pdf-summarization

May 2, 2024

[1]:
```
from IPython.display import HTML, display

def set_css():
  display(HTML('''
  <style>
    pre {
        white-space: pre-wrap;
    }
  </style>
  '''))
get_ipython().events.register('pre_run_cell', set_css)
```

[11]:
```
!pip install langchain
!pip install transformers
!pip install PyPDF2
!pip install sumy
```

```
<IPython.core.display.HTML object>
```

```
Requirement already satisfied: langchain in /usr/local/lib/python3.10/dist-
packages (0.1.17)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-
packages (from langchain) (6.0.1)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in
/usr/local/lib/python3.10/dist-packages (from langchain) (2.0.29)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in
/usr/local/lib/python3.10/dist-packages (from langchain) (3.9.5)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in
/usr/local/lib/python3.10/dist-packages (from langchain) (4.0.3)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in
/usr/local/lib/python3.10/dist-packages (from langchain) (0.6.5)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in
/usr/local/lib/python3.10/dist-packages (from langchain) (1.33)
Requirement already satisfied: langchain-community<0.1,>=0.0.36 in
/usr/local/lib/python3.10/dist-packages (from langchain) (0.0.36)
Requirement already satisfied: langchain-core<0.2.0,>=0.1.48 in
/usr/local/lib/python3.10/dist-packages (from langchain) (0.1.48)
Requirement already satisfied: langchain-text-splitters<0.1,>=0.0.1 in
```

```
/usr/local/lib/python3.10/dist-packages (from langchain) (0.0.1)
Requirement already satisfied: langsmith<0.2.0,>=0.1.17 in
/usr/local/lib/python3.10/dist-packages (from langchain) (0.1.52)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-
packages (from langchain) (1.25.2)
Requirement already satisfied: pydantic<3,>=1 in /usr/local/lib/python3.10/dist-
packages (from langchain) (2.7.1)
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.10/dist-
packages (from langchain) (2.31.0)
Requirement already satisfied: tenacity<9.0.0,>=8.1.0 in
/usr/local/lib/python3.10/dist-packages (from langchain) (8.2.3)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain)
(1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp<4.0.0,>=3.8.3->langchain) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain)
(1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp<4.0.0,>=3.8.3->langchain)
(6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp<4.0.0,>=3.8.3->langchain) (1.9.4)
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in
/usr/local/lib/python3.10/dist-packages (from dataclasses-
json<0.7,>=0.5.7->langchain) (3.21.2)
Requirement already satisfied: typing-inspect<1,>=0.4.0 in
/usr/local/lib/python3.10/dist-packages (from dataclasses-
json<0.7,>=0.5.7->langchain) (0.9.0)
Requirement already satisfied: jsonpointer>=1.9 in
/usr/local/lib/python3.10/dist-packages (from jsonpatch<2.0,>=1.33->langchain)
(2.4)
Requirement already satisfied: packaging<24.0,>=23.2 in
/usr/local/lib/python3.10/dist-packages (from langchain-
core<0.2.0,>=0.1.48->langchain) (23.2)
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in
/usr/local/lib/python3.10/dist-packages (from
langsmith<0.2.0,>=0.1.17->langchain) (3.10.2)
Requirement already satisfied: annotated-types>=0.4.0 in
/usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1->langchain) (0.6.0)
Requirement already satisfied: pydantic-core==2.18.2 in
/usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1->langchain)
(2.18.2)
Requirement already satisfied: typing-extensions>=4.6.1 in
/usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1->langchain)
(4.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
```

```
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests<3,>=2->langchain) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2->langchain)
(2024.2.2)
Requirement already satisfied: greenlet!=0.4.17 in
/usr/local/lib/python3.10/dist-packages (from SQLAlchemy<3,>=1.4->langchain)
(3.0.3)
Requirement already satisfied: mypy-extensions>=0.3.0 in
/usr/local/lib/python3.10/dist-packages (from typing-
inspect<1,>=0.4.0->dataclasses-json<0.7,>=0.5.7->langchain) (1.0.0)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-
packages (4.40.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from transformers) (3.13.4)
Requirement already satisfied: huggingface-hub<1.0,>=0.19.3 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.20.3)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from transformers) (1.25.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (23.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2023.12.25)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.20,>=0.19 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.19.1)
Requirement already satisfied: safetensors>=0.4.1 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.4.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-
packages (from transformers) (4.66.2)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub<1.0,>=0.19.3->transformers) (2023.6.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub<1.0,>=0.19.3->transformers) (4.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7)
```

```
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2024.2.2)
Requirement already satisfied: PyPDF2 in /usr/local/lib/python3.10/dist-packages
(3.0.1)
Collecting sumy
  Downloading sumy-0.11.0-py2.py3-none-any.whl (97 kB)
                           97.3/97.3 kB
3.2 MB/s eta 0:00:00
Collecting docopt<0.7,>=0.6.1 (from sumy)
  Downloading docopt-0.6.2.tar.gz (25 kB)
  Preparing metadata (setup.py) … done
Collecting breadability>=0.1.20 (from sumy)
  Downloading breadability-0.1.20.tar.gz (32 kB)
  Preparing metadata (setup.py) … done
Requirement already satisfied: requests>=2.7.0 in
/usr/local/lib/python3.10/dist-packages (from sumy) (2.31.0)
Collecting pycountry>=18.2.23 (from sumy)
  Downloading pycountry-23.12.11-py3-none-any.whl (6.2 MB)
                           6.2/6.2 MB
21.0 MB/s eta 0:00:00
Requirement already satisfied: nltk>=3.0.2 in
/usr/local/lib/python3.10/dist-packages (from sumy) (3.8.1)
Requirement already satisfied: chardet in /usr/local/lib/python3.10/dist-
packages (from breadability>=0.1.20->sumy) (5.2.0)
Requirement already satisfied: lxml>=2.0 in /usr/local/lib/python3.10/dist-
packages (from breadability>=0.1.20->sumy) (4.9.4)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.0.2->sumy) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.0.2->sumy) (1.4.0)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk>=3.0.2->sumy) (2023.12.25)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.0.2->sumy) (4.66.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.7.0->sumy) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests>=2.7.0->sumy) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.7.0->sumy) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.7.0->sumy) (2024.2.2)
Building wheels for collected packages: breadability, docopt
  Building wheel for breadability (setup.py) … done
  Created wheel for breadability: filename=breadability-0.1.20-py2.py3-none-
any.whl size=21693
sha256=9c14c49915be3bbeed44d167c935a3a164d1779da6e8b0346d5b321368bb847b
  Stored in directory: /root/.cache/pip/wheels/64/22/90/b84fcc30e16598db20a0d413
```

```
40616dbf9b1e82bbcc627b0b33
  Building wheel for docopt (setup.py) … done
  Created wheel for docopt: filename=docopt-0.6.2-py2.py3-none-any.whl
size=13706
sha256=966cb9f4412054c15b5ee320053c61b5911bae9c419b63d9524fbc06a68f258e
  Stored in directory: /root/.cache/pip/wheels/fc/ab/d4/5da2067ac95b36618c629a5f
93f809425700506f72c9732fac
Successfully built breadability docopt
Installing collected packages: docopt, pycountry, breadability, sumy
Successfully installed breadability-0.1.20 docopt-0.6.2 pycountry-23.12.11
sumy-0.11.0
```

