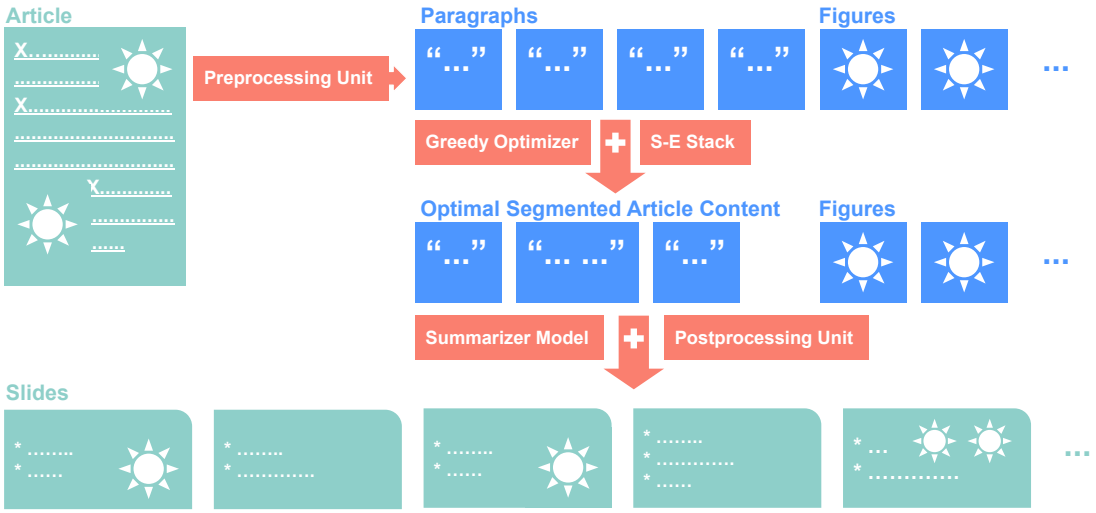


Figure 3.1: We present Paper2Slides, a model designed to create a slide deck from a scientific paper document.

The structure of the proposed software is shown in Figure 3.1. Begin by providing a scientific paper in PDF format, which then undergoes a sequence of three stages: preprocessing, model generation, and postprocessing. Ultimately, this series of steps

results in the generation of a corresponding presentation.

3.1. Data Processing

When conducting research in the field of natural language, it is customary to work with an input dataset comprising text that is often encoded in a user-friendly format like `.txt` or `.json`. As a result, for the implementation of our proposed software, it becomes imperative to develop a data reader program. This program serves the purpose of transforming raw data into a format that can seamlessly integrate into our research workflow. Simultaneously, to generate the desired slide-format output, an additional post-processing component is required. This post-processing unit plays a crucial role in converting the content generated by our system into a finalized PDF format.

3.1.1. Pre- & Post-Processing

The PDF reader utilized in this project consisted of two primary phases: text processing and detection of figure-like objects. The first phase was accomplished using the GROBID software, while the second part involved the application of the `PdfFigures2` software. To prepare the data for processing, the initial step required input files in `.pdf` format. After the first processing step, the output was a text string containing all the text from the document's body. In the subsequent step, the software created a separate directory containing the identified image-like objects.

`GROBID`. `GROBID` is software that focuses on scientific document processing and information retrieval. The link to this project is `https://github.com/kermitt2/GROBID`. The GROBID function is actively under maintenance and it has shown solid results. The `GROBID` model may remove from body texts a variety of unnecessary texts, including footers, headers, and table contents[GRO23].

`PdfFigures2`. The D2S research employs an AI model called the `Pdffigures` 2.0 model. This tool is utilized to detect 'Figure-like' objects, such as figures and tables (excluding formulas), within a `.pdf` document. Furthermore, it can accurately identify embedded images along with their corresponding captions (as mentioned earlier). Further information is available in their paper [CD16]. We explored the capabilities of the `Pdffigures` 2.0 tool by conducting tests on various `.pdf` documents. Our testing revealed that `Pdffigures` 2.0 effectively provides accurate and complete pairwise results for detecting pictures and tables. The software has its own output datatype. The program can take multiple document inputs and returns a list of values `[page_number, bounding_box_figure, texts_in_figure, ...]` for each detected image.

The processed results are utilized by the Paper2Slide model to generate content, and this content is then integrated by the post-processing unit to create slides. To achieve this, we employ a LaTeX compiler, using a predefined template. Once the paper-to-slide model produces textual slide content, it is inserted into the template along with figure-like objects. Subsequently, the LaTeX file is compiled, resulting in the final outputted slides.

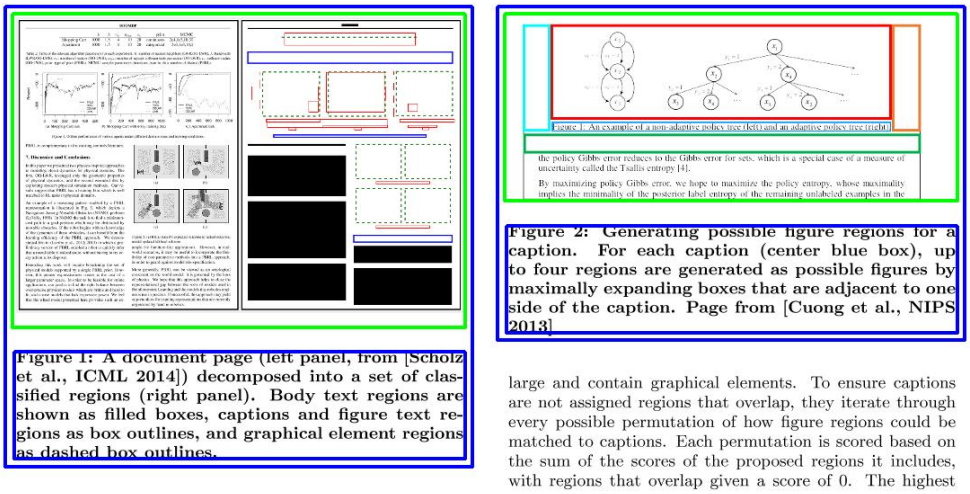


Figure 3.2: The `Pdffigures` 2.0 result.

3.2. Paper-to-Slide Problem

To tackle the limitation of the existing works, we approach the task of paper-to-slide generation as an optimization problem focused on minimizing information reconstruction loss. This approach is detailed in the upcoming sections. We start by formulating a redefined research question and subsequently progress through a sequence of structured steps to arrive at our proposed solution.

3.2.1. Task Re-Introduction

In the Introduction section, we presented our objective of developing software capable of generating presentation slides from scientific papers in PDF format. To provide a clearer understanding of the task, we intend to define it within the context of information extraction. Previous research has suggested that this task comprises two distinct components: structural decomposition (segmentation) and slides' content generation (summarization). However, this design leads to an overflowing and redundant ML system structure, significantly increasing the complexity of the task. Upon reconsideration, we propose a different perspective. We claim that the essence of the paper-to-slides generation task lies in **extracting the most valuable information and presenting it in the form of slides**, and we will illustrate later how redefining the research question enables a unified framework to address both segmentation and summarization.

To identify the most valuable information within a given paper, we recognized the necessity for a robust method to measure information quantity, leading us to introduce the reconstruction loss as a key metric. Through the adoption of the reconstruction loss as an information measure, we effectively reformulate the problem of *extracting the most valuable information* into an optimization problem. This optimization task entails simultaneously determining an optimal text segmentation and generating the most infor-

mative summaries for each segmented section. By unifying the structural decomposition (segmentation) and content generation (summarization) tasks under the framework of reconstruction loss, we present a cohesive and scientifically rigorous approach to address this complex challenge.

3.2.2. Summarizer-Expander Structure

When considering the information contained within the text, the first question arises: *How can we effectively describe or measure the quantity of information embedded in textual language?* To answer this question, we turn to summarization models as valuable tools, which serve as useful indicators of the most essential information within a given text.

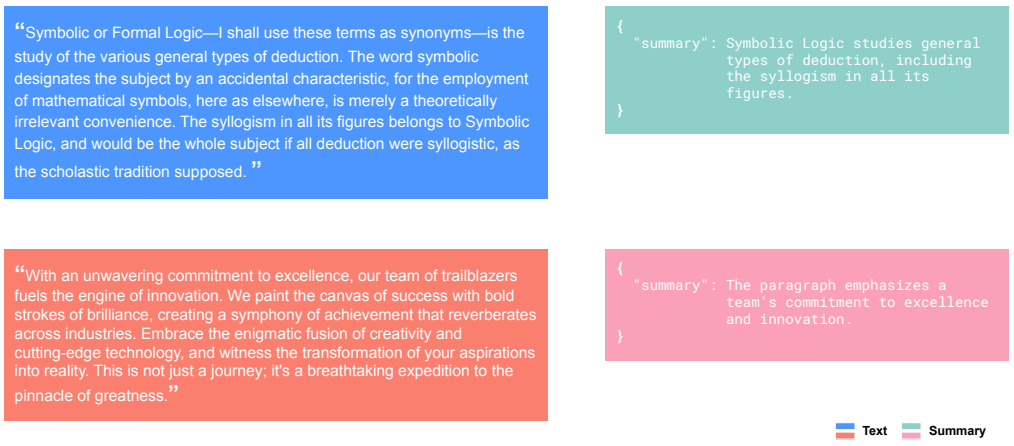


Figure 3.3: Examples of text containing different quantities of information and their corresponding generated summaries. The text in the blue textbox is extracted from Bertrand Russell's book, The Principles of Mathematics [Rus20]; the text in the red textbox is dummy text generated by GPT-3 using the prompt: Can you behave like a bullshit generator and give some long texts like paragraphs that have very low information content and are often found in marketing materials, corporate slogans, or motivational speeches, designed to sound impressive without conveying substantial content.

Summarization tasks can be viewed as an effort to capture the main ideas and concisely articulate them in human language. For example, texts with high information density present a greater challenge for summarization compared to texts with lower information density, even when both texts share the same length. Observing the example shown in Figure 3.3, we can discover the contrast between a highly informative paragraph and one lacking substance. A less meaningful paragraph tends to employ buzzwords, cliches, and vague phrases that may sound impressive but lack specificity and meaningful content. Consequently, the generated summary for such paragraphs can easily capture the main idea due to the relative simplicity of the text. On the other

Figure 3.4: Examples of an expander models input and output.

hand, paragraphs with a higher information density pose challenges in summarization, inevitably leading to some loss in the summarization process. In conclusion, the varying quantities of information present in the texts affect human understanding, thus influencing the lengths and quality of summaries generated by both the AI model and humans. The results obtained from summarization might lead humans to easily assess whether the original text contains substantial information or not. However, this method does not provide a quantified approach for measuring the actual quantity of information present in the text.

