Natural Language Processing
Assignment 2 - Report

Pon Vinayak Vickram M
24CS60R52

April 2025

Design Decision Part-A:Dataset and Preprocessing

This section outlines the design choices made during the data preparation stage for training a title generation model using Wikipedia articles. The goal was to improve data quality, reduce noise, and make the dataset more suitable for sequence-to-sequence modeling.

Dataset Splitting: Validation Set Extraction

Decision: Extracted 500 samples randomly from the original training dataset to form a validation/development set.

Rationale: A validation set is essential for tuning model hyperparameters, applying early stopping, and monitoring overfitting. A size of 500 (around 3.5% of the training data) is a reasonable balance between training data availability and validation reliability.

Text Cleaning

- **Punctuation and Non-ASCII Removal**: Regular expressions were used to remove punctuation symbols and characters outside the ASCII range.

  Rationale: These characters do not significantly contribute to title generation and may increase vocabulary size or introduce encoding issues.

- **Lowercasing**: All text was converted to lowercase.

  Rationale: This helps in reducing vocabulary size and normalizing words that would otherwise be treated as different tokens (e.g., "The" vs. "the").

Stopword Removal

Decision: Used NLTK's predefined list to remove English stopwords.

Rationale: Stopwords often do not carry meaningful information for tasks like summarization or title generation. Removing them can help the model focus on more relevant content words, potentially improving learning efficiency.

Lemmatization

Decision: Applied lemmatization using WordNetLemmatizer from NLTK.

Rationale: Lemmatization reduces words to their base or dictionary forms (e.g., "running" → "run"), which helps reduce vocabulary size without losing grammatical correctness, making it more appropriate than stemming for sequence generation tasks.

Additional Preprocessing: Sentence Truncation

Decision: Limited the input text length by truncating articles to a fixed number of tokens (e.g., 512).

Rationale: Transformer models typically have a maximum sequence length. Truncating the text ensures compatibility and efficiency during training while retaining the most relevant information (usually present in the beginning of Wikipedia articles).

RNN-based Title Generation: Results, Analysis and Design Decisions

It summarizes the design decisions, enhancements, and experimental results of RNN-based sequence-to-sequence (Seq2Seq) models applied to the task of title generation from Wikipedia article texts. We explored a variety of architectures and decoding strategies and evaluated each using ROUGE metrics (ROUGE-1, ROUGE-2, ROUGE-L) on a test set.

1. Baseline: Basic RNN Seq2Seq with Greedy Decoding

Architecture: The model comprises a bidirectional GRU encoder and a unidirectional GRU decoder. The encoder processes the article text and outputs a condensed hidden representation, which initializes the decoder. Decoding is performed greedily.

Design Rationale:

- Bidirectional GRU in the encoder captures both past and future context in the input.
- Unidirectional GRU in the decoder suits left-to-right generation.

- Teacher forcing during training stabilizes learning; greedy decoding enables fast inference.

Training Time: 7 minutes

Best performing Hyperparameters: max_len = 6, batch_size = 16, epochs = 6 (early stopping)
Scores: ROUGE-1: 0.2933, ROUGE-2: 0.0984, ROUGE-L: 0.2905
Analysis: This baseline confirms that even simple Seq2Seq models can learn meaningful title generation from input text.

2. Seq2Seq with GloVe Embeddings

Enhancement: The embedding layer was initialized with pre-trained 300-dimensional GloVe vectors (Wikipedia 6B), which were kept static during training.
Design Rationale:

- GloVe vectors enrich semantic understanding from large external corpora.
- Pretrained embeddings enhance generalization and often reduce training time.

Training Time: 7 minutes

Best performing Hyperparameters: Same as baseline; epochs = 4
Scores: ROUGE-1: 0.2935, ROUGE-2: 0.1017, ROUGE-L: 0.2935
Analysis: While scores slightly improved, the gain was limited due to vocabulary mismatch and static embeddings.