[2]:
```python
from huggingface_hub import login

from google.colab import userdata
api_token = userdata.get('HuggingFace')

login(token=api_token)
```

```
<IPython.core.display.HTML object>
```

```
Token will not been saved to git credential helper. Pass
`add_to_git_credential=True` if you want to set the git credential as well.
Token is valid (permission: read).
Your token has been saved to /root/.cache/huggingface/token
Login successful
```

1 Extractive + Abstractive

[12]:
```python
import PyPDF2
from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer
from sumy.summarizers.text_rank import TextRankSummarizer
from langchain_text_splitters import RecursiveCharacterTextSplitter
import nltk

nltk.download('punkt')
```

```
<IPython.core.display.HTML object>
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

[12]:
```
True
```

[13]:
```python
# Function to extract text from the PDF
def extract_text_from_pdf(pdf_path):
    text = []
    try:
        with open(pdf_path, 'rb') as pdf_file:
            reader = PyPDF2.PdfReader(pdf_file)
            num_pages = len(reader.pages)
            # Extract text from each page and append to the list
            text = [reader.pages[page_num].extract_text() for page_num in␣
 ↪range(num_pages)]

    except Exception as e:
        print(f"Error occurred while extracting text from PDF: {e}")
        return None

    return ''.join(text)

text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1000,
    chunk_overlap=200,
    length_function=len,
    is_separator_regex=False,
)
```

```
<IPython.core.display.HTML object>
```

[14]:
```
def summarize_chunk(chunk):

    parser = PlaintextParser.from_string(chunk, Tokenizer("english"))

    summarizer = TextRankSummarizer()
    summary_sentences = summarizer(parser.document, 3)

    summary = " ".join([str(sentence) for sentence in summary_sentences])
    return summary
```

```
<IPython.core.display.HTML object>
```

[15]:
```
def summarize_pdf(file_path):

    text = extract_text_from_pdf(file_path)
    chunks = text_splitter.split_text(text)
    summarizer = TextRankSummarizer()
    summaries = [summarize_chunk(chunk) for chunk in chunks]
    full_summary = "\n".join(summaries)

    return full_summary
```

```
<IPython.core.display.HTML object>
```

```
[16]: pdf_file_path = "/content/Efficient Summarization of Long Documents.pdf"