Taking a further stride in our investigation, we extend our focus beyond solely the summarization model to include an expander model. The expander model is a functional construct capable of generating longer text based on the information contained in a shorter text. Notably, in numerous instances, the expander model incorporates information that is inferred from or logically related to the original text, thereby augmenting the quality of the output. Figure 3.4 shows an example of how the expander model works.

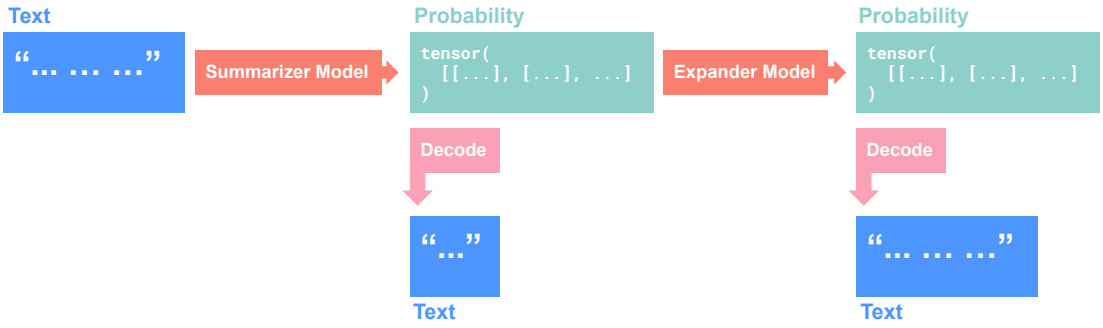


Figure 3.5: Visualized stacked S-E structure.

With the expander model at hand, we can now delve into the concept of reconstruction loss. The idea behind reconstruction loss is straightforward: consider a pair of models, a summarizer model, and an expander model (referred to as the S-E pair), in an ideal scenario, these models would be perfect, surpassing all others in their respective

summarization and expansion tasks, providing optimized outputs for any given task. By stacking this perfect S-E pair, we define a function where, for any textual input, the summarizer model generates a concise summary as an intermediate representation, compressing the information into a shorter length of text, and then the expander model works to expand this summary back to the original text, striving to preserve its essence faithfully. While a perfect S-E pair would effortlessly summarize any text and faithfully expand it back to the exact input, ensuring the existence of such a perfect pair remains uncertain. Thus, in practice, we typically utilize a pair of non-perfect summarizer and expander models, which often results in a difference between the input and generated texts. We define the difference as the reconstruction loss. It is important to note that, as discussed in the previous paragraphs, reconstruction loss exhibits the following properties:

- When using a stronger S-E pair for a particular paragraph of text, the reconstruction loss will be smaller;
- In the case of a specific S-E pair, a less informative input is likely to result in a smaller reconstruction loss;
- With a fixed length limit of the intermediate representation and considering a specific S-E pair, longer inputs tend to have a larger reconstruction loss.
- Without proof, we hypothesize that for a particular S-E pair, a sequence of logically coherent paragraphs would exhibit a smaller reconstruction loss compared to a text with greater logical incoherence. This hypothesis is very important and will be verified in the 4.3 section

3.2.3. Reconstruction Loss

Drawing on the thought experiment presented in the previous paragraph, we provide a mathematical definition of the reconstruction loss and subsequently demonstrate the unified optimization framework for the slide generation task.

For a piece of text t, and the S-E pair $S(\cdot), E(\cdot)$, the reconstruction loss is defined as follows:

$$\text{reconstractionLoss}(t) = \text{CrossEntropy}(E(S(t)), t) \tag{3.1}$$

Note that both the summarizer and the expander are language models, therefore, their outputs are probabilities, hence we could use the cross entropy loss to measure the difference. Figure 3.5 shows how the reconstruction loss is computed through a stacked S-E structure.

It is important to note that the position of $E(\cdot)$ and $S(\cdot)$ in the term $E(S(t))$ can not be switched, because the S-E structure's purpose is to compress information into a narrower channel, retaining only the essential details while filtering out irrelevant information. A similar structure of the S-E stack in the machine learning area is the auto8 ;encoder model, which is a type of neural network architecture that is designed for a compressed representation or encoding of the input data in a lower-dimensional space, known as the bottleneck layer. because the S-E structure's purpose is to compress information into a narrower channel, retaining only the essential details while filtering

out irrelevant information. a key difference between the S-E stack and the autoencoder model is that the S-E stack cannot be trained end-to-end. In the autoencoder model, there are no restrictions on the intermediate representations, whereas, in the S-E stack, the intermediate representations must be language model probabilities that can be decoded into human language. Given the potential negative impact on the model's ability to generate human language, it is not recommended to pursue end-to-end training of the S-E model.

Recalled that we defined the task of paper-to-slide generation as 'extracting the most valuable information and presenting it in the form of a slide', and with the reconstruction loss serving as a measurement of the information quantity, we can now reformulate the paper-to-slide generation task as the following optimization problem:

$$seg^* = argmin_{seg \in S} \sum^{t_i} \text{reconstructionLoss}(t_i) \tag{3.2}$$

where the symbol *seg* represents a segmentation method applied to the textual content of the paper. This method divides the content T into separate 'blocks' denoted as t_i, such that their union forms the whole content of the paper. The set S includes all the valid ways we can divide the content, following certain rules. For example, the rules ensure that no paragraph is split into multiple blocks, each block covers only one section at most, and that there is a maximum length limit for any block.

The solution to the optimization problem, denoted as seg^*, is referred to as the *optimal segment*. Utilizing this, we can generate slides that capture the most valuable information as follows: For each page in the slides, the content is produced by decoding the output from the summarizer models, using the blocks t_i as input. The block t_i acts as the receptive field for the generated content.

$$t_i^s = \text{decode}(S(t_i)) \tag{3.3}$$

In the software, we decided to use the BART (Bidirectional and Auto-Regressive Transformers) model as both the summarizer and expander model, as the sponsoring company possessed experience in fine-tuning and deploying the BART model. For the summarizer, we take the Bart-large-CNN model as the pre-trained model, which is pre-trained on the CNN Dailymail Dataset [SLM17, HKG+15], expecting to take some advantage of its proven summarization ability [LLG+19]. We have found no existing expander model, therefore we decide to train it from scratch. The details of the Paper2Slides model design will be given in the next section.

3.3. Paper2Slides Model

In this section, we delve into the specifics of implementing and training the sub-models within the Paper2Slides framework. We initiate with a concise overview of how the stacked S-E structure is implemented, along with the pre-training of the S-E stack performed on the expander model. Subsequently, we break down the S-E stack and proceed to fine-tune its components in parallel, employing the training methodology put forth in the Unsupervised Machine Translation literature. Following this, we provide

a comprehensive account of the greedy algorithm utilized to address the optimization challenge of the reconstruction loss.

3.3.1. Pre-training S-E Stack

As mentioned earlier, the S-E stack plays a crucial role in computing the reconstruction loss and achieving a more convincing measurement of information quantity. To ensure maximum effectiveness, both the summarizer model and the expander model need to be highly powerful. Therefore, we selected the most potent pre-trained model as our summarizer and fine-tuned it. However, since there were no pre-trained expanders available, we took on the task of pre-training one from scratch. For both summarization and expanding tasks, we utilized the BART model [LLG+19] as the underlying neural network structure. The decision was primarily driven by practical reasons, as the sponsoring company possessed experience in fine-tuning and deploying the BART model. To train the stacked S-E models, we used the CNN Daily Mail dataset[SLM17, HKG+15], which contains a large collection of English news articles and their corresponding highlights. While originally designed for machine reading and abstractive question answering, this dataset has found extensive application in training abstractive summarization models.

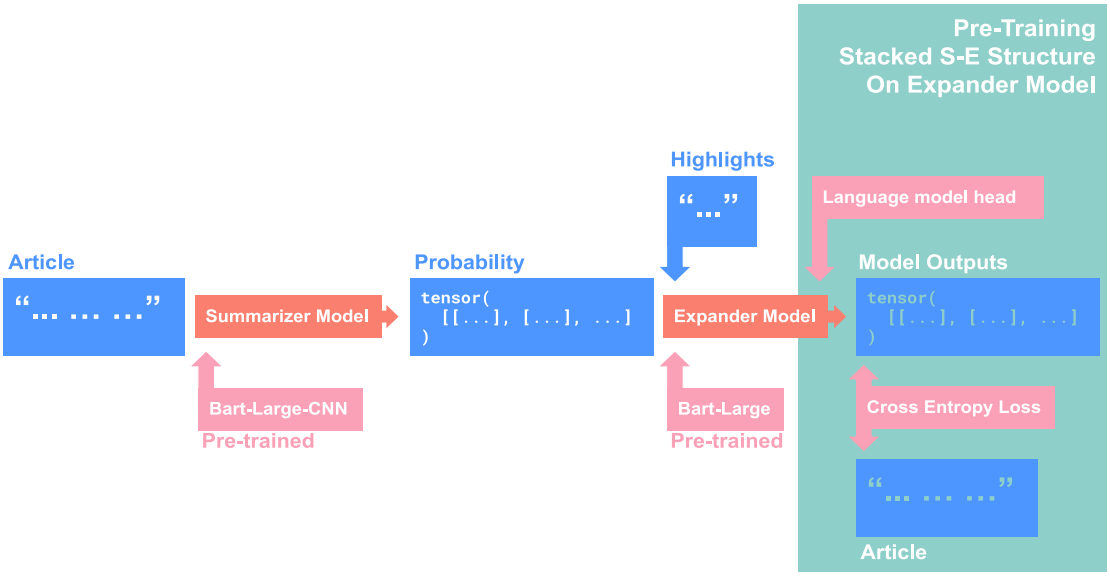


Figure 3.6: Pre-training the stacked S-E structure

We implement the S-E function in the following way: During the inference phase, for any text input, we first employ the Bart tokenizer to break down the text into tokens. These tokens are then fed into the summarizer model. Instead of decoding probabilities generated by the summarizer model, we directly input these probabilities into the expander model. However, we used an unusual way in our training phase. When provided with a pair of $\langle \text{article}, \text{highlight} \rangle$, we first fed the articles into the summarizer model, and then we utilized the outputs from the summarizer model as inputs for the

encoder in the expander model, while the highlights serve as input for the decoder in the expander model. Subsequently, we calculate the loss by comparing the output of the expander model with the ground truth labels.

