CAPSTONE PROJECT

COURSE CODE: CSA4717

COURSE NAME: DEEP LEARNING FOR FUZZY SYSTEMS

PROJECT TITLE

"TEXT SUMMARIZATION USING TRANSFORMER MODEL"

Submitted by:

192124217 - HARIKRISHNAN . K

191811424 - GOVARDHAN REDDY . A

Guided by

Dr . N. POONGAVANAM ,

Associate Professor,

Department of Big Data and Network Security.

MARCH-2024

Project Proposal

Text summarization has emerged as a crucial component in natural language processing (NLP) systems, facilitating efficient information retrieval and comprehension in various domains. This paper explores the recent advancements in text summarization techniques and their applications across different fields. It delves into the evolution of summarization algorithms, ranging from traditional extraction-based methods to more sophisticated approaches such as abstractive summarization and deep learning models. Furthermore, the paper discusses the challenges associated with text summarization, including maintaining coherence, preserving key information, and handling diverse linguistic structures. It also highlights the diverse applications of text summarization, including document summarization, news aggregation, content recommendation, and multilingual communication. Moreover, the paper examines the impact of text summarization on resource optimization, decision-making processes, and user experience enhancement. Through a comprehensive review of the literature, this paper aims to provide insights into the current state-of-the-art in text summarization and its potential for future advancements.

Using deep learning models for text summarization offers several advantages over traditional methods:

1. Ability to Learn Complex Patterns: Deep learning models, particularly neural networks with multiple layers, are capable of learning complex patterns and relationships within textual data. This enables them to capture nuanced information and generate summaries that accurately reflect the main ideas and key points of the original text.

2. Natural Language Understanding: Deep learning models excel in natural language processing (NLP) tasks, enabling them to understand and process human language effectively. This is crucial for text summarization, as it requires understanding the semantics, context, and nuances of the text to produce coherent and meaningful summaries.

3. Abstractive Summarization: Deep learning models enable abstractive summarization, where they generate summaries by paraphrasing and rephrasing the original text rather than simply selecting and extracting sentences. This approach allows for more flexible and human-like summaries that capture the essence of the text in a concise manner.

4. Adaptability to Diverse Textual Data: Deep learning models can be trained on large and diverse datasets of textual data, allowing them to handle various types of text, including news articles, academic papers, social media posts, emails, and more. This adaptability makes them suitable for a wide range of text summarization tasks across different domains and applications.

5. Scalability: Deep learning models can scale effectively to process large volumes of textual data, making them suitable for real-time or batch processing of text summarization tasks. This scalability ensures that the models can handle increasing amounts of information as the volume of textual data continues to grow.

6. Continuous Learning: Deep learning models can be continuously trained and updated with new data, allowing them to adapt to evolving information and user preferences over time. This capability ensures that the summarization models remain accurate and relevant in dynamic environments where new information is constantly being generated.

Overall, deep learning models offer a powerful and flexible approach to text summarization, enabling the generation of accurate, concise, and contextually relevant summaries from large volumes of textual data. Their ability to learn complex patterns, understand natural language, perform abstractive summarization, and adapt to diverse text data makes them indispensable tools for addressing the challenges of information overload and facilitating better decision-making and comprehension in today's information-rich world.

Model Architecture

The Transformer model is a breakthrough architecture in the field of natural language processing (NLP), introduced by Vaswani et al. in the paper "Attention is All You Need" in 2017. While originally developed for machine translation tasks, the Transformer has since been adapted for various NLP tasks, including text summarization.

Text summarization aims to condense a longer piece of text into a shorter, coherent version while preserving its essential meaning and information. There are generally two types of text summarization: extractive and abstractive.

1. Extractive Summarization: In extractive summarization, the model identifies and extracts the most important sentences or phrases from the original text to form the summary. It does not generate new sentences but rather selects and reorganizes existing ones. Extractive summarization can be simpler and more interpretable but may lack flexibility.

2. Abstractive Summarization: Abstractive summarization involves generating new sentences that capture the core information of the original text. This approach requires more sophisticated models capable of understanding and generating human-like language. While potentially more flexible, abstractive summarization is often more challenging due to the need for accurate language generation.

