# DAT410 - Module 8: Final Project - Group 49

Harry Denell
MPDSC
denell@chalmers.se
Personal number: 20000704-1379

Elvina Fahlgren
MPDSC
elvina@chalmers.se
Personal number: 20000207-8400

January 2025

*"We hereby declare that we have both actively participated in solving every exercise. All solutions are entirely our own work, without having taken part of other solutions."*

**Abstract**

This project investigates automated headline generation using a pretrained BART model fine-tuned on news article descriptions and corresponding headlines. By integrating Low-Rank Adaptation (LoRA), we significantly reduced computational overhead and training time. The model was trained on the Kaggle News Category Dataset via the Hugging Face platform. Evaluations showed steady improvements in BLEU scores, reaching approximately 50%, though analysis revealed occasional overfitting and sensitivity to minor discrepancies. Overall, our findings highlight the promise of LoRA for efficient headline generation and the potential for fine-tuning pretrained models, in order to solve domain-specific tasks, without the need to train new models from scratch.

# 1 Introduction and Motivation

This project explores automated headline generation by fine-tuning a pretrained Transformer model, specifically BART (1), on pairs of news article descriptions and corresponding headlines.

Given the critical role that headlines play in attracting readings, aswell as shaping readers interpretation of news, enhancing automatic generation is valuable for efficient and engaging news delivery. To address computational challenges commonly associated with fine-tuning large(r) models, we integrate Low-Rank Adaptation (LoRA), aiming to maintain quality outputs, while significantly improving computational efficiency. The effectiveness of this approach is evaluated to understand both practical performance on a domain-specific task (headline generation), as well as implications for resource-efficient model post-training.

We propose the following research questions to be addressed in this project:

- How effectively can a pretrained transformer model (BART) generate headlines from article descriptions?
- How does integrating LoRA for parameter-efficient fine-tuning impact model performance and computational efficiency?

## 1.1 News Headlines

News headlines form an first interaction between readers and news stories, serving not only as summaries but also as interpretive frames (2). Effective headlines must balance information density against the reader's cognitive effort, therefore optimizing relevance, by quickly guiding readers toward key contextual understanding (3).

Writing headlines requires editors to make strategic choices under tight deadlines, deciding whether to just summarize content, emphasize specific details, or even incorporate direct quotes, to create engaging yet informative entry points for readers (3).

Headlines in digital journalism are becoming shorter and sharper, designed for quick reading (2). This change is driven by technology and the need for higher engagement, such as more clicks. As a result, headlines often blend facts with 'catchy phrases' to grab attention, making it harder to separate what's true from what's exaggerated (2). Also, headlines are designed to evoke emotions and enhance memorability, which helps grab the readers' attention to news content (2). As such, headlines don't just sum up the facts but also convey varying degrees of certainty, tone, and expected outcomes, sometimes giving some misinformation because they're so brief and selective (4).

## 1.2 Previous Work

Previous work explored the use of encoder-decoder recurrent neural networks (RNNs) with attention mechanisms for automatic headline generation (5). The authors of this bachelors thesis trained a conditioned model on approximately 500,000 news articles paired with their headlines. Input data was preprocessed through tokenization using a white-space tokenizer, limiting vocabulary to the 40,000 most frequent words and truncating articles and headlines to maximum lengths of 200 and 50 tokens, respectively. The model architecture comprised 2 to 4 layers of Gated Recurrent Units (GRU), each containing between 512 and 1024 hidden units, implemented using TensorFlow's seq2seq library. The evaluation showed that although the model successfully identified important keywords within articles, it frequently struggled to generate grammatically coherent or meaningful sentences (5).

Other recent research underscores the limitations of traditional recurrent neural network (RNN)-based methods for automatic headline generation, particularly issues with repetitive wording and insufficient contextual accuracy (6). Transformer-based models is claimed as a promising alternative (6).

These findings seem to motivate our choice of a transformer-based model (BART) combined with parameter-efficient fine-tuning methods (LoRA) to generate accurate, informative, and readable headlines effectively. The transformer-based model should be able to handle grammatical understanding and generation, while the post-tranining with LoRA will help the model learn patterns of what constitute a 'good' headline.