In practice, we employ the Bart-Large-CNN checkpoint as the initial weight for our summarizer model. This choice is rooted in its popularity and extensive use as a pre-trained model for summarization tasks. The Bart-large-cnn model is also fine-tuned on the CNN DailyMail Dataset, regarding the expander model, we simply load the Bart-large model's weight for the transformer components and introduce a randomly initialized linear layer as the *language model head*. This language model head is typically added atop a pre-trained foundational language model, aiming to fine-tune it for a specific task. As it is shown in Figure 3.6, training is only performed on the expander model. We maintain the other parameters in a frozen state, excluding those within the language model head. This effectively results in training solely the final layer. We tried multiple training settings when fine-tuning our model on the CNN Daily Mail dataset and selecting the best model based on its results on the validation dataset.

The assessment of the pre-trained S-E stack can be found in section 4.1. In conclusion, we have pre-trained the expander model part in an S-E stack structure and got the desired result. In the next section, we will give the details of finetuning the S-E stack model on a scientific corpus, in order to get a stronger and more specialized model.

3.3.2. Fine-tuning S & E

Although we already have a pair of the summarizer and the expander models, we still would like to finetune them on the scientific corpus to adapt our models to the domain of scientific language. To achieve our objective, we utilize an unsupervised training algorithm discussed in the literature review section. We do this for two main reasons: Firstly, we couldn't find an appropriate summarization dataset that specifically caters to scientific language, which would have allowed us to separately train both models. Second, as we stated in the reconstruction loss section, end-to-end training of the S-E stack is not possible. If we were to fine-tune the S-E stack in an end-to-end manner, the gradient of the loss function would propagate through the entire neural network, covering both the summarizer and the expander models. Consequently, the summarizer model is fully updated thereby potentially losing its capacity to generate fluent language, which is to say, the model's decoded output might become unintelligible. The core problem here is the absence of a restriction that enforces the models to maintain the qualities of a language model. As a result, the trained model could essentially turn into an encoding model, producing compressed and vectorized representations for each input, without retaining meaningful language characteristics.

Figure 3.7 below illustrates different training approaches leading to distinct training paths. A significant portion of these paths only results in an encoding model, failing to ensure that the output maintains its role as a language model.

To address the issue of training, we employed the Unsupervised Machine Translation (UMT) algorithm. This algorithm is specially designed to train a pair of translation models without the need for explicit supervision. Among the two translation models, one model translates text from a source language to a target language, and the other model performs the opposite translation. We utilized this algorithm to parallelly train

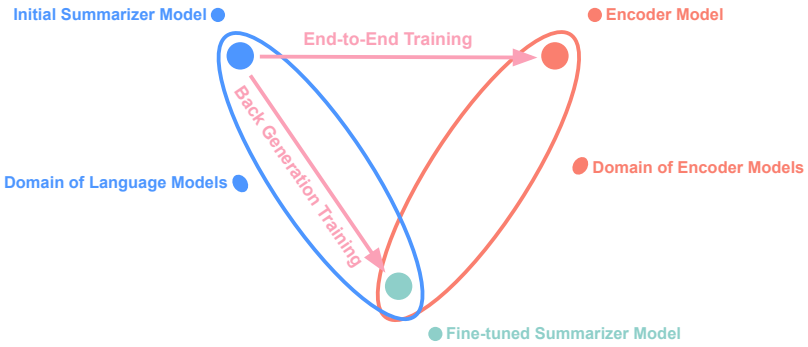


Figure 3.7: Different training methods could lead to different training results.

both the summarizer and the expander models. In this process, we broke down the S-E stack into its individual components: the summarizer and the expander models. During each iteration of training, both models were utilized to summarize or expand some given piece of text, then the generated text and its source text are used as training input and the label to train the other model. The process undertaken in a single training step is shown in Figure 3.8. Drawing inspiration from the concept of *back-translation*, we refer to this procedure as the *back-generation* step.

In practice, we perform the finetuning on the 'Automatic Slide Generation from Scientific Papers' dataset, a corpus of 5000 paper-slide pairs and it is specifically designed for presentation slide generation [SMWG21]. Although the articles and slides are paired in the dataset, the paragraph and the bullet points are not, therefore, we select all paragraphs in the articles that have more than 200 tokens and less than 1024 tokens (which is the maximum input length of BART), these will be fed into the summarizer models to generate the training data for the expander model, similarly, we select all bullet points that have between 30 to 200 tokens and use the expander model to prepare the training data for the summarizer model. During the whole finetuning procedure, we performed 7 times the *back-generation* training, and the same training hyperparameters are used across all 7 times: the number of epochs is 3, the batch size is 4, the learning rate is 2e-6, and the optimizer is AdamW. After each back-generation training phase, we update the models and prepare the training dataset for the next iteration.

Note although our training method is not exactly the same as the UMT algorithm in terms of the model usage and training details, our training method actually matches the three essential and necessary principles mentioned in the UMT work. First, we have a pair of pre-trained models as initial 'translater' models. Second, both summarize and expander models serve as strong language models. Third, at each training step, we use one of the models to generate a training dataset for the other model and train the other model under the back-generation step. This alignment with the fundamental principles in our training methodology enables us to achieve the desired model enhancement through unsupervised learning, even if our approach differs in certain aspects from the UMT algorithm. Both models are presently accessible on the huggingface hub. As of now, the summarizer model has downloaded more than 1400 downloads, and the

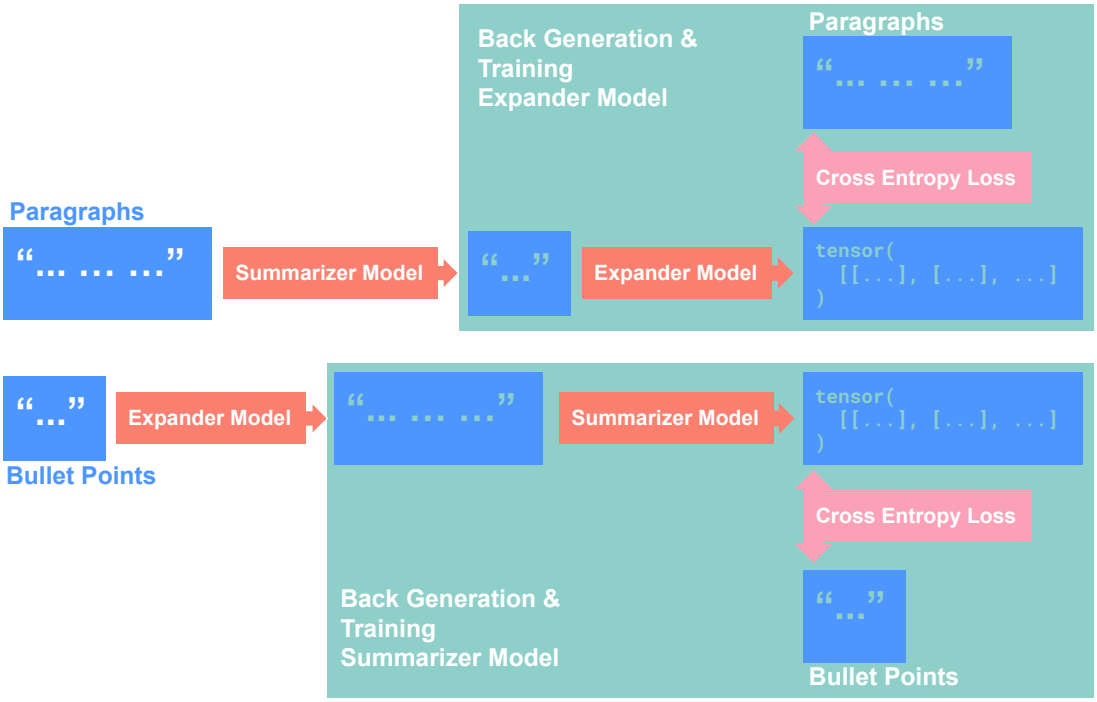


Figure 3.8: Back-generation step.

expander model has been downloaded over 300 times.

3.3.3. Greedy Optimizer

After pertaining and fine-tuning the S-E stack, we could finally focus on solving the optimization problem we defined in section 3.2.3. Recall that the problem is

$$seg^* = argmin_{seg \in S} \sum^{t_i} \text{reconstructionLoss}(t_i) \tag{3.4}$$

where the symbol seg represents a segmentation method and t_i are segmented blocks from the input article. As the reconstruction loss has no closed-form representation, it is trivial that the proposed problem is non-analytic. Similarly, we are unable to derive a precise mathematical formula for the optimal segmentation, denoted as seg^*. In light of this, we need to turn to iterative optimization methods that can dynamically adjust and adapt, rather than relying on fixed procedures. Ultimately, we opted to utilize a greedy search algorithm to uncover the solution, in order to take some advantage of its efficiency in development and robustness in deployment. Moreover, it is important to note that a neural network may also work on solving the optimization problem but that would take much longer time to implement and exceed our expected time.

Before we start explaining how the greedy algorithm works, it's crucial to lay out the segmentation rules. These rules serve as governing principles, dictating the behavior of the segmentation functions to align with our intended objectives. Here are the

segmentation rules we follow:

- A paragraph serves as the fundamental segmentation unit, and it should not be divided across multiple blocks.
- Each block should exclusively contain content from a single section within the input article.
- The total tokens within a block should not exceed 1024. This constraint is attributed to the input length restriction of the BART model.

With the rules now distinctly established, we can delve into the specifics of the greedy algorithm. The inputs for this optimization process encompass the article's text including the titles and the contents of individual sections, as well as the user's indicated range for desired slide length. This range is bounded by two values, namely minpage and maxpage. Importantly, minpage is expected to have a numerical value smaller than maxpage, and maxpage should be lesser than the total count of paragraphs within the input article.