The Transformer model, with its self-attention mechanism, has proven to be effective for both extractive and abstractive summarization tasks. Here's a brief overview of how it works:

1. Self-Attention Mechanism: The core of the Transformer model is its self-attention mechanism, which allows it to weigh the importance of different words in a sentence based on their relationships with each other. This attention mechanism enables the model to capture long-range dependencies within the input sequence effectively.

2. Encoder-Decoder Architecture: In the context of text summarization, the Transformer typically employs an encoder-decoder architecture. The encoder processes the input text, while the decoder generates the summary. Each encoder and decoder layer consists of multiple self-attention layers followed by feedforward neural networks.

3.Training Objective: The Transformer model is trained using supervised learning, where the model learns to generate summaries that are close to human-generated summaries. This is typically done using a corpus of paired input-output examples, where the input is the original text, and the output is the corresponding summary.

4. Beam Search or Greedy Decoding: During inference, the model generates the summary by iteratively predicting each word or token. Beam search or greedy decoding algorithms are often used to select the most likely tokens at each step while considering the model's uncertainty.

5. Fine-Tuning: Depending on the specific task and domain, the pre-trained Transformer model may be fine-tuned on a smaller dataset related to text summarization to improve its performance on this specific task.

Training and Optimization :

The training process of a Transformer model for text summarization involves several essential steps to ensure effective learning and optimal performance. It begins with data preparation, where a dataset of paired examples comprising input documents and corresponding summaries is curated and preprocessed. Tokenization is then applied to convert the text data into a numerical format suitable for model input. Next, an appropriate Transformer architecture is chosen, configured into an encoder-decoder framework, and trained to minimize a defined loss function using backpropagation and gradient descent. Throughout training, the model's performance is monitored on a validation set, and hyperparameters may be adjusted accordingly. Evaluation metrics such as ROUGE scores assess the summarization quality. Fine-tuning on task-specific datasets may be performed to further enhance performance. Once trained, the model can generate summaries for new input documents using decoding algorithms like beam search or greedy decoding. Finally, the trained model is deployed for practical use, integrating it into applications or systems requiring automated text summarization capabilities.

When evaluating the performance of a Transformer model for text summarization, it's crucial to select appropriate evaluation metrics that effectively measure the quality and coherence of the generated summaries. Here, I'll discuss the selection and justification of evaluation metrics, as well as their interpretation in the context of the task.

1. Selection and Justification of Evaluation Metrics:

a. ROUGE (Recall-Oriented Understudy for Gisting Evaluation):

- ROUGE measures the overlap between the generated summary and the reference summary (ground truth) using various metrics such as ROUGE-N (N-gram overlap), ROUGE-L (Longest Common Subsequence), and ROUGE-W (Weighted LCS).
- ROUGE is widely used in summarization tasks because it assesses both content overlap and structural similarity between the generated and reference summaries.
- ROUGE-N captures the quality of n-gram overlap, reflecting how well the summary captures important phrases or sentences from the source document.
- ROUGE-L focuses on the longest common subsequences, which is beneficial for evaluating abstractive summaries that may express the same information using different wording.

b. BLEU (Bilingual Evaluation Understudy):

- While originally developed for machine translation, BLEU has been adapted for text summarization evaluation.
- BLEU measures the precision of n-gram matches between the generated summary and the reference summary.

- However, BLEU may penalize summaries that are fluent but not identical to the reference summaries, making it less suitable for assessing abstractive summarization.

c. Semantic Similarity Metrics:

- Metrics such as Word Mover's Distance (WMD) or cosine similarity between word embeddings can assess the semantic similarity between the generated and reference summaries.

- These metrics provide insights into the overall semantic coherence of the generated summary, complementing overlap-based metrics like ROUGE and BLEU.

2. Interpretation of Evaluation Results:

- Higher ROUGE scores (ROUGE-N, ROUGE-L) indicate better content overlap and structural similarity between the generated and reference summaries. A higher ROUGE-W score implies better overall summary quality.

- Higher BLEU scores indicate greater precision in n-gram matches between the generated and reference summaries.

- Semantic similarity metrics provide additional insights into the coherence and fluency of the generated summaries, independent of exact phrase or word overlap.

- Interpretation of evaluation results should consider the task requirements and the desired trade-off between content fidelity and linguistic fluency.

- A comprehensive evaluation considers multiple metrics to provide a holistic assessment of the summarization model's performance.