# 2 Background

Semi-supervised learning can be used for languague learning tasks. Semi-supervised sequence learning uses unsupervised pre-training on large unlabeled datasets to produce robust initial model representations (7). Such pre-training improves model stability, convergence, generalization, and the handling of long-range dependencies, enhancing downstream task performance (7). The model representations can then be extended to more domain-specific tasks, as post-training, with labeled pairs of training data and training target.

## 2.1 Transformer Models

Transformer models are a type of neural network that use self-attention mechanisms instead of traditional methods like recurrence or convolutions (8). They work by comparing every word in a sequence with every other word, allowing the model to decide which parts of the input are most important. This approach not only speeds up the training process but also helps the model learn long-distance relationships between words more effectively (8).

The transformer architecture is typically divided into two parts: an encoder and a decoder (8). The encoder reads the input sequence and transforms it into a series of vectors that capture the meaning of the text. The decoder then uses these vectors to generate the output sequence, such as a translated sentence. Key components like multi-head attention, scaled dot-product attention, residual connections, and positional encodings all work together to make transformers powerful and efficient for tasks like translation, summarization, and language understanding (8).

## 2.2 BERT Model

Bidirectional Encoder Representations from Transformers (BERT) is an approach to language modeling that leverages contextual information from both preceding and following words simultaneously. This bidirectional context enables richer semantic understanding compared to unidirectional methods, enhancing performance across various natural language tasks (9).

A key strength of BERT is its simple yet effective fine-tuning strategy. After pre-training on extensive unlabeled text, BERT can be adapted for specific tasks by adding minimal task-specific components, typically just an additional output layer. This simplicity allows BERT-based models to achieve state-of-the-art performance across numerous downstream natural language processing tasks with minimal modifications (9).

Additionally, empirical evidence from BERT demonstrates clear advantages in scaling up model size, where larger models consistently outperform smaller counterparts. However, increasing the number of layers, hidden units, and attention heads substantially increases computational resource requirements, highlighting the need for efficient fine-tuning methods when adapting such models for practical applications (9).

## 2.3 BART Model

Building upon the transformer architectures of BERT, BART employs a denoising autoencoder objective during pre-training (10). Unlike traditional language models, BART learns by reconstructing original input text from artificially corrupted versions, using noising methods like token masking, text infilling, and sentence permutation (10). This denoising pre-

training method encourages the model to capture robust representations of the underlying meaning in noisy or incomplete textual inputs.

BART leverages a standard sequence-to-sequence transformer structure, integrating BERT's strength of bidirectional contextual encoding with GPT's autoregressive decoding capabilities (10). Its noising methods enable BART to develop a deeper structural and semantic understanding of language, which significantly enhances its generation capabilities across tasks, e.g. summarization (10).

Due to BERT's pre-training, and architecture, it serves as a suitable model for our project's goal of efficient headline generation. Fine-tuning BART on a task-specific dataset of news article descriptions and corresponding headlines allows the model to specialize its generative capabilities toward creating accurate and contextually relevant headlines.

## 2.4 LoRA

When dealing with very large models, fine-tuning all the parameters for a new task can be computationally expensive and require huge amounts of memory. Low-Rank Adaptation (LoRA) (11) addresses this challenge by modifying only a small portion of the model while leaving the original, pre-trained weights fixed. This makes the fine-tuning process much more efficient.

LoRA works by instead of updating the full weight matrix $W_0$ during fine-tuning, the weight update $\Delta W$ is decomposed into two low-rank matrices $A$ and $B$:

$$\Delta W = BA$$

Where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, and $r \ll \min(d, k)$ is the low rank. This gives an adapted weight matrix:

$$W = W_0 + \Delta W = W_0 + BA.$$

Here, $W_0$ is kept frozen (unchanged), and only the low-rank matrices $A$ and $B$ are updated during fine-tuning, which significantly reduces the number of trainable parameters.

Empirical results show that LoRA can reduce the number of trainable parameters by up to 10,000 times compared to full fine-tuning on very large models (11). This results in faster training and lower VRAM requirements (11).

LoRA matches full fine-tuning performance on models like RoBERTa, GPT-2, and GPT-3 even at very low ranks (e.g., $r = 1$ or $r = 2$) (11) .