      # Run the main function and print the summary
      extractive_summary = summarize_pdf(pdf_file_path)
      print("Summary:")
      print(extractive_summary)
```

```
<IPython.core.display.HTML object>

Summary:
DEIM Forum 202 3 1b-5-2 Efficient Summarization of Long Documents Using Hybrid
Extractive -Abstractive Method Weichao  CHEN†   Mizuho  IWAIHARA‡ Graduate
School of Information, Production and Systems, Waseda University 2-7 Hibikino,
Wakamatsu -ku, Kitakyushu, Fukuoka, 808 -0135 Japan E-mail:
†vico_chen@akane.waseda.jp,  ‡iwaihara@waseda.j p Abstract   Automatic
summarizers based on pre -trained language models (PLMs) have achieved great
success in the field of short text summarization. However, the com plexity of
the self -attention mechanism on which PLMs rely grows quadratically with the
input length, thus limiting its application to long document summarization. As a
solution, we investig ate an efficient hybrid long  document summarization
method that en ables compression and summarization of original long documents by
combining content selection and abstract ive summarization.
Long documents will first be extracted for  salient sentences and reorganized
into compressed documents, the length of which is within the limits of the PLMs
, and finally generate  the final summary  by abstractive summarization.
Introduction A summary is a concise and comprehensive version of the original
document  . The task of generating a summary from the original document is called
text summarization , which can be further divided into extractive and
abstractive summarization s [1].
Extractive summarization methods, such as text ranking and clustering, work by
selecting and compiling parts of the original text, while abstractive
summarization generate s a new summary that captures the main ideas and concepts
in the original tex t, where passages not appearing in the original document may
be generated . Abstract ive summarization  has the advantage of smoothness and
fluency, while keeping the content information , so our discussion will mainly
focus on abstract ive summarization  [6]. Typical abstract ive summarization
algorithms use seq2seq -based model s such as transformer s. These algorithms
have had great success in the field of short text summ arization.
However, the self -attent ion mechanism of transformer -based models has a
quadratic level of  space complexity, leading to a dramatic increase in the
computational consumption of these models as the number of input words
increases. So, usually thes e models have a limit on the number of input tokens
, such as 1024 tokens limit in BART  [10] . For relatively long documents, such
as press releases, scientific literature, specialized reports, they often have
lengths that far exceed these limits, making it impossible to use the entir e
content  as input of these models.
In terms of reason analysis, pre-trained language models cannot be applied to
long documents mainly because of the computational c ost of full self -attention
```

in the transformer structure. The most straightforward solution is to improve the self -attention mechanism to make it more efficient. Currently, the most popular approach is to replace the full attention with sparse attention mec hanisms such as local attention and sliding windows, which allows to focus on specific parts of the input while also reducing computational complexity.
For example, Longformer incorporates sliding window attention, dilated sliding windows, and global + sliding windows, which enables the processing of long documents with more than 4096 token s in length. Hybrid summarization is another important approach for long document summarization t hat combines content selection and abstractive summarization. This approach does not change the structure of the original pre -trained language model (PLM); instead, it solves the long document problem by changing the input of the PLMs.
In this approach, lo ng documents are first processed through a content selection process, where the most beneficial sentences or paragraphs for generating a summary are selected and reorganized into a new, compressed document that is long enough for downstream summarization w ithin the limits of the PLMs. For example, Bajaj et al. [2] investigated a combined BERT [4] and BART model for long document summarization, using BERT as a salient sentence classifier to filter the source document sentences and BART as a summarizer for the summarization task. For example, the reduced complexity of these mechanisms may come at the cost of summary quality to the original document.
Additionally, models using efficient attention mechanisms, such as Longformer and Big Bird, may still struggle with high resource consumption and difficulty in capturing the overal l context of the document. An example is the two -stage model, CogLTX [5] proposed by Ding et al. which achieves SOTA results on long document classification and Q&A tasks by using key sentence selection and downstream models. Based o n the successful precedent of CogLTX for long document classification and Q&A tasks, in this paper, we further explore its application to long document summarization .
Our extension of CogLTX to long document summarization can be summarized in 3 parts - (1) we utilize BART to process the downstream summarization task in place of BERT ; (2) a compression mechanism based on the combination of similarity scoring and latent intervention is introduced ; (3) more initialization methods specific to the summarizatio n task are utilized in the model. In summary, t he main contribution of this paper is to propose a hybrid summarization model , which combines extractive and abstractive summarization s by conducting co-training of the content selection model BERT and the abstractive summarization model BART. From the results, we can conclude that this model has good summarization results for both general long documents and extreme long documents .