3. Hierarchical Encoder RNN (HierEncoderRNN)

Architecture: A two-level encoder with a word-level GRU and a sentence-level GRU. Word-level outputs are averaged per sentence and passed to the sentence-level GRU.
Design Rationale:

- Designed to capture both local (intra-sentence) and global (inter-sentence) semantics.
- Suitable for longer, structured documents like Wikipedia articles.

Training Time: 7 minutes

Best performing Hyperparameters: max_len = 8, batch_size = 32, epochs = 7
Scores: ROUGE-1: 0.1252, ROUGE-2: 0.0107, ROUGE-L: 0.1252
Analysis: Performance dropped significantly due to potential loss of token-level detail and increased model complexity. Larger batch size may have also contributed to underfitting.

4. Decoder2RNN (Two-layer GRU Decoder)

Enhancement: The decoder consists of two stacked GRU layers. Output from the first GRU is refined by the second before projection.

Design Rationale:

- Increases decoder capacity to better model complex output dependencies.
- The second GRU layer allows deeper contextual processing before word prediction.

Training Time: 7 minutes

Best performing Hyperparameters: max_len = 8, batch_size = 16, epochs = 6

Scores: ROUGE-1: 0.3250, ROUGE-2: 0.1166, ROUGE-L: 0.3250

Analysis: This configuration achieved the best results, validating that decoder improvements can significantly enhance generation quality.

5. Beam Search Decoding

Enhancement: Replaced greedy decoding with beam search (beam width k).

Design Rationale:

- Beam search explores multiple high-probability sequences at each step.
- Aims to produce more coherent and fluent outputs.

Training Time: 7 minutes

Best performing Hyperparameters: max_len = 8, batch_size = 16, epochs = 5

Scores: ROUGE-1: 0.2683, ROUGE-2: 0.0854, ROUGE-L: 0.2661

Analysis: Slight drop in performance, possibly due to suboptimal beam width, lack of length penalty, or redundancy in output. Greedy decoding sufficed for the short outputs in this task.

6. Combined Model: HierEncoderRNN + Decoder2RNN

Architecture: Combined the hierarchical encoder and the two-layer decoder to leverage the strengths of both.

Design Rationale:

- Intended to jointly capture document structure and improve decoding.

Training Time: 7 minutes

Best performing Hyperparameters: max_len = 6, batch_size = 16, epochs = 6

Scores: ROUGE-1: 0.1752, ROUGE-2: 0.0107, ROUGE-L: 0.1742

Analysis: Performance degraded due to over-complexity and insufficient data to train deep models effectively. The combination increased training instability.

Final Observations and Recommendations

- **Decoder improvements** (like Decoder2RNN) had a more positive impact than encoder complexity.
- **Simpler models** often generalize better in low-resource settings.
- **Pretrained embeddings** help marginally, but fine-tuning may be necessary for significant gain.
- **Beam search** is not always superior—requires careful tuning and is data-sensitive.

Hyperparameter Summary

- **Hidden Dimension:** 300
  - This matches the size of the GloVe embeddings used, ensuring a seamless dimensional transition between input embeddings and the encoder. A dimension of 300 is commonly used and provides a good balance between expressiveness and computational efficiency.
- **Batch Size:** 16 or 32 (tuned)
  - The batch size was tuned based on available GPU memory. A smaller batch size like 16 ensures stable gradient updates when memory is limited, while 32 can improve training speed and generalization when resources allow.
- **Max Output Length:** 6 or 8 tokens (tuned based on GPU memory)
  - Titles are generally short, so a maximum of 6–8 tokens was sufficient to capture meaningful headlines. Keeping the output length short reduces decoding time and GPU memory usage, helping optimize inference efficiency.
- **Teacher Forcing Ratio:** 0.5
  - A 50% teacher forcing ratio balances learning from ground-truth sequences and encouraging the model to rely on its own predictions. This helps avoid overfitting and improves robustness during inference.
- **Learning Rate:** 0.001 (Adam optimizer)
  - The learning rate of 0.001 is a standard starting point for the Adam optimizer. It provides stable and efficient convergence for training RNN-based sequence models.