## 2.5 BLEU

BLEU (12) is a widely used metric for evaluating machine translation and other text generation tasks. It compares the n-gram overlap between a candidate output and one or more reference texts, using a clipping mechanism to avoid over-counting. The BLEU score is computed as the geometric mean of the modified n-gram precisions, adjusted by a brevity penalty to discourage overly short outputs (12).

Mathematically, the BLEU score is defined as:

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

where $p_n$ is the modified precision for n-grams of order $n$, $w_n$ are the weights, and $BP$ is the brevity penalty, given by:

$$BP = \begin{cases} 1, & \text{if } c > r, \\ e^{1-\frac{r}{c}}, & \text{if } c \leq r, \end{cases}$$

where $c$ is the length of the candidate text and $r$ is the effective reference length (usually determined by the closest reference length) (12). BLEU is valued for its simplicity and its strong correlation with human judgments when evaluated over large test sets.

# 3 Methods

The objective of this work is to fine-tune a pretrained BART model for generating news headlines from article descriptions. In the following sections, we describe our approach to data collection and preprocessing, model configuration, training, and evaluation. The code is found in A.

## 3.1 Data Retrival and description

The experiments use the 'News Category Dataset' (13), from Kaggle. It contains around 210,000 news headlines, sourced from HuffPost, spanning the years 2012 to 2022. Each entry in the dataset includes the following fields:

- **Category**: The primary classification of the news article (e.g., Politics, Entertainment, Sports).
- **Headline**: The title of the news article.
- **Authors**: The names of the authors who wrote the article.
- **Link**: The URL to the full news article.
- **Short Description**: A brief summary of the article's content.
- **Date**: The publication date of the article.

## 3.2 Data Collection and data exploration

The dataset was downloaded from Kaggle and was provided in JSON format. Our first step was to convert the JSON data into a data frame for analysis. This enabled us to examine some basic statistics, such as the distribution of article description and headline lengths, shown in Figure 1. Further, Figure 2 shows the diversity and frequency among the news categories.