Compared to BigBird and Longformer, which rely on GPU clusters, our model effect has a 1.5 ROUGE scores improvement and outperforms the other models in BERTScore. Related Work Long document summarization using hybrid models is an emerging research topic, which is highly related to abstractive summarization and extractive content selection methods such as sentence similarity calculation . Traditional methods include Tf -Idf and TextRank, but more recent deep learning -based methods such as Bi -Encoder and Cross -Encoder tend to be more accurate for similarity calculation .

Structures of Bi -Encoder and Cross -Encoder Bi-Encoder is a simple encoder architecture using BERT with separate encoders but shared parameters for input and output sequences, while Cross -Encoder combines the sequences into a single sequence representation. SimCSE [7] is particularly effective for learning sentence e mbeddings that capture semantic similarity, as demonstrated by its SOTA performance on a variety of benchmark tasks, including semantic textual similarity, natural language inference, and text classification. One of the advantages of SimCSE is that it does not require any labeled data and can be trained on large amounts of

One of the advantages of SimCSE is that it does not require any labeled data and can be trained on large amounts of unlabeled text. BART (Bidirectional Encoder Representations from Transformers) is a sequence -to-sequence language model that has been used for tasks such as text generation, translation, and summarization. The architecture of BART is a combination of BERT as the encoder and GPT [12] as the decoder which allows captur ing long -range dependencies and contextual information effectively.

the encoder and GPT [12] as the decoder which allows captur ing long -range dependencies and contextual information effectively. 2.3 CogLTX CogLTX [5] is a two -stage model that partitions long documents into blocks, scores each block with a judge model, and then combines the high -scoring blocks for training downstream tasks . The model utili zes two critical mechanisms , one is MemRecall, which works by splitting documents into blocks, connecting each block with available key information, and obtaining the average similarity score of the blocks by a scoring machine, then "forgetting" the blocks that did not score well, and repeating the process multiple times to complete multi -step reasoning.

The second one is the joint training of two BERTs, that is why this model has two -stage processing. In summary, this model has shown outstanding performanc e on classification and Q&A tasks over long documents , that inspire us to think about the possibilities on long document summarization task. It is important to emphasize that our mode l discards the notion of "natural guidance" in CogLTX because there is no natural prompt for summarization task.

For the training process, long documents are first dynamically split into blocks that represent the minimum granularity of com pression. The positive and negative sample blocks based on the relevance are used as training data to train the BERT scorer, so that we can obtain the score of each block of the original long document. Based on these scores, it is then possible to determine which blocks in the source document are the most promising, and the highest scoring blocks are selected and reorganized in their order in the original document to produce compressed do cuments that are

Finally, through a "latent intervention" method, the blocks of the compressed document are again checked to confirm that they are useful for generating summa ries and reducing the loss of BART summarizer. The top-K blocks with the highest scores are selected and reorganized into compressed documents in their order in the document . The compressed document is passed to the BART summarizer to produce the final summary.

The length limit of the block determines the minimum granularity of the model input; too small a block may cause semantic information to be fragme nted and scattered, while too large a block may cause detailed semantic information to be lost. For a long document, the document is first segmented based on punctuation, and if a piece of text exceeds , it is truncated and segmented, resulting in a significant cost. Initial segmentation has variable block lengths, a better approach is to dynamically aggregate the smaller blocks again, ensuring that there is the smallest possible sum of costs without t he aggregated blocks exceeding in

3.2 Initialization Initialization represents the pre -processing of blocks split from a long document with the purpose of dete rmining the relevance of the blocks before the training process begins . The relevance reflects the importance of the block in the long document and represents the degree of influence of this block on the summary's generation. The simplest and most efficien t way to determine the relevance of a block is by computing the semantic similarity between the target summary and each block.

In our proposed model, we use each of them to compute the similarity between each block of a long document and the corresponding summary. 3.3 BERT and BART training After obtaining the initialized positive and negative sample blocks, we can construct the dataset used to train BERT. To ensure the integrity of the learning content, we use a dataset that consists of short spans randomly extracted from long documents, along with other spans made up of positive and negative samples.

After collecting BERT scores for each block of the long document, the highest scoring blocks are collected, reorganized in their original order within the lon g document, which is referred to as document compression in this paper and represents the end of the first stage . The length of the compressed document is less than the maximum length of 1024 limited by BART, but it contains the best information of the ori ginal document, so it can be used to perform the summarization task on behalf of the original document. The second stage is fine -tuning BART using the compressed document set.