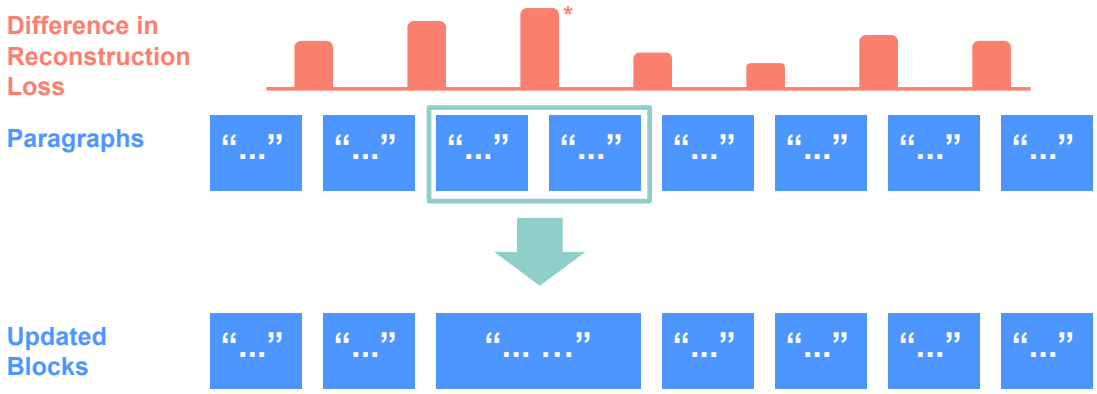


Figure 3.9: The update rule of the greedy algorithm, the highest value of the difference in reconstruction loss is marked with a *.

The greedy algorithm starts from the most subdivided state of the article, where the article is fully decomposed into paragraphs. The algorithm computes and stores the reconstruction loss associated with each paragraph. During each update cycle, the algorithm computes and records the reconstruction loss for every pair of consecutive paragraphs. This computation is performed under the condition that no rule is violated after merging these two paragraphs into a single block within a segmentation. Following this, the algorithm selects and concatenates the two consecutive paragraphs, denoted as $T^* = (t_1^*, t_2^*)$, that exhibit the highest difference in reconstruction loss before and after merging them. This selection represents the optimal update, aimed at minimizing the overall reconstruction loss.

Note that the aim of the greedy optimization algorithm is to minimize the global reconstruction loss, therefore at each update step, the algorithm should compare the *difference in reconstruction loss* before and after the update, denote as $\mathrm{D}(seg_1, seg_2)$,

$$\mathrm{D}(seg_1, seg_2) = \sum^{t_i \in seg_1} \text{reconstructionLoss}(t_i) - \sum^{t_i \in seg_2} \text{reconstructionLoss}(t_i) \tag{3.5}$$

In our greedy algorithm, the update is only performed on two paragraphs, $T^* = (t_1^*, t_2^*)$, in a segmentation, therefore the *difference in reconstruction loss* could also be written as,

$$\begin{aligned} \mathrm{D}(T) &= \sum^{t_i \in seg_1} \text{reconstructionLoss}(t_i) - \sum^{t_i \in seg_2} \text{reconstructionLoss}(t_i) \\ &= \text{reconstractionLoss}(t_1^*) + \text{reconstractionLoss}(t_2^*) - \text{reconstractionLoss}(T) \end{aligned} \tag{3.6}$$

where the reconstruction loss is computed through the fine-tuned S-E stack. In the rest of this paper, we will refer to $\mathrm{D}(T)$ simply as the *difference in reconstruction loss* for a pair of paragraphs $T = (t_1, t_2)$.

Applying formula 3.6, it becomes possible to compute the difference in reconstruction loss by comparing the concatenated sequential paragraphs with the sum of reconstruction loss of the same paragraphs taken individually. This computation can be efficiently done using stored data, allowing for the identification of the optimal update based on the difference in reconstruction loss, as depicted in Figure 3.9.

After each update, we assess whether the total number of blocks remains greater than a given maxpage value. If this criterion is met, we repeat the previous step. However, if the maxpage limit is reached, we modify the update rule. In this case, we merge the consecutive paragraphs $T^* = (t_1^*, t_2^*)$ that possess the highest $\mathrm{D}(T^*)$, exclusively when $\mathrm{D}(T^*)$ is positive. Essentially, we only update the reconstruction loss if this specific update results in a decrease in the global reconstruction loss. If no such paragraphs satisfy these conditions, or if the minimum required number of pages (minpage) is attained, the program is terminated.

Once we get the segmented texts, the segmented text subsequently undergoes processing through the fine-tuned summarizer model to generate content for the slides. The generated content is further refined by employing post-processing utility functions, which transform the content into a LaTeX file. This file is eventually compiled into slides, serving as the final output.

Chapter 4

Experiments

We offer an extensive assessment of the Paper2Slides framework, encompassing evaluations of the main model and its sub-components. These evaluations involve multiple dimensions, including gauging the performance of the pre-trained expander model on the CNN-daily dataset, conducting human evaluations comparing the fine-tuned summarizer model with the baseline Bart-large-CNN model, and executing a comprehensive system-level comparison between the proposed Paper2Slides model and the D2S model, which acts as the baseline for comparison. Additionally, we provide illustrative examples showcasing the outputs of the individual sub-models.

4.1. Pre-trained Expander Assessment

Within this section, we provide an overview of the evaluation outcomes concerning the pre-trained model. It's crucial to highlight that the pre-training procedure exclusively targets the expander model, consequently, our evaluative endeavors are concentrated on analyzing the expander model's performance within the pre-trained S-E stack. Upon carefully assessing the expander model's performance using the CNN-daily dataset, we will subsequently provide a selection of exemplary outputs generated by both our summarizer and pre-trained expander models. These illustrative examples serve to effectively showcase the practical outcomes and capabilities of our models in real-world scenarios.

4.1.1. Experimental Results

In this section, we present assessment results for the pre-trained model, specifically focusing on the extracted expander model from the pre-trained S-E stack. Our testing procedure involved the following steps: For each highlight in the test dataset, we employed the expander model to extend it into longer text, aiming for a length equivalent to the corresponding article. Subsequently, the Rouge score is computed between the expanded text and the ground truth article.

We utilized the ROUGE score as an automated assessment tool, which is a commonly employed metric that assesses the quality of generated text by measuring the degree of token or n-gram overlap [Lin04]. The results from the tests, as illustrated in Figure 4.1, align with our expectations, revealing a significant coherence between the expander's

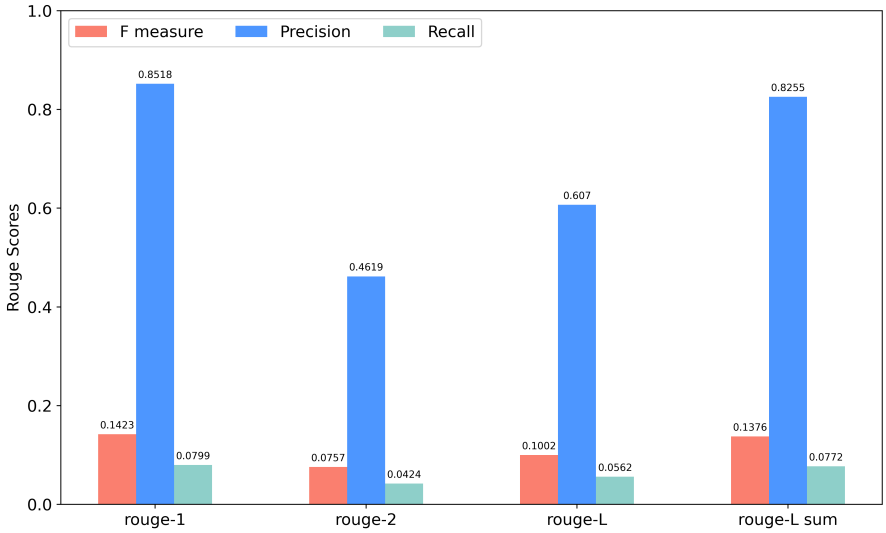


Figure 4.1: Rouge scores on test dataset

outputs and the provided information. However, it's important to highlight that the expander model may have limitations in generating entirely novel information. The Rouge precision of our experiment results is notably high, indicating that the expanded text encompasses nearly all the details found in the article. Meanwhile, the recall rate is relatively lower, suggesting that substantial information within the expanded text remains absent. This matches the fact that many intricate details are usually not included in the initial loaded highlights. To provide a visual representation, the Venn diagram in Figure 4.2 illustrates the information coverage across highlights, articles, and expanded highlights.

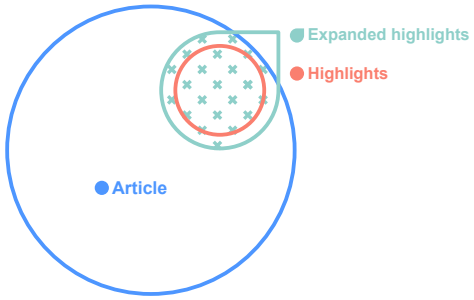


Figure 4.2: The Venn diagram of the information in the article, highlights, and expanded highlights. The area with a crossed texture represents the shared information between the article and the expanded highlights (outputs of the S-E model). The ratio of the crosshatched area to the blue disk's area represents the recall value, while the ratio of the crosshatched area to the green teardrop shape's area corresponds to the precision value.

4.1.2. Examples

In this section, we showcase two illustrative cases that exhibit the performance of our summarizer and pre-trained expander model. These cases are drawn from original articles in the CNN-daily test dataset. To conduct the assessment, we input these articles into our unfine-tuned summarizer model, which is aligned with the BART-large-CNN model. Subsequently, the outputs from the summarizer are fed into the pre-trained expander model. Throughout this section, we provide two examples to highlight various attributes exhibited by the pre-trained models. A consistent observation across these examples is the expander model's ability to generate new information through logical inferences or by tapping into broader world knowledge.

In every case, we employ a color-coded method to underscore information that is present in both the original news and the summaries. This information is highlighted in blue. Moreover, we identify content that is introduced for the first time in the expander outputs using a green color highlight.

The first example comes from the news of an accident at a zoo.