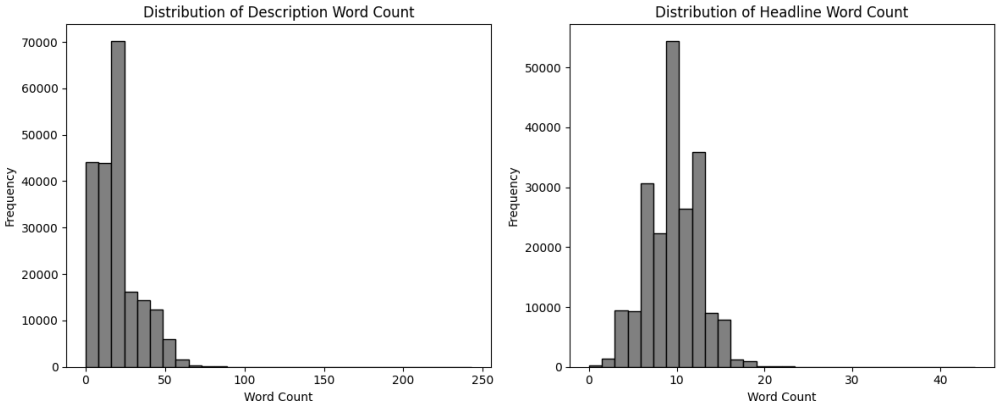


Figure 1: Distribution of number of words in the news description and headline

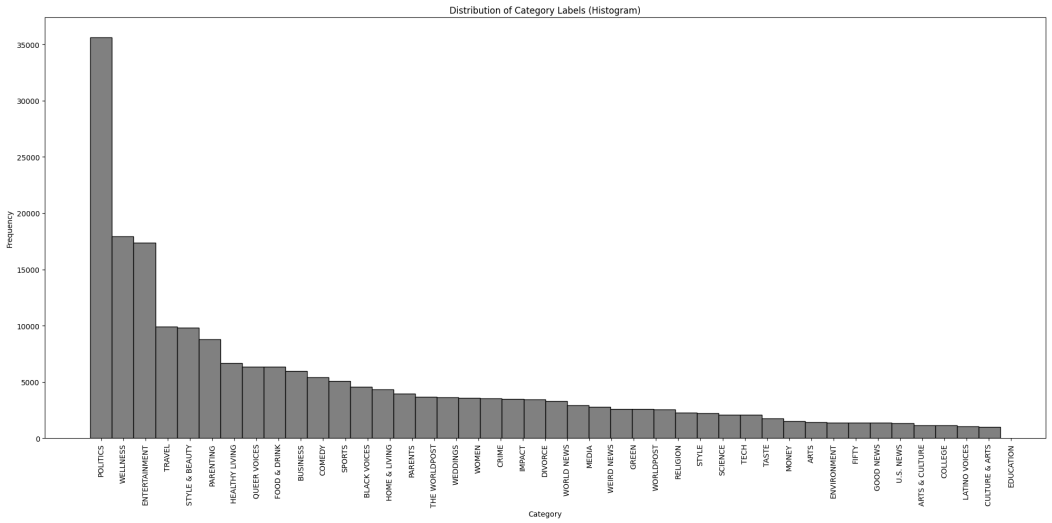


Figure 2: Distribution of news categories

## 3.3 Data preprocessing

During preprocessing, we focused on extracting two key fields: the article description (input text) and the corresponding headline (target text). We ensured that all samples contained non-empty values.

Tokenization was performed using the BART tokenizer. The maximum input sequence length was set to 250 tokens and the maximum target sequence length to 30 tokens, values chosen after the previous data analyzing step.

## 3.4 Model Setup

We used the `facebook/bart-base` model available from Hugging Face (1) as backbone for our sequence-to-sequence headline generation task. To improve training efficiency and reduce computational overhead, we integrated the Low-Rank Adaptation (LoRA) technique using the PEFT framework.

In our implementation, LoRA was configured with:

- A rank $r$=*8*
- A scaling factor (`lora_alpha`) of 32.
- A dropout rate of 0.1.
- Targeting the attention projection layers (specifically, `q_proj` and `v_proj`).

## 3.5 Training Procedure

The fine-tuning process was done using the Hugging Face `Seq2SeqTrainer`. The training configuration was designed to fully leverage available hardware, specifically an NVIDIA A100 GPU with 40 GB VRAM (an option available after purchasing Google Colab Pro).

Some key hyperparameters were set as follows:

- Batch size 16
- The learning rate was set to $2 \times 10^{-5}$, which seems standard for fine-tuning transformer models.

- 5 number of epochs.
- FP16 was enabled to speed up training and reduce memory consumption.
- Checkpoints were saved at the end of each epoch to allow for training resumption if needed.

Here we tried to push both batch size and number of epochs, but this was the highest we were able to achieve with our setup.

A training script processes the entire training dataset, and evaluation is performed at the end of each epoch to monitor model convergence.

All code is available in Appendix A

### 3.6 Evaluation Metrics

For a quantitative evaluation, we computed the BLEU score using the Hugging Face `evaluate` library. In our implementation, generated headlines are decoded from token IDs into strings, and reference headlines are similarly decoded. BLEU is computed on a per-example basis, with the final score scaled to a percentage.

While a higher BLEU score indicates a closer n-gram match between the generated and reference headlines, it is important to note that BLEU has limitations in capturing the full quality of creative headline generation.

We supplement the quantitative evaluation with qualitative analysis by manually inspecting sample predictions and comparing them to the references in Results.

## 4 Results

Table 1 shows the training and validation losses across five epochs, along with the corresponding BLEU scores.

| **Epoch** | **Training Loss** | **Validation Loss** | **BLEU** |
|---|---|---|---|
| 1 | 5.1588 | 4.7555 | 0.0286 |
| 2 | 5.1230 | 4.7085 | 0.3869 |
| 3 | 5.0799 | 4.6893 | 0.4447 |
| 4 | 5.0875 | 4.6837 | 0.5015 |
| 5 | 5.0762 | 4.6797 | 0.5041 |

Table 1: Training and validation results per epoch, including BLEU scores.

By the final epoch, the model achieved an evaluation loss of 4.6797 and a BLEU score of approximately 50.4%. These quantitative results provided a rather clear indication of the model's quantitative performance on the headline generation task. We see a clear increase in the BLEU score per epoch, but loss values are rather stable, not really decreasing much. Figure **??** show the frequency of the BLEU scores.