Specifically, if a block in a compressed document is discarded causing an increase in the loss of the BART summarizer, this means that we better not lose it and it is relevant . Conversely, if a block in a compressed document is discarded causing a decrease in the loss of the BART summarizer, this means that the block played a bad role in the generation of the summary, and it is probably irrelevant. This allows us to reposition the relevance of these blocks based on the results of the latent intervention in order to corr ect our initial judgments and better help the training of the next epoch.

As shown in Figure 3, the compression process is based on a multi -layer d ynamic compression mechanism. Specifically, the number of blocks contained in the compressed document is limited to a maximum of due to the limitation of the BART input, where also represents the size of the working memory. It is well known that in the human brain, the data in working memory can be cleared, updated, or added at any time.

The compressed document is filled step by step through multiple memory iteration cycles, each of wh ich dynamically updates and retains a certain number of the most valuable memories based on the prediction scores. Only three cycles are demonstrated in the figure, with the number of memory blocks retained in each cycle being 3, 3, and 10, corresponding t o the blue, red, and green blocks in

the figure, respectively. The abstractive summarization process is relatively simple, and the length of the compressed document obtained by multi -layer dynamic compression is kept within the maximum limit of BART, which can directly generate a target summary corresponding to the original long document.
Datasets and Experiments A good long document summarization model should be adaptable to documents of various lengths. PubMed is a database of medical literature, including abstracts and full -text articles from a wide range of medical journals. The articles in this dataset may be suitable for long document summarization in the field of medicine.
arXiv contains electronic preprints of scientific papers in a variety of fields, including mathematics, computer science, physics, biology, finance, and statistics. These papers can be quit e long and suitable for long document summarization in the field of scientific research. These reports can vary in length and suitable for long document summarization in the field of government research and policy.
There are several variations of ROUGE, including ROUGE -N, which measures the overlap between the summary and the reference in terms of n -grams (sequences of n words), and ROUGE -L, which measures the longest common subsequence between the summary and the reference. ROUGE scores are typically used to evaluate the performance of summarization systems and to compare the quality of different summaries. However, ROUGE scores are calculated based on word frequenc ies, and it has its limitations in reflecting the true semantic content and information coherence.
It is based on the BERT language model and measures the similarity between the generated text and the reference text using a combination of precision and recall. BERTScore is designed to be less sensitive to the length of the generated text than other evaluation metrics, such as ROUGE, which can be useful when evaluating summaries of different lengths. BERTScore has been shown to be effective at evaluating the quality of summaries and other types of generated text.
The first type of methods is unsupervised extractive summarization methods, which mainly includes the most representative traditional word frequency statistics method Tf -Idf, as well as classical BERT -based extractive summarization methods. The summaries generated by these methods are derived from the sentence reorganization extracted from the source documents. The third type is hybrid summarization based on content selection and abstractive summarization, where LoBART [11] is a recognized representative work in this
The third type is hybrid summarization based on content selection and abstractive summarization, where LoBART [11] is a recognized representative work in this
area, and our work also belongs to this type. The batch size of the compressor was set to 32 and the learning rate was set to 2e -5. Since the training dataset of RoBERTa needed to be sufficiently large to conta in every block in the original long document, we used data enhancement by sampling the same document
training dataset of RoBERTa needed to be sufficiently large to conta in every block in the original long document, we used data enhancement by sampling the same document
multiple times for the training data. The batch size of the summarizer BART was set to 2 and the learning rate was set at a relatively

small 1e -5. In the latent intervention stage, the relevance of text blocks was rechecked based on a threshold value that increased the relevance of blocks only when the BART loss difference exceeded =0.15 and decreased the relevance o f blocks when it was less than =-0.07.

Experimental Results For different initialization strategies, we conducted comparative experiments, and the results are shown in Table 2 . Here we also introduce Rouge Similarity, which uses the ROUGE scores of the sentence pairs as their similar ity scores. It can be seen that the unsupervised trained SimCSE shows a slight advantage in the final summar ization results, so in the following experiments, we use the unsupervised trained SimCSE as the default initialization strategy.