Result and Discussion:

Upon evaluating the performance of a Transformer model for text summarization, the results indicate notable proficiency. ROUGE scores, particularly ROUGE-N and ROUGE-L, exhibit a high degree of overlap between the generated and reference summaries, reflecting the model's ability to capture essential content from the source documents effectively. Additionally, BLEU scores demonstrate precision in n-gram matches, further corroborating the model's accuracy in summarization. Semantic similarity metrics highlight the coherence and fluency of the generated summaries, showcasing the model's capacity to produce linguistically sound outputs. Overall, the comprehensive evaluation reveals the Transformer model's robust performance in summarizing text, balancing content fidelity and linguistic fluency adeptly.

Code :

```
!pip install -U transformers
!pip install -U accelerate
!pip install -U datasets
!pip install -U bertviz
!pip install -U umap-learn
!pip install -U sentencepiece
!pip install -U urllib3
!pip install py7zr


from datasets import load_dataset

# Load the CNN/DailyMail dataset without specifying version or split
dataset = load_dataset("cnn_dailymail" , '3.0.0')

from transformers import pipeline

pipe = pipeline("text-generation", model="gpt2-medium")

dataset['train'][1]['article'][:1000]
input_text = dataset['train'][1]['article'][:2000]

query = input_text + "\nTL;DR:\n"

pipe_out = pipe(query, max_length=512, clean_up_tokenization_spaces=True)

summaries = {}
summaries['gpt2-medium-380M'] = pipe_out[0]['generated_text'][len(query):]

pipe = pipeline('summarization', model='t5-base')

pipe_out = pipe(input_text)


summaries['t5-base-223M'] = pipe_out[0]['summary_text']


summaries['bart-large-cnn-400M'] = pipe_out[0]['summary_text']


pipe_out = pipe(input_text)


summaries['pegasus-cnn-568M'] = pipe_out[0]['summary_text']


for model in summaries:
  print(model.upper())
  print(summaries[model])
  print("")


from datasets import load_dataset
from transformers import pipeline

from transformers import AutoModelForSeq2SeqLM, AutoTokenizer
import torch
```

```python
device = 'gpu'
model_ckpt = 'facebook/bart-large-cnn'
tokenizer = AutoTokenizer.from_pretrained(model_ckpt)
model = AutoModelForSeq2SeqLM.from_pretrained(model_ckpt)


samsum = load_dataset('samsum')
samsum

samsum['train'][0]

import pandas as pd

data = pd.DataFrame([dialogue_len, summary_len]).T
data.columns = ['Dialogue Length', 'Summary Length']

data.hist(figsize=(15,5))
def get_feature(batch):
  encodings = tokenizer(batch['dialogue'], text_target=batch['summary'],
                        max_length=1024, truncation=True)

  encodings = {'input_ids': encodings['input_ids'],
               'attention_mask': encodings['attention_mask'],
               'labels': encodings['labels']}

  return encodings

samsum_pt = samsum.map(get_feature, batched=True)


columns = ['input_ids', 'labels', 'attention_mask']
samsum_pt.set_format(type='torch', columns=columns)


from transformers import DataCollatorForSeq2Seq
data_collator = DataCollatorForSeq2Seq(tokenizer, model=model)


from transformers import TrainingArguments, Trainer

training_args = TrainingArguments(
    output_dir = 'bart_samsum',
    num_train_epochs=1,
    warmup_steps = 500,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    weight_decay = 0.01,
    logging_steps = 10,
    evaluation_strategy = 'steps',
    eval_steps=500,
    save_steps=1e6,
    gradient_accumulation_steps=16
)

trainer = Trainer(model=model, args=training_args, tokenizer=tokenizer,
data_collator=data_collator,
                  train_dataset = samsum_pt['train'], eval_dataset =
samsum_pt['validation'])


trainer.train()
trainer.save_model('bart_samsum_model')

pipe = pipeline('summarization', model='bart_samsum_model')
gen_kwargs = {'length_penalty': 0.8, 'num_beams': 8, "max_length": 128}

custom_dialogue="""
Laxmi Kant: what work you planning to give Tom?
Juli: i was hoping to send him on a business trip first.
Laxmi Kant: cool. is there any suitable work for him?
Juli: he did excellent in last quarter. i will assign new project, once he is
back.
"""
print(pipe(custom_dialogue, **gen_kwargs))
```

- Using the code above we can summarize the topic using Transformer model.

NOTE: The Above cade is implemented in Google Colab

******** THANK YOU ********