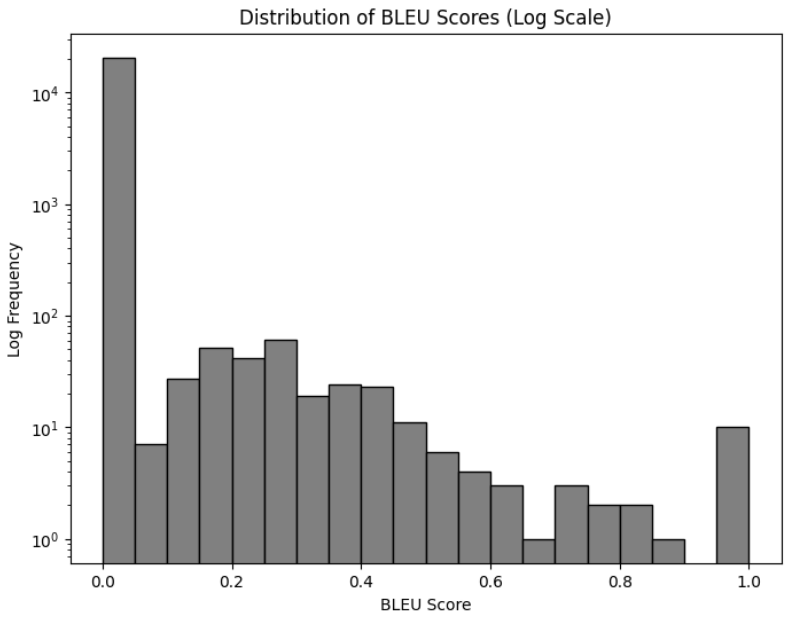


Figure 3: Distributions of BLEU scores.

The model was further evaluated by examining sample generated headlines for fluency, readability, and relevance. In many cases, the generated headlines closely capture the main idea of the article descriptions, though certain nuances may be lost or simplified. Below are some predicted examples:

```
First 4 samples:

Sample 1:
  Prediction: This Is What You Should Do If You Want To Be A Girl (PHOTOS)
  Actual: HuffPost Her Stories: Let's Talk About Therapy

Sample 2:
  Prediction: Hottest Male Models From New York Fashion Week (PHOTOS)
  Actual: New York Fashion Week Fall 2012: Glamour Asks, 'What Are You Wearing?' (VIDEO)

Sample 3:
  Prediction: How To Save Your Life In The New Year
  Actual: Is 'X-Men: Days Of Future Past' Worth Your Money?

Sample 4:
  Prediction: Trump Is Not The President Of The United States
  Actual: Donald Trump Should Remember Commitment To America And Stay Out Of Syria

Top 4 examples by BLEU score:

Sample 1865:
  Prediction: Stress Relief Tips From Around The World
  Actual:  Stress Relief Tips From Around The World

Sample 6112:
  Prediction: Two Ingredient Caramel Ice Cream
  Actual:  Two Ingredient Caramel Ice Cream

Sample 13797:
```

```
  Prediction: How To Make Mozzarella
  Actual:  How To Make Mozzarella

Sample 14469:
  Prediction: 12 Life Lessons You Can Learn From Crossing the Street in Vietnam
  Actual:  12 Life Lessons You Can Learn From Crossing the Street in Vietnam

4 samples with BLEU scores between 45.0% and 55.0%:

Sample 6041:
  Prediction: Zombie Joe's Underground Theatre Group
  Actual:  Othello at Zombie Joe's Underground Theatre

Sample 15546:
  Prediction: The Silver Linings of Divorce
  Actual:  Sharing the Silver Linings of Divorce

Sample 8337:
  Prediction: Best Things to Do in Belize
  Actual:  10 Things to Do in Belize in 2014

Sample 20593:
  Prediction: Donald Trump And Hillary Clinton In New Hampshire
  Actual:  Bernie Sanders Surpasses Hillary Clinton In New Hampshire Poll
```

# 5 Discussion

Using LoRA improved our computational efficiency. By updating fewer parameters than the base models total weights, we trained the model faster and used less RAM. The training and validation losses stayed mostly stable, which suggests that while our approach is efficient, there may still be room for further performance improvements, or we could possibly have used a different loss function. In general, our result aligns with previous studies showing that LoRA can likely closely match full fine-tuning performance while using far fewer trainable parameters.