Early Stopping Strategy

To prevent overfitting and ensure efficient training, **early stopping** was implemented based on validation loss. The training process continuously monitored the model's performance on the validation set after each epoch.

- **Patience Counter:** 3 Training was halted if the validation loss did not improve for 3 consecutive epochs, allowing the model to stop training once performance plateaued.
- **Validation-Based Monitoring:** The validation set provided an unbiased signal of generalization performance, guiding the early stopping mechanism to avoid overfitting to the training data.
- **Justification:** A patience value of 3 offered a good trade-off between giving the model enough chances to recover and avoiding unnecessary training cycles. This helped reduce overfitting and improve training efficiency.

0.1 Model Variants and ROUGE Score Comparison

Table 1 summarizes the ROUGE evaluation metrics for different variants of the Seq2Seq model. Each configuration introduces a change to the baseline to observe its impact on performance.

Table 1: ROUGE Scores for Different Model Variants

Model Variant	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
Vanilla Seq2Seq	0.2933	0.0984	0.2905
+ GloVe Embeddings	0.2935	0.1017	0.2935
+ Hierarchical Encoder	0.1252	0.0107	0.1252
+ Decoder2RNN	0.3250	0.1166	0.3250
+ Beam Search	0.2683	0.0854	0.2661
Improved Model	0.1752	0.0107	0.1742

Part C: Transformer-based Title Generation using T5 — Results, Analysis and Design Decisions

Objective

The objective of this part is to fine-tune a pretrained Transformer-based sequence-to-sequence (Seq2Seq) model—`google-t5/t5-small`—for the task of generating article titles from text using the provided dataset. The task involves using the raw (non-preprocessed) form of the dataset and evaluating the model's performance using ROUGE metrics.

Motivation and Model Selection

Transformer architectures like T5 have revolutionized NLP by relying entirely on self-attention mechanisms. They enable efficient parallelization, superior long-range dependency handling, and strong performance in sequence-to-sequence tasks such as summarization and title generation. Unlike RNNs that suffer from vanishing gradients and sequential bottlenecks, T5's encoder-decoder structure captures context more effectively.

We selected `google-t5/t5-small` from HuggingFace due to:

- Pretraining on the large C4 corpus using span-masking objectives
- Strong performance across many NLP tasks in a unified text-to-text framework
- Compatibility with HuggingFace's ecosystem for easy fine-tuning and evaluation

Preprocessing and Tokenization

- Raw, unprocessed articles were used without stopword removal or punctuation stripping, aligning with T5's pretraining setup.
- Tokenized using `AutoTokenizer.from`$_{p}retrained(”t5-small”).Inputformat:$ "summarize: <article text>".
- Maximum input length: 512 tokens (longer inputs truncated).

- Tokenizer used with `padding='max`$_{l}ength' and$`truncation=True`.

Training Strategy

- Frameworks: HuggingFace's `transformers` and `datasets`.
- Fine-tuning via `Seq2SeqTrainer` with customized `Seq2SeqTrainingArguments`.
- Hyperparameters:
  - **Learning rate:** Tuned across 3×10^{-5} to 5×10^{-4}
  - **Batch size:** Set to 8 or 16 based on available GPU memory
  - **Evaluation strategy:** `"epoch"` or `"steps"` based on ROUGE metrics
- Early stopping on validation loss (500 validation samples used).
- Beam search enabled during evaluation with varying beam widths.

Training Time: Approximately 58 minutes on available GPU hardware.

Decoding Strategies

- **Greedy Decoding:** Selects the highest probability token at each step.
- **Beam Search Decoding:** Retains top-k sequences at each step for improved output fluency. Beam widths such as 3 and 5 explored.
- Post-processing involves decoding token IDs into strings and removing padding/special tokens.