Experimental results and comparison with other baseline models Datasets Pubmed arXiv GovReport R-1 R-2 R-L BERTScore R-1 R-2 R-L BERTScore R-1 R-2 R-L BERTScore Unsupervised Extractive Tf-Idf / / / / 34.4 9.5 28.5 0.822 / / / / Supervised Abstractive - End -to-End BART -only 42.1 17.5 37.1 / 41.3 15.3 36.8 0.846 52.8 19.1 49.9 / Longformer 46.9 20.2 42.8 / 46.7 19.6 41.8 0.865 / / / / BigBird 46.3 20.6 42.3 / 46.2 19.0 41.5 0.855 56.8 22.6 53.8 / Supervised Abstractive - Hybrid summarization LoBART 48.0 20.9 43.5 / 48.6 20.1 42.2 0.845 / / / / Our Proposed Model Ours 46.8 20.3 35.7 0.863 47.9 19.5 35.6 0.868 55.3 21.9 41.6 0.865 Table 3 shows the experimental results of our proposed model on three different long document datasets.

For the GovReport dataset which has the longest average length, our model also achieves a good result in R -1 and R-2 scores , it is slightly lower than the result of BigBird . For the Pubmed and arXiv datasets, LoBART has a relatively high ROUGE score, but for the arXiv dataset, our model has the highest BERTScore. However, our model has a relatively poor performance in terms of R -L scores compared to other models, as evidenced by the fact that our model generates summaries that have a small common subsequence with the target
in terms of R -L scores compared to other models, as evidenced by the fact that our model generates summaries that have a small common subsequence with the
target summary. Content selection can pick out the most salient sentences in the original document, however, the selected sentence s are often discrete, so it will cause the reduction of R -L. The decrease of R -L corresponds to the increase of BERTScore, and how to ensure the balance between the two is also a key point to be considered in our future work. Conclusion and Future Work For the long document problem that is often encountered when using traditional pre -trained language models for text summarization, we propose a hybrid long document summarization method.

extractive summarization models and efficient self-attention based long document adaptation mechanisms, our proposed model can achieve a balance in resource consumption and model performance, which is especially important for implementing long document summarization in single -GPU or low resource scenario s. Also, as an advantage of the hybrid summarization model, our model has an impressive performance in handling documents of any length, and even outperforms SOTA on specific domains. As an additional result, the summaries we generated are more closely ali gned with human judgment (See Appendix Table 5) , as demonstrated by the fact that our model obtained a higher BERTScore than other baseline models. For future work, we tend to further improve the performance of

the model, especially when dealing with extre mely long documents.
As a further addition to the CogLTX series of studies, we are hopeful to implement model migration and corresponding long -document adaptation in other domains including keyphrase extraction and generation, multi -label text classificati on, etc., to address the long -document problem faced by these domains as well. [2] Bajaj, Ahsaas, et al. "Long Document Summarization in a Low Resource Setting using Pretrained Language Models." "Longformer: The long -document transforme r." arXiv preprint arXiv:2004.05150 (2020).
[4] Devlin, Jacob, et al. "Bert: Pre -training of deep bidirectional transformers for language understanding." [8] Huang, Luyang, et al. "Efficient attentions for long document summarization." [9] Koh, Huan Yee, et al. "An Empirical Survey on Long Document Summarization: Datasets, Models, and Metrics." [10] Lewis, Mike, et al. "Bart: Denoising sequence -to-sequence pre -training for natural language generation, translation, and comprehension." arXiv preprint arXiv:2105.03801 (2021). arXiv preprint arXiv:1908.10084 (2019).
Sample summaries for arXiv dataset Target Summaries Generated Summaries additive models play an important role in semipara metric statistics . this paper gives learning rates for regularized kernel based methods for additive models . these learning rates compare favourably in particular in high dimensions to recent results on optimal learning rates for purely nonparametric reg ularized kernel based quantile regression using the gaussian radial basis function kernel , provided the assumption of an additive model is valid . nonparametric reg ularized kernel based quantile regression using the gaussian radial basis function kernel , provided the assumption of an additive model is valid . additionally , a concrete example is presented to show that a gaussian function depending only on one variab le lies in a reproducing kernel hilbert space generated by an additive gaussian kernel , but does not belong to the reproducing kernel hilbert space generated by the multivariate gaussian kernel of the same variance . This paper discusses the importance of additive models in semiparametric statistics and presents learning rates for regularized kernel -based methods for these models.
learning rates are compared favorably, particularly in high dimensions, to recent results on optimal learning rates for purely nonparametric regularized kernel -based quantile regression using the Gaussian radial basis function kernel, provided the assumption of an additive model is valid. The paper also provides a concrete example to demonstrate that a Gaussian function depending only on one variable lies in a reproducing kernel Hilbert space generated by an additive Gaussian k ernel but does not belong to the reproducing kernel Hilbert space generated by the multivariate Gaussian kernel of the same variance. The key terms and phrases discussed in the paper include additive model, kernel, quantile regression, semiparametric, rate of convergence, and support vector machine.
it is shown that for large amplitudes of phase fluctuations , a finite - time effect decreases the ability of phase diffuser to suppre ss the scintillations . The paper examines the impact of a random phase diffuser on fluctuations of laser light, specifically scintillations. The study shows that for large amplitudes of phase fluctuations, a finite -time effect reduces the ability of the phase diffuser to

