BME646 and ECE60146: Homework 10

Spring 2025
Due Date: 11:59pm, Apr 14, 2025
TA: Akshita Kamsali (akamsali@purdue.edu)

Turn in typed solutions via Gradescope. Post questions to Piazza. Additional instructions can be found at the end. Late submissions will be accepted with penalty: -10 points per-late-day, up to 5 days.

1 Introduction

Transformer models have revolutionized natural language processing through their ability to handle longer range dependencies between the words compared to what could be achieved with RNN based implementation using GRUs and LSTMs. These models achieve state-of-the-art results across various NLP tasks, including machine translation, text classification, and text generation.

In this homework, we will compare the performance of two Transformer models — TransformerFG and TransformerPreLN — on a machine translation task. The TransformerFG model is implemented with a standard feed-forward architecture, while the TransformerPreLN model integrates Layer Normalization before going through the Multi-Headed Attention (MHA) layer.

We will use the Levenshtein distance, a commonly used metric for measuring the similarity between two sequences. The Levenshtein distance calculates the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one sequence into another.

2 Getting Ready for This Homework

Before embarking on this homework, do the following:

1. Carefully review Week 13 slides on "Transformers". Make sure you understand how Self-Attention, Cross-Attention and Multi-Headed Attention works

2. Download the dataset required for transformer learning from DLStudio website. https://engineering.purdue.edu/kak/distDLS/en_es_corpus_for_learning_with_Transformers.tar.gz

3. Make sure you have the latest version of DLStudio installed and setup.

   Note: that there is a module packaging error in Version 2.5.2 (currently the latest version) of DLStudio. You need to cd into the following directory:

   ```
   cd   installation_dir/Transformers/
   ```

   and edit the following file

   ```
   __init__.py
   ```

   Comment out the following line in the above file:

   ```
   from Transformers.Transformers import AutoencoderWithTransformer
   ```

3 Programming Tasks

1. Run the `seq2seq_with_transformerFG.py`.
2. Run the `seq2seq_with_transformerPreLN.py`.
3. Train both models for 40 epochs (this will take approximately 2 hours on a CPU).
4. In your report, include the loss plots of both models.
5. In your report, include 5 examples of outputs same input to both the models.
6. The model outputs contain stop words such as `SOS` and `EOS`. Remove these stop words from the output before computing the Levenshtein distance.
7. Compare the cleaned outputs with the ground truth using the following Levenshtein distance implementation.

```
def levenshtein_distance(str1, str2):
    if len(str1) < len(str2):
        str1, str2 = str2, str1  # Ensure str1 is the
                                 longer string

    len_str1, len_str2 = len(str1), len(str2)

```

```
    # Initialize two rows for dynamic programming
    previous_row = list(range(len_str2 + 1))
    current_row = [0] * (len_str2 + 1)

    for i in range(1, len_str1 + 1):
        current_row[0] = i
        for j in range(1, len_str2 + 1):
            cost = 0 if str1[i - 1] == str2[j - 1] else 1
            current_row[j] = min(
                previous_row[j] + 1,      # Deletion
                current_row[j - 1] + 1,   # Insertion
                previous_row[j - 1] + cost # Substitution
            )
        previous_row, current_row = current_row,
                                    previous_row
    return previous_row[-1]
```

8. For each model, calculate and report the following statistics in a table of 2 size, one row per model and one column per metric:

    - Mean
    - Median
    - Standard Deviation (std)
    - Maximum
    - Minimum

```
import numpy as np

def report_statistics(distances):
    print(f"Mean: {np.mean(distances):.2f}")
    print(f"Median: {np.median(distances):.2f}")
    print(f"Standard Deviation: {np.std(distances):.2f}")
    print(f"Maximum: {np.max(distances):.2f}")
    print(f"Minimum: {np.min(distances):.2f}")

# Example usage with dummy distances
distances_fg = [3, 5, 2, 1, 4, 6, 7, 5, 4]
distances_ln = [2, 4, 1, 3, 5, 6, 4, 2, 3]

print("TransformerFG Statistics:")
report_statistics(distances_fg)

print("\nTransformerLN Statistics:")
report_statistics(distances_ln)
```

9. After processing the model outputs and calculating the Levenshtein distance, report the comparison statistics for each model. Discuss which model performed better based on the statistics.

4 Submission Instructions

Include a typed report explaining how you solved the given programming tasks. You may refer to the homework solutions posted at the class website for the previous years for examples of how to structure your report

1. **Turn in a PDF file and mark all pages on gradescope**.
2. Submit your code files(s) as zip file.
3. **Code and Output Placement:** Include the output directly next to the corresponding code block in your submission. Avoid placing the code and output in separate sections as this can make it difficult to follow.
4. **Output Requirement:** Ensure that all your code produces outputs and that these outputs are included in the submitted PDF. Submissions without outputs may not receive full credit, even if the code appears correct.
5. For this homework, you are encouraged to use `.ipynb` for development and the report. If you use `.ipynb`, please convert code to `.py` and submit that as source code. **Do NOT submit .ipynb notebooks.**
6. You can resubmit a homework homework as many times as you want up to the deadline. Each submission will overwrite any previous submission. **If you are submitting late, do it only once.** Otherwise, we cannot guarantee that your latest submission will be pulled for grading and will not accept related regrade requests.
7. The sample solutions from previous years are for reference only. **Your code and final report must be your own work.**