Article:

> "A drunk teenage boy had to be rescued by security after jumping into a lions' enclosure at a zoo in western India. Rahul Kumar, 17, clambered over the enclosure fence at the\xa0Kamla Nehru Zoological Park in Ahmedabad, and began running towards the animals, shouting he would 'kill them'. Mr Kumar explained afterwards that he was drunk and 'thought I'd stand a good chance'against the predators. Next level drunk: Intoxicated Rahul Kumar, 17, climbed into the lions' enclosure at a zoo in Ahmedabad and began running towards the animals shouting 'Today I kill a lion!' Mr Kumar had been sitting near the enclosure when he suddenly made a dash for the lions, surprising zoo security. The intoxicated teenager ran towards the lions, shouting: 'Today I kill a lion or a lion kills me!' A zoo spokesman said: 'Guards had earlier spotted him close to the enclosure but had no idea he was planing to enter it. 'Fortunately, there are eight moats to cross before getting to where the lions usually are and he fell into the second on"

Summarizer outputs:

> "Rahul Kumar, 17, clambered over the enclosure fence at the\xa0Kamla Nehru Zoological Park in Ahmedabad. He ran towards the animals, shouting he would 'kill them' Mr Kumar explained afterwards that he was drunk and 'thought I'd stand a good chance' "

Expander outputs:

> "Rahul Kumar, 17, clambered over the enclosure fence at the\xa0Kamla Nehru Zoological Park in Ahmedabad. He ran towards the animals, shouting he would 'kill them' Mr Kumar explained afterwards that he was drunk and 'thought I'd stand a good chance' He said: 'I thought I would be able to protect the animals. I was drunk. I didn't think I'd be able to kill them. I thought I'd protect them"

In this example, the expander model attempted to produce previously unencountered information by making logical inferences about Rahul Kumar's intuitions. The subsequent example originates from a sports news article.

Article:

> "Dougie Freedman is on the verge of agreeing a new two-year deal to remain at Nottingham Forest. Freedman has stabilised Forest since he replaced cult hero Stuart Pearce and the club's owners are pleased with the job he has done at the City Ground. Dougie Freedman is set to sign a new deal at Nottingham Forest . Freedman has impressed at the City Ground since replacing Stuart Pearce in February . They made an audacious attempt on the play-off places when Freedman replaced Pearce but have tailed off in recent weeks. That has not prevented Forest's ownership making moves to secure Freedman on a contract for the next two seasons."

Summarizer outputs:

> "Dougie Freedman is on the verge of agreeing a new two-year deal. Freedman has stabilised Forest since he replaced Stuart Pearce. Forest made an audacious attempt on the play-off places when Freedman replaced Pearce but have tailed off in recent weeks."

Expander outputs:

> "Dougie Freedman is on the verge of agreeing a new two-year deal. Freedman has stabilised Forest since he replaced Stuart Pearce. Forest made an audacious attempt on the play-off places when Freedman replaced Pearce but have tailed off in recent weeks. CLICK HERE for all the latest transfer news with our live updates from the Premier League and the Capital One Cup. Back to Mail Online home. back to the page you came from. Click here for the latest Premier League news."

In this example, the expander model endeavored to create previously unencountered information by using the world's knowledge about the commonly seen hint buttons on sports website.

4.2. Fine-tuned Models Assessment

After obtaining the fine-tuned summarizer and the expander model, we proceeded to conduct a human evaluation experiment. Unfortunately, we were unable to find a baseline expander model, leading us to exclusively carry out evaluations on the summarizer model. In these experiments, we compared the performance of our fine-tuned Bart model against the original Bart-large-cnn model. The results of these experiments are given in section 4.2.1 and several examples from the fine-tuned models are shown in section 4.2.2.

4.2.1. Experimental Results

Our experiments focused on 75 randomly selected paragraphs from 18 scientific articles, most of them are published in scientific journals and else are master's or PhD theses. The paper includes multiple science areas including pure mathematics [iK11], philosophy [Bac20], human geography [SL23], climate science [FC23, CF23], history [Bel05],

artificial intelligence [VSP+17, HLC+22, BB19, HCX+22, MBB21, BCLR22, ZCLS22, CvD22, SCVD22], electronic engineering, mechatronic engineerings [ZFW+23, PSC+22], biology [LMS23], and statistics [CJLZ22].

To ensure the credibility of the data, the papers were sourced from someone who possesses extensive familiarity with them (either as the author or by recently reading them) at our request. Alongside the papers, this individual also supplied feedback regarding the quality of the summaries produced by both the fine-tuned model and the Bart-large-CNN model. They were requested to evaluate whether the summaries effectively extract important information from the original text and determine their suitability for use in slides. Notably, the summaries were generated using identical parameters and were directly copied into the investigation document without undergoing any post-editing.

The article's providers are presented with three choices: they can either indicate the preferred summary number (0/1) or select the 'no decision' option. The ultimate findings reveal that in **45.33**% of instances, providers favored the result from our fine-tuned model. In **28.00**% of cases, providers perceived both summaries as having equal quality, while in **26.67**% of cases, providers favored the baseline model's result. The consistent preference for our model's summaries by providers, coupled with a notable percentage of cases where our model was deemed superior, strongly supports the conclusion that our fine-tuned model outperforms the baseline.

In many cases, the fine-tuned and baseline models generate similar results. It's interesting to note that the fine-tuned model shows a preference for shorter sentences and avoids using first-person expressions to a greater extent. Additionally, the fine-tuned model tends to produce more sentences overall, despite their shorter length, when compared to the baseline model under the same settings. We also observed a shared pattern between both language models: they excel when important information is located at the beginning or end of the input context. However, their performance notably declines when crucial information is situated in the middle of the context. This observation also matches with a recent paper on the topic of LLM behavior in long contexts [LVdMJ+21].

4.2.2. Examples

In this section, we created several examples to demonstrate and compare the performance of our fine-tuned summarizer and expander models. These examples showcase how the fine-tuned models handle scientific input texts, highlighting their ability to generate summaries and expand content. By using real-world texts and comparing them with the outputs of your models, we illustrate the effectiveness, features, and advancements of our approach compared to existing models.

The input texts for the summarizer models are carefully selected from the paper titled *Attention is all you need* [VSP+17]. All the examples presented in this context are directly extracted from the outputs of the program, without any subsequent editing. The same method of information highlighting utilized in the previous section is retained here, with an additional emphasis on information found both in the original source material and the outputs of the baseline summarizer model, denoted by an orange underline.

In the first example, the original text material revolves around paragraphs discussing the bag-of-features representations.

Paper contents:

> "We here recount the main elements of a classic bag-of-features model before introducing the simpler DNN-based BagNets in the next paragraph. Bag-of-feature representations can be described by analogy to bag-of-words representations. With bag-of-words, one counts the number of occurrences of words from a vocabulary in a document. This vocabulary contains important words (but not common ones like "and" or "the") and word clusters (i.e. semantically similar words like "gigantic" and "enormous" are subsumed). The counts of each word in the vocabulary are assembled as one long term vector. This is called the bag-of-words document representation because all ordering of the words is lost. Likewise, bag-of-feature representations are based on a vocabulary of visual words which represent clusters of local image features. The term vector for an image is then simply the number of occurrences of each visual word in the vocabulary. This term vector is used as an input to a classifier (e.g. SVM or MLP). Many successful image classification models have been based on this pipeline (Csurka et al., 2004; Jurie & Triggs, 2005; Zhang et al., 2007; Lazebnik et al., 2006), see O'Hara & Draper (2011) for an up-to-date overview."

Fine-tuned Summarizer's outputs:

> "Bag-of feature representations can be described by analogy to bag-of-words representations. Bag-of feature representations are based on a vocabulary of visual words which represent clusters of local image features. We here recount the main elements of a classic bag-of features model before introducing the simpler DNN-based BagNets in the next paragraph"

Baseline summarizer model's outputs:

> "Bag-of-feature representations are based on a vocabulary of visual words. The counts of each word in the vocabulary are assembled as one long term vector. This term vector is used as an input to a classifier (e.g. SVM or MLP) Many successful image classification models have been based on this pipeline."

In this example, it becomes evident that both the baseline and fine-tuned models produced summaries primarily centered around three sentences within the paper. However, the fine-tuned model's output places emphasis on the essential attributes of the bag-of-features representation, capturing the core concept of the paragraph. In contrast, the baseline model's output pertains to addresses more minor details of the bag-of-features representation.

Moving to the second example of the summarizer model's outputs, the selected input paragraphs serve as an introduction to the self-attention mechanism.

Paper contents:

> "The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions

grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions [12]. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 27, 28, 22].

End-to-end memory networks are based on a recurrent attention mechanism instead of sequencealigned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [34].

To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequencealigned RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [17, 18] and [9]."

Fine-tuned Summarizer's outputs:

"Self-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2. "

Baseline summarizer model's outputs:

"Self-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution. We counteract this effect with Multi-Head Attention as described in section 3.2."

In this example, both models generate summaries based on the same information in the original test dataset. However, the fine-tuned model offers a marginally more detailed account of parallel computation and sequential computation within neural networks compared to the baseline model.

For the expander model's examples, we utilize the output from the fine-tuned summarizer model as the input for illustrating the performance of the expander model. In

the subsequent example, only the outputs of the fine-tuned expander model are displayed. In the first example from the expander model, the inputs are derived from the outputs of the summarizer model in the first summarization instance.

Input tests:

> "Bag-of feature representations can be described by analogy to bag-of-words representations. Bag-of feature representations are based on a vocabulary of visual words which represent clusters of local image features. We here recount the main elements of a classic bag-of features model before introducing the simpler DNN-based BagNets in the next paragraph."

Expander's outputs:

> "Bag-of feature representations can be described by analogy to bag-of-words representations. Bag-of features representations are based on a vocabulary of visual words which represent clusters of local image features. We here recount the main elements of a classic bag- of features model before introducing the simpler DNN-based BagNets in the next paragraph. In the following paragraph, we describe the main features of a BagNet model. We then describe the DNN model that is used to represent the bagNets model."

The subsequent example involving the expander model utilizes the outputs of the summarizer model from the second summarization example as its inputs.

Input texts:

> "Self-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2."

Expander's outputs:

> "Self-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions. The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In section 3.2.2, we counteract this effect with Multi-Head Attention as described in section 3, 3.3.2 and 3,3.4. We describe how to compute hidden representations using self-att attention and multi-head attention. We also discuss how to reduce the number of sequential operations by averaging

attention weighted positions to reduce effective resolution. We conclude by discussing how to use self attention and Multi-head Attention to reduce sequential operations."