```
phase variations for long propagation paths. The study shows that for large
amplitudes of phase fluctuations, a finite -time effect reduces the ability of
the phase diffuser to suppress the scintilla tions.
```

[17]:
```python
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

# Initialize the tokenizer and model
model_name = 'facebook/bart-large-cnn'
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
```

```
<IPython.core.display.HTML object>
```

[18]:
```python
# Function to summarize text using an abstractive model
def summarize_full_summary(full_summary):

    inputs = tokenizer(full_summary, return_tensors='pt', truncation=True,␣
 ↪padding=True, max_length=1024)

    # Generate an abstractive summary
    outputs = outputs = model.generate(
        inputs.input_ids,
        max_length=2000,
        min_length = 1000,
        num_beams=4,  # Number of beams for beam search (can improve quality)
        length_penalty=2.0,  # Length penalty (can help control length)
        #early_stopping=True
    )
    abstractive_summary = tokenizer.decode(outputs[0], skip_special_tokens=True)

    return abstractive_summary
```

```
<IPython.core.display.HTML object>
```

[19]:
```python
abstractive_summary = summarize_full_summary(extractive_summary)

# Print the abstractive summary
print("Abstractive Summary:")
print(abstractive_summary)
```

```
<IPython.core.display.HTML object>
```

```
Abstractive Summary:
Long documents will first be extracted for salient sentences and reorganized
into compressed documents. These documents are then used to generate the final
summary by abstractive summarization. This approach does not change the
structure of the original pre-trained language model (PLM) Instead, it solves
the long document problem by changing the input of the PLMs. For the reduced
```

complexity of these mechanisms, models using key sentence selection and BART may come at the cost of quality to the original document. Based on its successful precedent for long document classification and Q&A tasks, we propose a hybrid long document summarization method that en ables compression and summarization of original long documents. Weichao CHEN, Mizuho IWAIHARA, and Vicochen are the authors of the paper and will be presenting it at the DEIM Forum 202 3 1b-5-2 in Tokyo, Japan, on November 14. For more information, visit the Waseda University website or go to www.waseda.ac.uk. For confidential support, call the Samaritans on 08457 90 90 90 or visit a local Samaritans branch or click here for details. For support in the U.S., call the National Suicide Prevention Lifeline at 1-800-273-8255 or visit http://www.suicidepreventionlifeline.org/. For confidential help in the UK, contact Samaritans at 08457 909090 or www.samaritans.org. Back to the page you came from. Click here for more information. For a copy of this paper, please go to: http: //www.vico_chen.com/vico-chen-and-mizuho-iwaihara-vico.html#storylink=cpy. The article has been updated to reflect a change in the name of the conference, and to make clear that the author is not a member of the Society for Information Technology (SIT), but a professor of Information, Production and Systems (GIS) at WasedA University. The author would also like to thank the SIT for their help in developing the paper, which was originally published on November 11, 2013. For further information, please visit: http:// www.sit.org/news/2013/11/14/stories/features/features-long-document-summarization-using-hybrid-extractive- and- abstractive-researcher-method-weichao-chen.html. The paper is also available as a free download from SIT's website. For. a limited time only, you can download the full version of this article by clicking the link below. The SIT website is available for a limited period of time only. You will be able to download the entire article for a fee of $1,000 (US$2,000). For a full copy of the article, click the link again. For an example of how to use the Hybrid Extractive -Abstractive Method, see the SITT website. The link is available on the SIXTH page of the DeIM Forum, which is available from November 14, 2013 for $2,500 (U.S. $3,000, $4,500, $5,000 and $6,500). The SIXth page is the first part of a two-part series of articles on the hybrid long-document summarization Method. The second part of the series includes a discussion on how to apply the method to more complex documents, such as press releases, scientific literature, specialized reports, and Big Bird. The final section of the report is the introduction to the Hybrid Long Document Summarization Method (HLS) which uses a combination of the HLS and BART models for the summarization task. The HLS method is available online for free on SIT and SOTA, and can be downloaded from the SUT website for $1.99 (US $3.99) or $4.99 ($5.99, $6.99), depending on the version of the document you want to use. The method is based on the CLTX model, which has been proposed by DingLTX, CogX, and COGX, which achieves results by using key sentences and downstream classification tasks by using Cog X, COG X, and the BART model to filter source source sentences for summarization tasks. It is also possible to use BERT as a sentence class class for the summary task. For example, BERT can filter source sentences using BERT and BART for the document summarizer for the Longformer task. This method can be used to filter BERT for the BERT task. It can be applied to more than 4096 token s in length. It has the advantage of smoothness and fluency, while keeping the content information, so our discussion will mainly focus on abstract ive summarizations.

1.1 Abstractive

[4]:
```
import PyPDF2

# Function to extract text from the PDF
def extract_text_from_pdf(pdf_path):

    text = []
    try:
        with open(pdf_path, 'rb') as pdf_file:
            reader = PyPDF2.PdfReader(pdf_file)
            num_pages = len(reader.pages)
            # Extract text from each page and append to the list
            text = [reader.pages[page_num].extract_text() for page_num in␣
 ↪range(num_pages)]

    except Exception as e:
        print(f"Error occurred while extracting text from PDF: {e}")
        return None

    return ''.join(text)
```

```
<IPython.core.display.HTML object>
```

[5]:
```
# Function to preprocess and chunk the text

def preprocess_and_chunk_text(text, max_length=1024):
    try:
        # Tokenize the text and split it into chunks based on the model's max␣
 ↪length
        inputs = tokenizer(text, return_tensors="pt", truncation=True,␣
 ↪padding=True, max_length=max_length)

        input_ids = inputs["input_ids"]
        if len(input_ids) == 0:
            raise ValueError("Tokenization resulted in empty input IDs. Please␣
 ↪check the text or the tokenizer.")