The qualitative evaluation shows that many generated headlines are both readable and relevant, capturing the main ideas of the article descriptions. In several cases, the outputs are nearly identical to the actual headlines, which is promising. However, there are instances where the headlines, despite their fluency, miss some contextual details or simplify the intended message. In one interesting example (sample 20593), the model predicted "Donald Trump And Hillary Clinton In New Hampshire" instead of the actual headline "Bernie Sanders Surpasses Hillary Clinton In New Hampshire Poll." This discrepancy may indicate that the model is overfitting to certain recurring patterns, causing it to favor familiar combinations (like Donald Trump and Hillary Clinton) even when the context suggests otherwise.

Although LoRA allowed us to train a large-scale model with significantly fewer trainable parameters, the corresponding improvements of BLEU were incremental. The stability of the loss values across epochs suggests that after a certain point, further gains in quality may require either more advanced fine-tuning strategies or additional decoding techniques.

The BLEU score improved throughout training, reaching around 50% by the final epoch. While this progress is encouraging, we aslo saw that the BLEU scores are heavily skewed towards zero. However, it's important to note that BLEU, being an n-gram based metric, may not fully capture aspects like contextual relevance. A low BLEU score does not necessarily indicate a poor headline if the message is effectively conveyed. In fact, we observed that sometimes the BLEU score reached 50% simply because one letter was lowercase instead of

uppercase, which suggests that the metric can be overly sensitive to minor differences.

Finally, we note that the model was trained on a rather limited number of epochs, and with rather small batch sizes. It is possible that scaling such up, would lead to inherently better performance, without other adjustments.

Therefore, future evaluations should consider complementing BLEU with additional metrics such as ROUGE and human judgment to achieve a more comprehensive assessment of headline quality.

# 6 Future Work

Building on our findings, future work should focus on refining the model to address current limitations. Some previous challenges have been observed in transformer-based headline generation models. Li et al. (6) improved output quality by incorporating specialized decoding strategies such as top-k sampling, top-p sampling, and repetition penalties, which effectively reduced repetitive and incoherent outputs. Consequently, exploring and adopting these more advanced decoding techniques could be a valuable direction for improving our model.

Another interesting approach for future work is personalized headline generation, where headlines are tailored based on individual readers' interests and prior engagement. As noted by Cai et al. (14), personalized headline generation can significantly boost user engagement, although it also introduces challenges in balancing conciseness, relevance, and user specificity. Future work could then investigate leveraging readers' historical interactions to select signature phrases that resonate with their preferences, thereby enhancing headline effectiveness.

# 7 Conclusion

This project explored the feasibility of using a pretrained transformer model, BART, for automated news headline generation, with a particular focus on integrating Low-Rank Adaptation (LoRA) for parameter-efficient fine-tuning. Our primary research questions examined how effectively BART can generate headlines from article descriptions and the impact of LoRA on computational efficiency.

Using the Hugging Face platform, we fine-tuned the model with LoRA configured to update only a small subset of parameters. This approach allowed us to significantly reduce RAM usage and training time, with training and validation losses remaining relatively stable, demonstrating that LoRA can achieve performance levels similar to full fine-tuning while drastically reducing computational overhead.

Quantitative evaluations showed that the BLEU score improved steadily, reaching approximately 50% by the final epoch. However, the BLEU metric was sometimes overly sensitive to minor discrepancies, suggesting that while it offers a useful baseline, it may not fully capture headline quality, including creativity, contextual relevance, and emotional appeal. Therefore, future assessments should integrate additional metrics like ROUGE and human evaluations.

Qualitative analysis revealed that many generated headlines were both readable and relevant, often capturing the core ideas of the article descriptions. In several cases, the generated headlines were nearly identical to the actual headlines. Yet, there were instances where the model produced headlines that either oversimplified or missed key contextual subtleties.

In summary, our findings demonstrate that integrating LoRA into the fine-tuning process is a promising strategy for efficient headline generation. While the model shows potential in producing coherent and relevant headlines, there remains room for improvement in captur-

ing nuanced details. Future work should explore advanced decoding strategies, additional evaluation metrics, and possibly personalized headline generation to further enhance both the performance and quality of generated headlines.