4.3. Property of Reconstruction Loss

Recall that without proof, we claimed that for a certain S-E pair, the more coherent paragraphs would show a smaller reconstruction loss compared to a text that displays more significant logical inconsistency. This suggests that the optimization framework based on reconstruction loss could potentially address the segmentation challenge inherent in the paper-to-slide generation process.

4.3.1. Hypothesis Testing

In this section, we provide evidence to support the above-mentioned claim through a verification test. We conducted a hypothesis testing experiment that involved comparing the difference in reconstruction loss between a two-paragraph piece of text and the sum of reconstruction losses of the same two paragraphs when separated. Mathematically, given a piece of text T consisting of two paragraphs t_1 and t_2, the *difference in reconstruction loss* can be formulated as follows:

$$\mathrm{D}(T) = \mathrm{reconstractionLoss}(t_1) + \mathrm{reconstractionLoss}(t_2) - \mathrm{reconstractionLoss}(T)$$

where the reconstruction loss is computed using the fine-tuned S-E stack.

It's important to note that for the majority of cases, the value $\mathrm{D}(T)$ is negative due to the fact that longer text typically results in a higher reconstruction loss. In our experiment, we carefully selected 2000 pairs of coherent and consecutive paragraphs from the dataset 'Automatic Slide Generation from Scientific Papers' [SMWG21]. We calculated the difference in reconstruction loss for these pairs, and the resulting scores are illustrated in the pink violin plot presented in Figure 4.3. For the sake of comparison, we also conducted an exercise where we randomly shuffled the pairs of those paragraphs. This was done in a way that each preceding paragraph was paired with a subsequent paragraph in a random manner. Subsequently, we computed the difference in reconstruction loss for these shuffled pairs and depicted the distribution of these scores in the green violin plot within 4.3.

Upon analyzing Figure 4.3, a distinct trend becomes evident: the difference in reconstruction loss of coherent paragraphs is notably greater when compared to randomly paired paragraphs. To provide a more robust assessment, we employed the Wilcoxon signed-rank test, a nonparametric statistical method available in the `Scipy` package. The Wilcoxon signed-rank test serves as a nonparametric alternative to the T-test and is used here to determine whether the difference between paired samples from two distributions is symmetrically centered around zero. The Wilcoxon can not be used on normal distributions. Therefore, prior to conducting the Wilcoxon signed-rank test, we undertook an important step - evaluating the normality of the distribution of the difference in reconstruction loss using the Shapiro-Wilk test. The testing results show that the distributions are not normal.

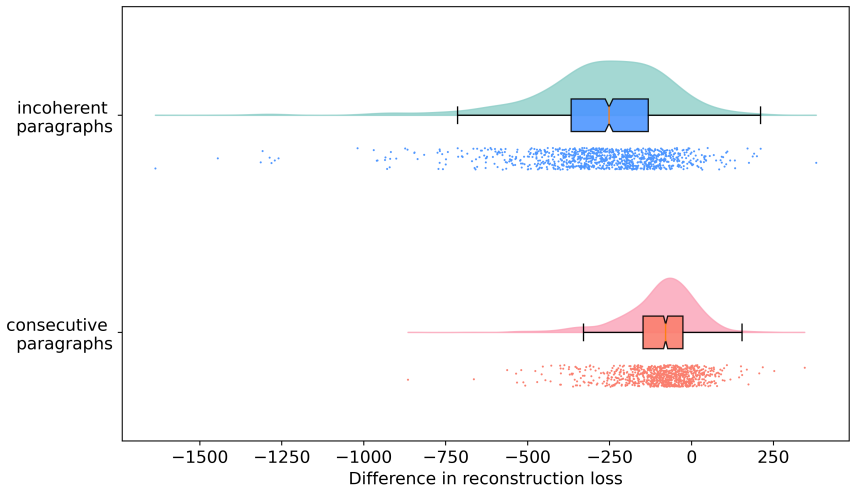


Figure 4.3: *Difference in reconstruction loss* of coherent paragraphs and randomly matched paragraphs.

Our Wilcoxon signed-rank test centers around the examination of the null hypothesis, which posits that the difference in reconstruction loss for incoherent paragraphs is statistically equal to or greater than that of consecutive paragraphs. The results of this examination yield a test statistic of 964303.0 and a p-value of **0.03784**. The significance of this p-value, being below the conventional threshold of 0.05, leads us to reject the null hypothesis. This rejection highlights a significant finding: the difference in reconstruction loss for consecutive paragraphs significantly surpasses that of incoherent paragraphs.

The observation and the statistical test results validate our hypothesis and provide additional support for the notion that optimal segmentation not only serves as a solution to the optimization challenge pertaining to information quantity but also functions effectively as an article decomposition or topical detection technique.

4.4. System Assessment

In this section, we present the outcomes of our testing in comparison to existing state-of-art models. Initially, our intention was to assess our model's performance against both the DOC2PPT [FWMS22] and D2S [SHW+21] models. However, we encountered a limitation with the DOC2PPT dataset. The slides in the DOC2PPT dataset were available only as images, without the corresponding extracted text. As a result, we couldn't carry out a meaningful comparison against the results of the DOC2PPT model. On the other hand, the D2S team graciously provided their processed dataset for evaluation. Therefore, we solely conduct the comparison against the D2S model.

4.4.1. Experimental Results

The D2S dataset includes a test dataset comprising 81 pairs of published papers and slides in the NLP area. Out of these pairs, we successfully obtained and processed 80 pairs utilizing our software to generate slide content. By comparing the content produced through our software with the reference ground truth from the dataset, we calculated Rouge scores. These scores were subsequently compared to the metrics presented in the D2S paper.

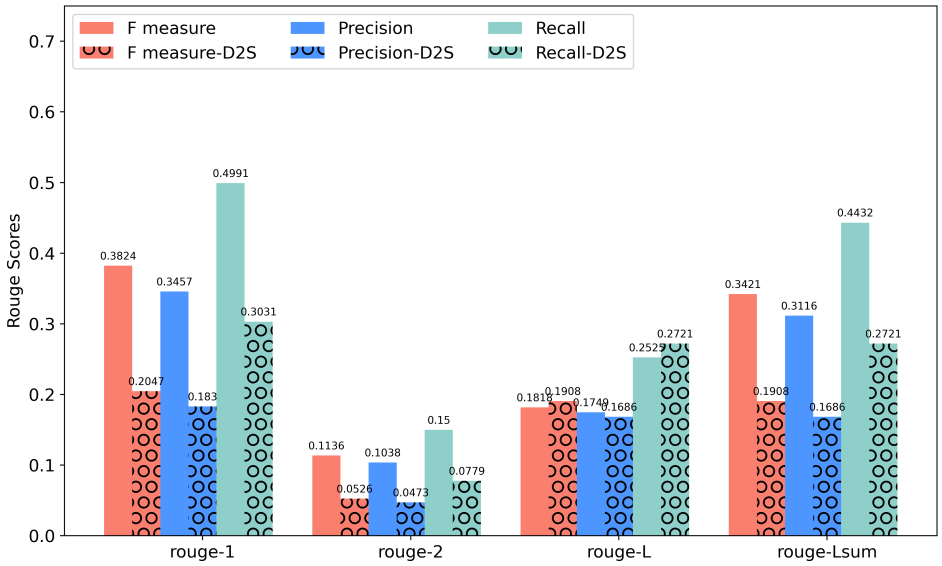


Figure 4.4: Comparison of test results between our model and D2S model. The result from the D2S model is textured with little circular markings.

In Figure 4.4, we observe that our model's Rouge-1 and Rouge-2 scores are notably higher than those of the D2S model. However, this doesn't hold for the Rouge-L score. This inconsistency intrigued us, prompting further investigation. Upon examining the code of the D2S model available on GitHub, we noticed that the authors might have utilized version 4.0.0 of the `nlp` library to calculate Rouge scores [LVdMJ+21]. Interestingly, a GitHub issue posted after the publication of D2S work, indicating a potential confusion in the implementation of Rouge-L and Rouge-Lsum scores within the `nlp` package. As a consequence, it's likely that the scores presented in the D2S work might actually represent the Rouge-Lsum score rather than Rouge-L. This discrepancy could provide an explanation for the inconsistency observed across all Rouge scores. It's important to highlight that the `nlp` package is no longer actively maintained, as its code has been integrated into another package under a different namespace. Unfortunately, due to this circumstance, it is uneasy to perform a definitive test to verify our hypothesis. Therefore, to ensure a rigorous approach, we have chosen to report both the Rouge-L and Rouge-Lsum scores in our analysis.

It's important to highlight that the D2S model leverages factual slide titles to generate slide content, which constitutes supplementary data in contrast to our approach. However, our comprehensive comparative study demonstrates that our outcomes con-

sistently outperform or are similar to the scores achieved by the D2S model across all metrics. This strong validation further highlights the effectiveness and superiority of our proposed approach over the state-of-art D2S model.

Chapter 5

Conclusion

In conclusion, this project has been driven by the aim of automating the process of generating presentation slides from paper documents. To achieve this goal, we have framed the problem as an optimization task centered around minimizing the global reconstruction loss. Drawing inspiration from the principles of autoencoder networks and unsupervised machine translation algorithms, we have meticulously designed an optimization framework tailored to address the unique challenges of this task. Through extensive automated evaluations, our system has demonstrated its superiority over baseline models, positioning itself as a benchmark for the evolving field of paper-to-slide generation. Looking ahead, we envision further enhancing the accessibility and usability of our solution by implementing a web-hosted application via the Huggingface Hub. This not only underscores the practical applicability of our approach but also underscores our commitment to fostering wider engagement and utilization within the academic and professional communities. Ultimately, our work signifies a substantial step forward in automating the process of turning scholarly material into visually captivating presentation slides. This has profound implications for improving how knowledge is shared and communicated effectively.

Bibliography

[Bac20] Andrew Bacon. Logical Combinatorialism. *The Philosophical Review*, 129(4):537–589, 10 2020.

[BB19] Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv:1904.00760*, 2019.

[BCB16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[BCLR22] Jawwad Baig, Guanyi Chen, Chenghua Lin, and Ehud Reiter. DrivingBeacon: Driving behaviour change support system considering mobile use and geo-information. In *Proceedings of the First Workshop on Natural Language Generation in Healthcare*, pages 1–8, Waterville, Maine, USA and virtual meeting, July 2022. Association for Computational Linguistics.