Evaluation Results

ROUGE scores were computed using HuggingFace's `evaluate.load("rouge")`:

- **Greedy Decoding:**
  - ROUGE-1: 0.8744
  - ROUGE-2: 0.6615
  - ROUGE-L: 0.8755

  arduino Copy Edit
- **Beam Search (beam width = 4):**
  - ROUGE-1: 0.8864
  - ROUGE-2: 0.6748
  - ROUGE-L: 0.8855

Analysis and Observations

- **Model Efficiency:** T5 demonstrated significantly improved performance over RNN-based models (Parts A and B) due to its global self-attention and large-scale pretraining.
- **Use of Raw Text:** Preserving stopwords and punctuation led to better alignment with the model's pretraining data, enhancing generalization.
- **Decoding Techniques:** Beam search offered improved output fluency and content coverage compared to greedy decoding.
- **Generalization:** Despite limited fine-tuning data, the pretraining allowed T5 to generalize effectively and generate coherent, relevant titles.

Part C2: Prompt Engineering with Flan-T5

This section explores the effectiveness of prompt engineering using instruction-tuned models for zero-shot title generation. We investigate how prompt phrasing and model size impact the quality of generated titles, without any additional fine-tuning, using the `google/flan-t5-base` and `google/flan-t5-large` models.

Results and Analysis

We evaluated two prompt variations with both model variants, using greedy decoding and computing ROUGE scores to assess performance:

- **Flan-T5 Base + Prompt V1** (`"Generate a title for the following article: <article text>"`):
  - ROUGE-1: 0.7958
  - ROUGE-2: 0.5688
  - ROUGE-L: 0.7918

  yaml Copy Edit

- **Flan-T5 Base + Prompt V2** (`"Write a concise and informative title for this news article: <article text>"`):
  - ROUGE-1: 0.5967
  - ROUGE-2: 0.4059
  - ROUGE-L: 0.5967
- **Flan-T5 Large + Prompt V1**:
  - ROUGE-1: 0.8688
  - ROUGE-2: 0.6410
  - ROUGE-L: 0.8688
- **Flan-T5 Large + Prompt V2**:
  - ROUGE-1: 0.7425
  - ROUGE-2: 0.5460
  - ROUGE-L: 0.7425

Key Observations:

- **Prompt Impact:** Prompt V1 consistently outperforms Prompt V2 across both models, suggesting that phrasing the instruction more directly and generically leads to better adherence and title relevance.
- **Model Size Effect:** Flan-T5 Large significantly outperforms the base variant, confirming the positive impact of model capacity on zero-shot generation quality.
- **Zero-shot Strength:** Despite no task-specific fine-tuning, the best configuration (Flan-T5 Large + Prompt V1) achieves high ROUGE scores, comparable to supervised baselines.

Design Decisions

- **Model Choice:** We selected `google/flan-t5-base` and `google/flan-t5-large` for their instruction-following capabilities and strong performance in zero-shot tasks. The base model is efficient, while the large model offers better generative quality.

  css Copy Edit

- **Prompt Engineering Strategy:** Two prompt formats were crafted to evaluate how different phrasings affect model behavior:

  - **Prompt V1:** "Generate a title for the following article:"
  - **Prompt V2:** "Write a concise and informative title for this news article:"

  The goal was to test the impact of tone (generic vs. specific) and prompt length on output quality.

- **Dataset Usage:** Raw article texts were used directly—without preprocessing—to align with how instruction-tuned models are designed to interpret natural input text.

- **Inference Strategy:** Greedy decoding was used for simplicity and speed, and inference was run in batches to manage GPU memory and throughput, especially for the large model.

- **Evaluation Metrics:** ROUGE-1, ROUGE-2, and ROUGE-L were used to measure unigram, bigram, and longest common subsequence overlap with ground-truth titles.