# References


[1] https://huggingface.co/facebook/bart-base.

[2] https://thereader.mitpress.mit.edu/headlines-journalism/.

[3] D. Dor, "On newspaper headlines as relevance optimizers," *Journal of pragmatics*, vol. 35, no. 5, pp. 695–721, 2003.

[4] J. M. Scacco and A. Muddiman, "The current state of news headlines," *Centre for Media Engagement*, 2015.

[5] A. Evert, J. Genander, N. Lallo, R. Lantz, and F. Nilsson, "Generating headlines with recurrent neural networks," 2016.

[6] Z. Li, J. Wu, J. Miao, and X. Yu, "News headline generation based on improved decoder from transformer," *Scientific Reports*, vol. 12, no. 1, p. 11648, 2022.

[7] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," *Advances in neural information processing systems*, vol. 28, 2015.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[10] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[12] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02. USA: Association for Computational Linguistics, 2002, p. 311–318. [Online]. Available: https://doi.org/10.3115/1073083.1073135

[13] R. Misra, "News category dataset," *arXiv preprint arXiv:2209.11429*, 2022.

[14] P. Cai, K. Song, S. Cho, H. Wang, X. Wang, H. Yu, F. Liu, and D. Yu, "Generating user-engaging news headlines," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 3265–3280.

# A Code

## A.1 BART Model

```
model_name = "facebook/bart-base"
tokenizer = BartTokenizer.from_pretrained(model_name)
base_model = BartForConditionalGeneration.from_pretrained(model_name)

lora_config = LoraConfig(
    task_type=TaskType.SEQ_2_SEQ_LM,
    r=8,
    lora_alpha=32,
    lora_dropout=0.1,
    target_modules=["q_proj", "v_proj"]
)

model = get_peft_model(base_model, lora_config)
model.print_trainable_parameters()
```

## A.2 Preprocessing

```
max_input_length = 250
max_target_length = 30

def preprocess_function(examples):
    model_inputs = tokenizer(inputs, max_length=max_input_length, truncation=True)
    with tokenizer.as_target_tokenizer():
        labels = tokenizer(targets, max_length=max_target_length, truncation=True)
    model_inputs["labels"] = labels["input_ids"]
    return model_inputs

tokenized_datasets = dataset.map(
    preprocess_function,
    batched=True,
    remove_columns=dataset["train"].column_names
)

train_dataset = tokenized_datasets["train"]
eval_dataset = tokenized_datasets["test"]
```

## A.3 BLEU Metric

```
data_collator = DataCollatorForSeq2Seq(tokenizer, model=model)
bleu = evaluate.load("bleu")

def compute_metrics(eval_pred):
    predictions, labels = eval_pred
    decoded_preds = []
    for pred_ids in predictions:
        try:
            text = tokenizer.decode(pred_ids, skip_special_tokens=True)
            decoded_preds.append(text)
        except:
            decoded_preds.append("")

    labels = np.where(labels != -100, labels, tokenizer.pad_token_id)
    decoded_labels = []
    for label_ids in labels:
        try:
```

```
            text = tokenizer.decode(label_ids, skip_special_tokens=True)
            decoded_labels.append(text)
        except:
            decoded_labels.append("")

    result = bleu.compute(
        predictions=decoded_preds,
        references=[[ref] for ref in decoded_labels]
    )
    return {"bleu": result["bleu"] * 100}
```

## A.4 Training and Prediction

```
training_args = Seq2SeqTrainingArguments(
    output_dir="/content/drive/MyDrive/checkpoints",
    evaluation_strategy="epoch",
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    gradient_accumulation_steps=1,
    learning_rate=2e-5,
    num_train_epochs=10,
    weight_decay=0.01,
    save_strategy="epoch",
    logging_steps=100,
    predict_with_generate=True,
    fp16=True,
    report_to=[],
    label_smoothing_factor=0.1
)

trainer = Seq2SeqTrainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset,
    data_collator=data_collator,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)

trainer.train()
eval_results = trainer.evaluate()
print("Evaluation results:", eval_results)

pred_obj = trainer.predict(eval_dataset)
predictions = pred_obj.predictions
label_ids = pred_obj.label_ids
```