[Bel05] David A Bello. To go where no han could go for long: Malaria and the qing construction of ethnic administrative space in frontier yunnan. *Modern China*, 31(3):283–317, 2005.

[Ben09] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009.

[BMR+20] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[CD16] Christopher Clark and Santosh Divvala. Pdffigures 2.0: Mining figures from research papers. 2016.

[CF23] Nan Chen and Xianghui Fang. A simple multiscale intermediate coupled stochastic model for el niño diversity and complexity. *Journal of Advances in Modeling Earth Systems*, 15(4):e2022MS003469, 2023. e2022MS003469 2022MS003469.

[CGCB14] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[CJLZ22] Xi Chen, Wenbo Jing, Weidong Liu, and Yichen Zhang. Distributed estimation and inference for semi-parametric binary response models, 2022.

[CvD22] Guanyi Chen and Kees van Deemter. Understanding the use of quantifiers in Mandarin. In *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*, pages 73–80, Online only, November 2022. Association for Computational Linguistics.

[CvMG+14] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[FC23] Xianghui Fang and Nan Chen. Quantifying the predictability of enso complexity using a statistically accurate multiscale stochastic model and information theory. *Journal of Climate*, 36(8):2681 – 2702, 2023.

[FWMS22] Tsu-Jui Fu, William Yang Wang, Daniel McDuff, and Yale Song. Doc2ppt: Automatic presentation slides generation from scientific documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 634–642, 2022.

[GK18] Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Int. Res.*, 61(1):65–170, jan 2018.

[GRO23] Grobid. https://github.com/kermitt2/grobid, 2008–2023.

[HBD+20] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020.

[HBFC19] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *CoRR*, abs/1904.09751, 2019.

[HCX+22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022.

[HKG+15] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701, 2015.

[HLC+22] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 4083–4091, New York, NY, USA, 2022. Association for Computing Machinery.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[HW15] Yue Hu and Xiaojun Wan. Ppsgen: Learning-based presentation slides generation for academic papers. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):1085–1097, 2015.

[iK11] Ulr ich Krähmer. Notes on koszul algebras. 2011.

[JHG00] Nathalie Japkowicz, Stephen José Hanson, and Mark A. Gluck. Nonlinear Autoassociation Is Not Equivalent to PCA. *Neural Computation*, 12(3):531–545, 03 2000.

[Kes07] Srinivasan Keshav. How to read a paper. *ACM SIGCOMM Computer Communication Review*, 37(3):83–84, 2007.

[KKD+17] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation, 2017.

[KKM+19] Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551, Hong Kong, China, November 2019. Association for Computational Linguistics.

[Kra91] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[LCDR18] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only, 2018.

[LHML21] Da-Wei Li, Danqing Huang, Tingting Ma, and Chin-Yew Lin. Towards topic-aware slide generation for academic papers with unsupervised mutual learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13243–13251, 2021.

[Lin04] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[LLG+19] 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. *CoRR*, abs/1910.13461, 2019.

[LMS23] Xipeng Liu, Siyu Mei, and Joana Falcão Salles. Do inoculated microbial consortia perform better than single strains in living soil? a meta-analysis. *bioRxiv*, 2023.

[LOC+18] Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Phrase-based & neural unsupervised machine translation, 2018.

[LOG+19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[LVdMJ+21] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[MBB21] Andrei Manolache, Florin Brad, and Elena Burceanu. Date: Detecting anomalies in text via self-supervision of transformers. *arXiv preprint arXiv:2104.05591*, 2021.

[PSC+22] Zhaoqin Peng, Zhengyi Sun, Juan Chen, Zilong Ping, Kunyu Dong, Jia Li, Yongling Fu, and Enrico Zio. A fault diagnosis approach for electromechanical actuators with simulating model under small experimental data sample condition. *Actuators*, 11(3), 2022.

[PXS17] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization, 2017.

[RSR+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

[Rus20] Bertrand Russell. *Principles of mathematics*. Routledge, 2020.

[SCVD22] Fahime Same, Guanyi Chen, and Kees Van Deemter. Non-neural models matter: a re-evaluation of neural referring expression generation systems. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5554–5567, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[SHW+21] Edward Sun, Yufang Hou, Dakuo Wang, Yunfeng Zhang, and Nancy XR Wang. D2s: Document-to-slide generation via query-based text summarization. *arXiv preprint arXiv:2105.03664*, 2021.

[SII05] Mostafa Shaikh, Mitsuru Ishizuka, and Md Tawhidul Islam. 'auto-presentation': a multi-agent system for building automatic multi-modal presentation of a topic from world wide web information. pages 246– 249, 10 2005.

[SL23] Yue Shen and Xueyao Luo. Linking spatial and temporal contexts to multi-contextual segregation by hukou status in urban china. *Journal of Transport Geography*, 107:103540, 2023.

[SLM17] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[SMWG21] Athar Sefid, Prasenjit Mitra, Jian Wu, and C Lee Giles. Extractive research slide generation using windowed labeling ranking. In *Proceedings of the Second Workshop on Scholarly Document Processing*, pages 91–96, Online, June 2021. Association for Computational Linguistics.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[TFH+22] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas,

Claire Cui, Marian Croak, Ed Chi, and Quoc Le. Lamda: Language models for dialog applications, 2022.

[TLI+23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[VSP+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.

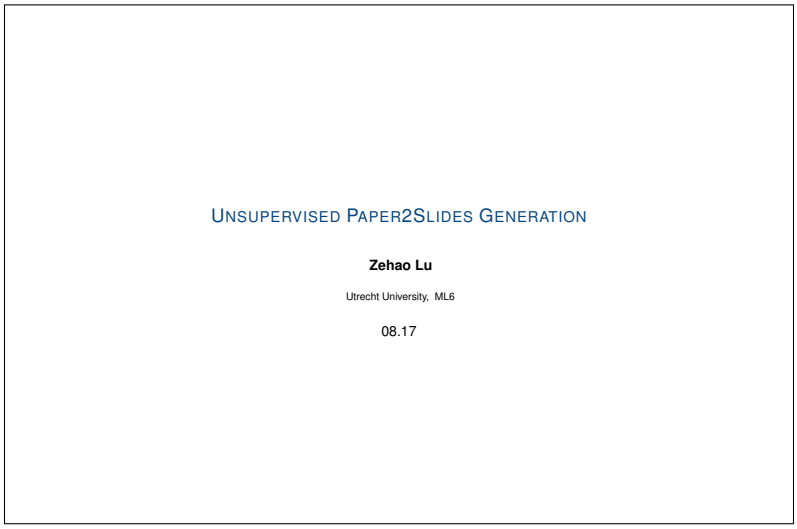
[ZCLS22] Yinhe Zheng, Guanyi Chen, Xin Liu, and Jian Sun. MMChat: Multi-modal chat dataset on social media. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5778–5786, Marseille, France, June 2022. European Language Resources Association.

[ZFW+23] Shicheng Zheng, Yongling Fu, Deyi Wang, Ziyu Liu, Junlin Pan, and Juan Chen. Effect of structural parameters and load states on ptrb's load distribution. *Advances in Mechanical Engineering*, 15(3):16878132231161004, 2023.

Appendix A

Example Slides Outputs

In this section, we have included the slide deck that was generated based on **this** paper. To create the slides, a modified version of the thesis was used, wherein we excluded the part of the literature review section and the examples subsections. This was done to manage the overall length of the generated slides effectively. The slide generation process was parameterized with 'minpage = 10' and 'maxpage = 30', ensuring that the resulting slide deck contains between 10 to 30 pages.



ABSTRACT

- There has been a scarcity of research into automating the "paper to slides" generation task, and a lack of publicly available datasets.
- We propose an inventive optimization framework based on reconstruction loss, harnessing cutting-edge Large Language Models (LLMs) and unsupervised learning.
- This approach facilitates the creation of high-quality slide decks from scientific papers, offering heightened adaptability and flexibility.

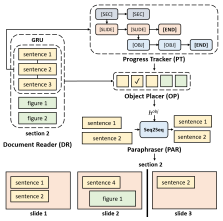
INTRODUCTION

- Paper2Slides software is composed of three core elements: a pre-processing unit that reads and loads data from .pdf files, the Paper2Slides model responsible for generating slide content, and a post-processing unit in charge of arranging the layout and producing the slides.
- Each sub-model within the Paper2Slideds model has undergone a comprehensive training process, which includes both pre-training and fine-tuning phases.

PAPER-TO-SLIDE GENERATION

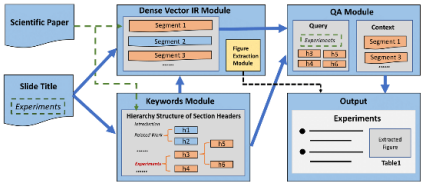
- The challenge of producing slides to accompany academic papers has not been thoroughly investigated and poses a significant difficulty.
- Earlier researcher mainly focuses on rule-based methods, their works focus on selecting the most important sentences and re-organize them.
- More recent approaches such as the D2S model and the DOC2PPT model are using neural networks on both sentence selection and summarization tasks.

DOC2PPT



- DOC2PPT model is composed of multiple blocks, including the document reader, progress tracker, object placer, and paraphraser.
- One of the major drawbacks of the model is that the paragrapher block can only rephrase sentences but not paragraph-level summarization.
- In different contexts, the appropriate segmentation of a given piece of text can vary significantly. DOC2PPT framework's RNN-based Document Reader struggles with long-range dependencies, making it hard to fully understand the bigger picture for accurate segmentation.

D2S

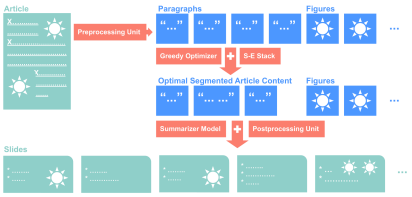


- D2S model operates within a framework centered on closed-domain question answering.
- Users input desired slide titles, prompting IR module to identify relevant keywords by comparing word embeddings based similarity and Levenshtein distance.
- These keywords are then matched with section titles using the keyword module, which subsequently extracts content from corresponding sections.
- QA module is applied to generate the actual slide content.

