

# ANLP-FALL2025-HW1 Report

Andrew\_id: peterwa2

## Introduction

The baseline code used right-padded sequences without any attention mask. This meant that during inference, the model could attend to padded tokens, which introduced noise and degraded performance. To address this, I implemented an attention mask mechanism that prevents the model from attending to padded tokens.

## Implementation

### run\_llama.py

- **Change:** Added a new command-line argument `--use_attn_mask`

代码块

```
1 parser.add_argument("--use_attn_mask", action="store_true",
2     help="Enable attention mask to ignore padding tokens")
```

- **Change:** Updated `test_with_prompting` to automatically rename output files when attention mask is enabled

代码块

```
1 if args.use_attn_mask:
2     dataset = os.path.basename(args.dev).split("-")[0]
3     args.dev_out = f"{dataset}-dev-advanced-output.txt"
4     args.test_out = f"{dataset}-test-advanced-output.txt"
```

In training/evaluation loops, `attention_mask` from the dataloader is passed into the model:

代码块

```
1 logits = model(b_ids, attention_mask=batch['attention_mask']).to(device))
```

## LlamaDataset class

**Mask construction:** Built a binary attention mask where `1` marks real tokens and `0` marks padding

代码块

```
1 attention_mask = [[1] * len(sentence) + [0] * (max_length_in_batch -  
len(sentence))  
2                 for sentence in encoding]  
3 attention_mask = torch.LongTensor(attention_mask)
```

Returned inside `collate_fn`, so every batch includes:

代码块

```
1 batched_data = {  
2     ...  
3     'attention_mask': attention_mask,  
4 }
```

## llama.py

- **Attention:** Added `attn_mask` argument, used to mask out padded positions in score computation.
- **LlamaLayer:** Forward now calls attention with `attn_mask`.
- **Llama.forward:** Passes the same `attention_mask` down through all layers, and finds the correct last token, since we cannot assume that the last token is the one we want to predict if an attention mask is provided.

代码块

```
1 # Attention  
2 if attn_mask is not None:  
3     attn_scores = attn_scores.masked_fill(attn_mask == 0, float("-inf"))  
4  
5 # LlamaLayer  
6 attn_out = self.attention(norm, attn_mask=attn_mask)  
7  
8 # Llama.forward  
9 for layer in self.layers:  
10     h = layer(h, attn_mask=attention_mask)  
11 ...  
12 if attention_mask is not None:
```

```

13 token_counts = attention_mask.sum(dim=1)
14 last_indices = token_counts - 2 # -2 to get the index of the last token
15 batch_indices = torch.arange(_batch_size, device=h.device)
16 logits = self.output(h[batch_indices, last_indices].unsqueeze(1))
17

```

## Experimental Setup

I evaluated the SST and CFIMDB dataset for binary sentiment classification using zero-shot prompting with the same prompt template as the baseline. The model is a 42M-parameter pretrained Llama backbone, run with right-padded batching. We added a padded attention mask to exclude the pad tokens.

## Result

Dataset	Setting	Dev Accuracy	Test Accuracy
sst	Baseline	0.237	0.250
sst	Attention Mask	<b>0.252</b>	0.233
cfimdb	Baseline	0.490	0.109
cfimdb	Attention Mask	<b>0.551</b>	<b>0.863</b>

The attention mask leads to a **+0.754** improvement in test accuracy for cfimdb, showing that masking padded tokens significantly improves inference quality in the zero-shot prompting setting.

## Usage

Added a `--use_attn_mask` flag to the command line. When enabled, results are saved as `{dataset}-dev-advanced-output.txt` and `{dataset}-test-advanced-output.txt`

代码块

```

1 python run_llama.py \
2     .../ # others arguments
3     --use_attn_mask

```