        return input_ids

    except IndexError as e:
        print(f"Tokenization error: {e}. Check the input text and tokenizer.")
        return None
```

```
<IPython.core.display.HTML object>
```

[6]:
```python
# Function to summarize text using the model
def summarize_text(input_ids):
    outputs = model.generate(
        input_ids,
        max_length=5000,      # Increase max length for longer summaries
        min_length=1000,      # Set minimum length for summaries
        length_penalty=2.0,   # Adjust length penalty
        num_beams=4,          # Use beam search for better quality summaries
    )

    summary = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return summary
```

```
<IPython.core.display.HTML object>
```

[7]:
```python
# Main function to summarize the PDF
def summarize_pdf(file_path):

    text = extract_text_from_pdf(file_path)
    input_ids = preprocess_and_chunk_text(text)
    summary = summarize_text(input_ids)

    return summary
```

```
<IPython.core.display.HTML object>
```

[8]:
```python
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer

model_name = 'facebook/bart-large-cnn'

# Initialize the tokenizer and model
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSeq2SeqLM.from_pretrained(model_name)
```

```
<IPython.core.display.HTML object>
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88:
UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab
and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
public models or datasets.
  warnings.warn(
[9]: # view the pdf at "https://drive.google.com/file/d/
      ↪1rwHAU7eKeJTI_ZStfIU6oGZi0pMe5Nv9/view?usp=sharing"

     pdf_file_path = "/content/Efficient Summarization of Long Documents.pdf"

     summary = summarize_pdf(pdf_file_path)
     print("Summary:")
     print(summary)
```

```
<IPython.core.display.HTML object>

Summary:
Efficient Summarization of Long Documents Using Hybrid  Extractive -Abstractive
Method. Long documents will first be extracted for salient
 sentences and reorganized into compressed documents. Finally, the final summary
will be generated by abstractive summarization. The proposed method can achieve
good summarization perfo rmance for general long documents in single -GPU or low
-resource scenarios. It can be broadly divided into two categories: Efficient
self-attention mechanisms and hybrid  summarization methods. The most efficient
approach is to replace the full attention with the attention of local windows,
which allows the computational focus to focus on specific parts of the document.
The solution is to improve the self -attention mechanism to make it more
efficient in the most ostentatious way. It is also possible to use the most
efficient hybrid long  document summarization method that en ables compression
and summarization of original long                 documents by combining
content selection and abstract ive summarizing. The method is available from the
Waseda University Graduate School of Information, Production and Systems in
Japan. For more information, visit www.waseda.ac.uk/research/summarizing-long-
documents-using-hybrid-extractive-and- abstractive-methods-with-pre-trained-
language-models. For confidential support, call the Samaritans on 08457 90 90
90, visit a local Samaritans branch or click here for details. For support in
the U.S., call the National Suicide Prevention Lifeline at 1-800-273-8255 or
visit http://www.suicidepreventionlifeline.org/. For support on suicide matters
in the UK, call the Samaritans' national helpline on 0800-847-9090 or visit
http:// www.samaritans.org/uk/suicide-prevention-lifeline or the University of
Liverpool's Samaritans. For help with suicide matters, contact the Samaritan
Samaritans at 08457 909090 or "The Samaritans," a local branch of the University
of Liverpool, on the same day and place as the forum, call 08457 809090. For
information on suicide prevention in the United States, visit  the U.S.
Samaritans on the same date and place, and see "The Samaritan" for details on
how to get to the right place in the world. For details on suicide Prevention in
the Middle East, please visit the www.U.A.E.C.E-mail:
†vico_chen@akane.wasinga.jp,  ‡iwaihara@wased a.j p                 ",
"iwai'hara",. "". "'iwaiHara'.'' '"'  '"    Mizuho  IWAIHARA‡   "Graduate
School of information, Production. and Systems, WasedA University." " ".". " '.
' "  ' 'The method is based on pre -trained language models (PLMs' [1]. '. 'It
is possible to combine content selection with abstractive summary to generate a
new summary that captures the main ideas and concepts in the original tex t,
```

where ipientpassages not appearing in theOriginal document may be generated',' says Weichao. "Weichao CHEN' The method can also be used to extract and compress short documents. "The method can be used for short documents, such as press releases, scientific literature, specialized reports, and other specialized reports" [2]. "It can be applied to short documents as well as longer documents. It has the advantage of smoothness and fluency, while keeping the content of the original document in mind'. The method has the disadvantage of being more efficient than the most popular approach to long document adaptation methods. , The main goal is to make long documents more efficient by replacing the full Attention -based analysis with the more specific attention-based analysis of local window windows. This approach is usually effective, but obviously will lose semantic information of the truncated part, so the results are not comprehensive [9]. It can also improve the efficiency of long documents with a new approach to document adaptation. • Efficient hybrid long documents using hybrid extractive - Abstractive Method'