Supervised Approach Limitations

- An important decision we made during our model's implementation process was to utilize unsupervised learning method, even though there are a few accessible datasets for the paper-to-slide generation task.
- ~~We perform a comparative analysis of various training frameworks used in prior research and contrast them with our approach.~~
- Limited size of publicly available publicly available dataset poses challenge for training modern large language model (LLM) powered software from scratch.
- Modern LLMs commonly encompass more than 300 million parameters, while existing paper-to-slide datasets consist of 1000 to 10000 training data pairs.
- The paired paper-slides dataset does not give paragraph-'bullet point' level matching labels.
- The 'content generation' units are trained on noisy labeled dataset.

METHOD



- The PDF reader utilized in this project consisted of two primary phases: text processing (GROBID) and detection of figure-like objects (pdffigures2).
- Paper2Slides model serves as core component of the proposed software.
- Paper2Slide model produces textual slide content, which is inserted into a predefined template along with figure-like objects.
- Post-processing unit plays a crucial role in converting the content generated by our system into a finalized PDF format.

PAPER-TO-SLIDE PROBLEM

- ~~To tackle the limitation of the existing works, we approach the task of paper-toslide paper-to-slides generation as an optimization problem focused on minimizing information reconstruction loss.~~
- ~~We start by formulating a redefined research question and subsequently progress through a sequence of structured steps to arrive at our proposed solution.~~

TASK RE-INTRODUCTION

- Previous research has suggested that this task comprises two distinct components: structural decomposition (segmentation) and slides' content generation (summarization) We propose a different perspective.
- To tackle the limitation of the existing works, we approach the task of ~~paper-toslide~~ paper-to-slides generation as an optimization problem focused on minimizing information reconstruction loss.
- We start by formulating a redefined research question and subsequently progress through a sequence of structured steps to arrive at our proposed solution.
- We claim that the essence of the paper-to-slides generation task lies in extracting the most valuable information and presenting it in the form of slides.

SUMMARIZER-EXPANDER STRUCTURE

- How can we effectively measure quantity of information embedded in textual language?
- To answer this question, we turn to summarization models as valuable tools, which serve as useful indicators of the most essential information within a text.

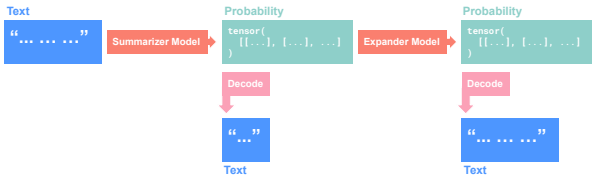
SUMMARIZER-EXPANDER STRUCTURE

- Summarization tasks can be viewed as an effort to capture the main ideas and concisely articulate them in human language.
- Texts with high information density present a greater challenge for summarization compared to texts with lower information density, even when both texts share the same length.
- The varying quantities of information present in the texts affect human understanding, thus influencing the lengths and quality of summaries generated by both the AI model and humans.

SUMMARIZER-EXPANDER STRUCTURE

- We extend our focus to the expander models.
- The expander model is a functional construct capable of generating longer text based on the information contained in a shorter text.
- The expander model incorporates information that is inferred from or logically related to the original text, thereby augmenting the quality of the output.

Reconstruction Loss

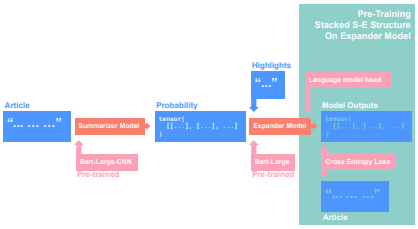


- reconstractionLoss(t) = CrossEntropy(E(S(t)), ~~E([?]dot)~~, t) where cross entropy loss is a measure of ~~the cross entropy of a piece of~~ the difference between text t, and S-E pair ~~S(•), E[?]dot~~ output E(S(t)).
- Both the summarizer and the expander are language models, therefore, their outputs are probabilities, hence we could use cross entropy to measure the difference.
- Figure shows how the reconstruction loss is computed.
- With the S-E stack at hand, we could now reformulate the problem of 'extracting the most valuable information and presenting it in the form of slides' in mathematical form $seg^* = argmin_{seg \in S} \sum^{t_i} reconstructionLoss(t_i)$.
- Without proof, we claimed that for a certain S-E pair, the more coherent paragraphs would show a smaller reconstruction loss compared to a text that displays more significant logical inconsistency.

Paper2Slides Model

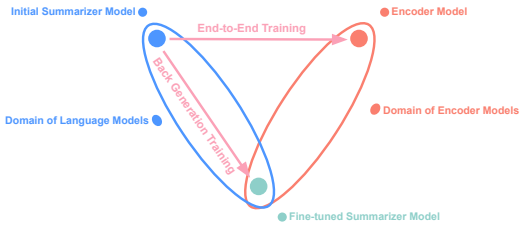
- In this section, we delve into the specifics of implementing and training submodels within the Paper2Slides framework.
- We provide a comprehensive account of the greedy algorithm utilized to address the optimization challenge of the reconstruction loss.
- We also discuss the pre-training of the S-E stack performed on the expander model, textcolorblueplus the fine-tuning of both summarizer and expander models.

Pre-training S-E Stack



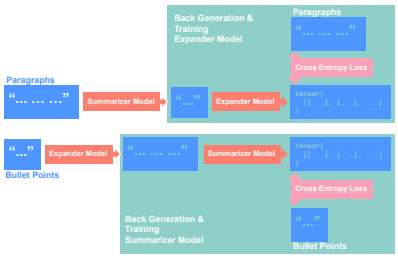
- The S-E stack plays a crucial role in computing the reconstruction loss and achieving a more convincing measurement of information quantity.
- To ensure maximum effectiveness, both the summarizer model and the expander model need to be highly powerful.
- Since there were no pre-trained expanders available, we took on the task of pre-training one from scratch.
- For both summarization and expanding tasks, we utilized BART model as the underlying neural network structure.

FINE-TUNING S & E



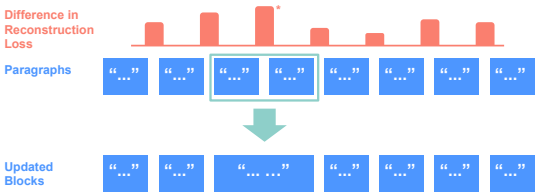
- End-to-end training of the S-E stack is not possible,

FINE-TUNING S & E



- End-to-end training of the S-E stack is not possible, so we use an unsupervised training algorithm discussed in the literature review section.
- We utilize this algorithm to parallelly train both the summarizer and the expander models.
- During each iteration of training, both models were utilized to summarize or expand some given piece of text.
- The generated text and its source text are used as training input and the label to train the other model.

GREEDY OPTIMIZER

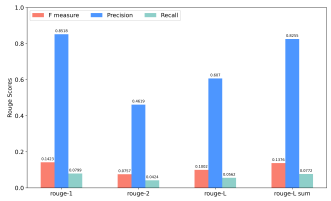


- where the symbol "seg" represents a segmentation method and t_i are segmented blocks from the input article.
- We are unable to derive a precise mathematical formula for the optimal segmentation, denoted as seg * * .
- In light of this, we need to turn to iterative optimization methods that can dynamically adjust and adapt, rather than relying on fixed procedures.
- We opted to utilize a greedy search algorithm to uncover the solution, in order to take some advantage of its efficiency in development and robustness in deployment

Experiments

- We offer an extensive assessment of the Paper2Slides framework, encompassing evaluations of the main model and its sub-components.
- These evaluations involve multiple dimensions, including gauging the performance of the pretrained expander model on the CNN-daily dataset.
- We provide illustrative examples showcasing the outputs of the individual sub-models.

Pre-trained Expander Assessment

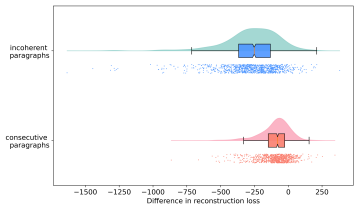


- We used the ROUGE score as an automated assessment tool, which is a commonly employed metric that assesses the quality of generated text by measuring the degree of token or n-gram overlap lin-2004-rouge.
- For each highlight in the test dataset, we employed the expander model to extend it into longer text, aiming for a length equivalent to the corresponding article.
- Subsequently, the Rouge score is computed between the expanded text and the ground truth article.
- ~~The~~

Fine-tuned Models Assessment

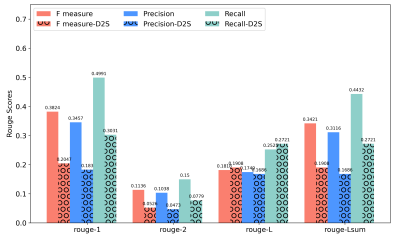
- Our experiments focused on 75 randomly selected paragraphs from 18 scientific articles, most of them are published in scientific journals and else are master's or PhD theses.
- We were unable to find a baseline expander model, leading us to exclusively carry out evaluations on the summarizer model.
- The ultimate findings reveal that in 45.33% of instances, providers favored the result from our fine-tuned model. In 28.00% of cases, providers perceived both summaries as having equal quality, while in 26.67% of cases, providers favored the baseline model's result.
- The consistent preference for our model's summaries by providers strongly supports the conclusion that our fine-tuned model outperforms the baseline.

Property of Reconstruction Loss



- In this section, we provide evidence to support the above-mentioned claim through a verification test.
- We claimed that for a certain S-E pair, the more coherent paragraphs would show a smaller reconstruction loss compared to a text that displays more significant logical inconsistency.
- This suggests that the optimization framework based on reconstruction loss could potentially address the segmentation challenge inherent in the paper-to-slide generation process.
- We carefully selected 2000 pairs of coherent and consecutive paragraphs from the dataset 'Automatic

System Assessment



- In this section, we present the outcomes of our testing in comparison to existing state-of-art models.
- Initially, our intention was to assess our model's performance against both the DOC2PPT and D2Smodels.
- However, we encountered a limitation with the DOC 2PPT dataset, without the corresponding extracted text.
- On the other hand, we solely conduct the comparison against the D2S model.
- model's Rouge-1 and Rouge-2 scores are significantly higher than the D2S model.

Conclusion

- This project has been driven by the aim of automating the process of generating presentation slides from paper documents.
- We have meticulously designed an optimization framework tailored to address the unique challenges of this task.
- Through extensive automated evaluations, our system has demonstrated its superiority over baseline models, positioning itself as a benchmark for the evolving field of paper-to-slide generation.