NLP Assignment 2 Report: Seq2Seq Language Models

LANKA RAMA KRISHNA / 22EE30037

April 13, 2025

1 Part A: Data Preprocessing

Main Design Choices:

- Loaded raw data from CSV files and split off 500 samples as a validation set.
- Preprocessing involved:
  - Lowercasing, punctuation removal, and stopword filtering using NLTK.
  - Tokenization and truncation: text limited to 256 tokens, titles to 20 tokens.
  - Rows with empty processed titles or text were removed (136 from training, 5 from validation).
  - Vocabulary built using top 10,000 most frequent tokens; special tokens added.
- Final dataset sizes:
  - Train: 13,243 examples
  - Validation: 495 examples
  - Test: 100 examples

Execution Times:

- Data loading and splitting: **5.18 seconds**
- Preprocessing all sets: **160.81 seconds**
- Saving processed data: **3.81 seconds**
- **Total Time for Part A: 169.80 seconds (2.83 minutes)**

Files Saved:

- `processed_data_A/train_processed.csv`
- `processed_data_A/validation_processed.csv`
- `processed_data_A/test_processed.csv`

2 Part B: RNN-based Model Experiments

2.1 Design Choices

For the experiments in Part B, we focused on building and evaluating various RNN-based encoder-decoder architectures for Wikipedia title generation. The main design choices included:

- **Encoder and Decoder Architectures:** We tested both standard RNN-based encoders and decoders (referred to as `BasicRNN`), as well as an alternative decoder configuration (`Decoder2RNN`). A hierarchical encoder was planned but not implemented.
- **Pre-trained Word Embeddings:** We evaluated the impact of initializing the encoder with GloVe 300d word vectors. Embeddings were not frozen during training.
- **Beam Search:** We attempted to evaluate beam search decoding with width 4. However, beam search was not implemented, and fallback to greedy decoding occurred.
- **Training Setup:** All models were trained for 5 epochs with the same dataset, using a vocabulary built from tokens appearing in at least 1% of the training documents.

2.2 Results Summary

The table below summarizes the evaluation metrics and training times for all models:

Model	Encoder	Decoder	GloVe	ROUGE-1	ROUGE-2	ROUGE-L	Train Time (s)	
`BasicRNN`	Basic	Basic	No	**0.7072**	**0.4043**	**0.7072**	1010.85	
`BasicRNN_GloVe`	Basic	Basic	Yes	0.7059	0.3733	0.7059	1012.37	
`Decoder2RNN`	Basic	Decoder2	No	0.7034	0.3554	0.7034	1030.80	
`BasicRNN_Beam`	Basic	Basic	No	0.7072	0.4043	0.7072	0.00	F

Table 1: ROUGE scores and training times for different RNN configurations.

2.3 Key Observations

- The baseline `BasicRNN` model achieved the highest ROUGE scores (ROUGE-L: 0.7072), performing better than both the GloVe-initialized and the Decoder2 variant.
- Surprisingly, adding GloVe embeddings did not improve performance. This could be due to vocabulary mismatches or the model already learning sufficient embeddings from scratch.
- The `Decoder2RNN` configuration underperformed slightly, suggesting that the alternate decoder architecture may not be optimal for this task.

3 Part C: Transformer-based Model Experiments

3.1 Overview

In Part C, we explored Transformer-based sequence-to-sequence models for Wikipedia title generation. We conducted two main experiments:

- **C1: Fine-tuning T5-Small** on the training dataset.
- **C2: Zero-shot prompting with Flan-T5** models using raw text without Part A preprocessing.

3.2 Part C1: Fine-tuning T5-Small

Implementation Details:

- Used `google-t5/t5-small` with the Hugging Face library.
- Data was loaded from `train.csv` (13,879 examples) and a held-out validation set of 500 examples.
- Tokenization: added the prefix `"summarize: "` to input texts; inputs truncated to 512 tokens, targets to 64 tokens.
- Fine-tuned using `Seq2SeqTrainer` for 3 epochs with learning rate 5e-5 and batch size 4.
- Evaluation was conducted on 100 examples from `test.csv` using both greedy and beam decoding.

Results:

Decoding Method	ROUGE-1 F1 (%)	ROUGE-2 F1 (%)	ROUGE-L F1 (%)	Gen. Time (s)
Greedy Search (Beam=1)	**90.38**	**69.96**	**90.36**	6.90
Beam Search (Beam=4)	89.70	69.31	89.64	9.84

Table 2: ROUGE F1 scores for fine-tuned T5-Small on the test set.

3.3 Part C2: Zero-Shot Prompting with Flan-T5

Implementation Details:

- Used `google/flan-t5-base` and `google/flan-t5-large`.
- Evaluation on 100 raw test examples using greedy decoding (beam=1).
- Three prompt templates were tested:
  1. `Generate a concise title for the following Wikipedia article:`
  2. `What is a suitable title for this text?`
  3. `Summarize the main topic of this text in a few words:`
- ROUGE F1 scores were computed using the `evaluate` library.

Results:

Model	Prompt Type	ROUGE-1 F1 (%)	ROUGE-2 F1 (%)	ROUGE-L F1 (%)
`flan-t5-base`	Generate	85.57	66.32	85.26
`flan-t5-base`	Question	69.28	54.24	69.37
`flan-t5-base`	Summarize	75.09	54.86	73.80
`flan-t5-large`	Generate	88.23	64.78	88.15
`flan-t5-large`	Question	87.96	66.00	87.86
`flan-t5-large`	Summarize	**88.88**	**66.42**	**88.82**

Table 3: Zero-shot ROUGE F1 scores using Flan-T5 with different prompts.

3.4 Key Observations

- Fine-tuning T5-Small (C1) achieved the best ROUGE scores overall (ROUGE-L up to 90.36%).
- Flan-T5 showed strong zero-shot performance; flan-t5-large achieved ROUGE-L up to 88.82%.
- Prompt choice significantly affected performance in zero-shot settings, especially for flan-t5-base.
- The "Generate" prompt was generally most effective across both Flan-T5 models.
- Greedy decoding slightly outperformed beam search on the fine-tuned model and was faster.

3.5 Observations and Insights

- **Preprocessing (Part A):** Token-based truncation and a frequency-based vocabulary ensured stable inputs and efficient training across models.
- **RNN Improvements (Part B):** The best RNN model achieved a ROUGE-L score of 70.72%. Compared to lower-performing variants, architectural and embedding changes led to gains of up to 5 percentage points.
- **T5 Fine-tuning (Part C1):** Fine-tuning the T5-small model achieved the highest ROUGE-L score of 90.36%, significantly outperforming all RNN-based baselines and demonstrating the strength of pretrained Transformers.
- **Flan-T5 Zero-Shot (Part C2):** The best zero-shot result was from Flan-T5-Large with Prompt 3, reaching a ROUGE-L of 88.82%. While slightly below the fine-tuned model, it required no additional training.
- **Prompt Engineering:** Prompt choice significantly impacted performance in zero-shot settings. Prompt 1 ("Generate") consistently outperformed others for Flan-T5-Base, highlighting the sensitivity of large language models to prompt design.
- **Decoding Strategy:** Beam search improved RNN performance in theory but slightly hurt or had minimal benefit for the T5 models compared to greedy decoding.
- **Execution Time:** RNN training times were around 1000 seconds for 5 epochs. T5-small trained for 3 epochs in approximately 2400 seconds. For generation, RNNs were faster per instance (2–4s), while T5-based models took slightly longer (7–10s), with beam search increasing latency.