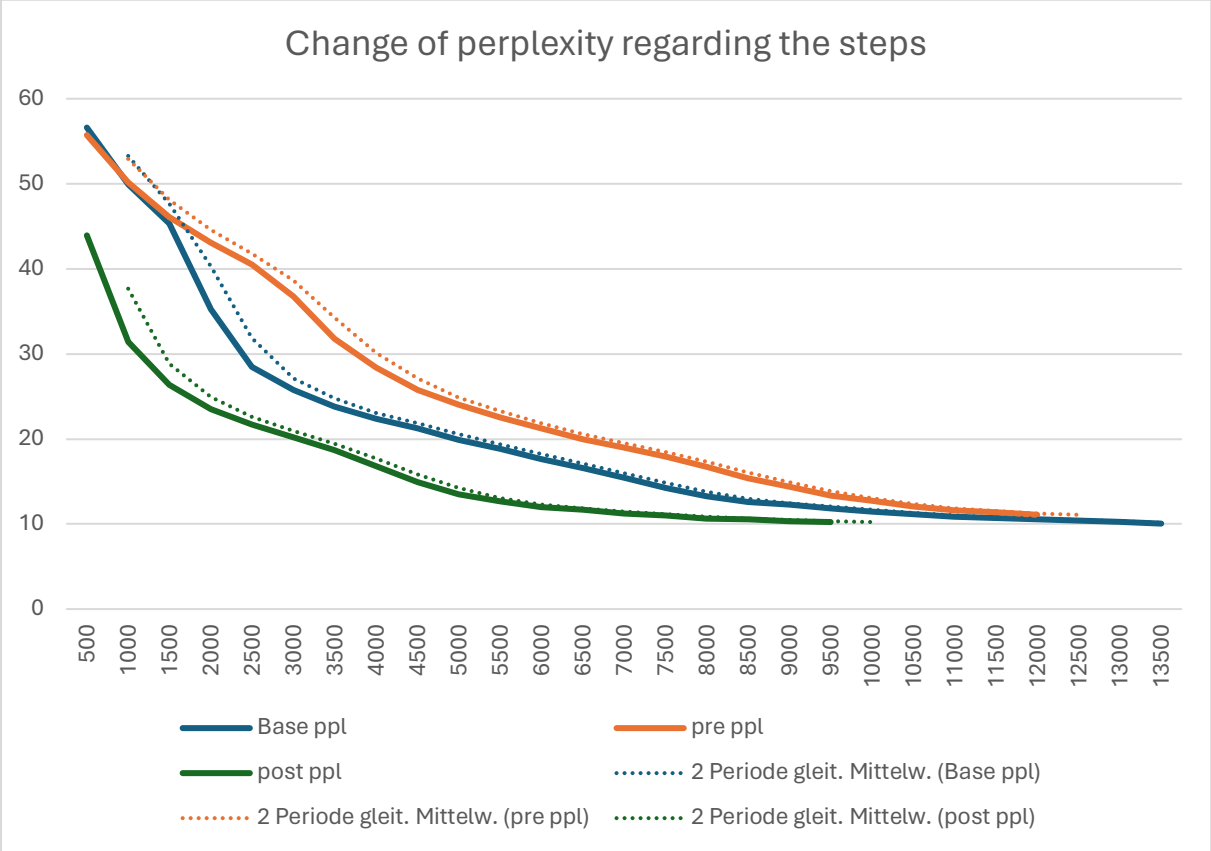
Inhaltsverzeichnis

1.2 Instances of LayerNorm in JoeyNMT

transformer_layers.py

1. **Target-Target Self-Attention**:
    - Before the self-attention operation, the input **x** is stored as **residual**.
    - If **_layer_norm_position** is set to "pre", layer normalization is applied to **x** using **self.x_layer_norm(x)**.
    - The self-attention operation **self.trg_trg_att(x, x, x, mask=trg_mask)** is performed.
    - Dropout is applied to the self-attention output **h1**.
    - The output of the self-attention operation, after dropout, is added to **alpha * residual**, where **alpha** is a scalar parameter.
    - If **_layer_norm_position** is set to "post", layer normalization is applied to **h1** using **self.x_layer_norm(h1)**.
2. **Source-Target Cross-Attention**:
    - Similar to self-attention, layer normalization is applied before and after cross-attention.
    - Before the cross-attention operation, the current output **h1** is stored as **h1_residual**.
    - If **_layer_norm_position** is set to "pre", layer normalization is applied to **h1** using **self.dec_layer_norm(h1)**.
    - The cross-attention operation **self.src_trg_att(memory, memory, h1, mask=src_mask, return_weights=return_attention)** is performed.
    - Dropout is applied to the cross-attention output **h2**.
    - The output of the cross-attention operation, after dropout, is added to **alpha * h1_residual**.
    - If **_layer_norm_position** is set to "post", layer normalization is applied to **h2** using **self.dec_layer_norm(h2)**.

Task 2



Observation:

- As training progresses, all three models show a decrease in perplexity, indicating improvement in their ability to predict the data.
- The pre-model tends to have lower perplexity values compared to the base model, suggesting that applying layer normalization before each step enhances performance.
- The post-model generally exhibits the lowest perplexity values, indicating that layer normalization after each step further improves performance beyond pre-model and base model.

- **Pre-Model**: Applying layer normalization before each training step helps stabilize the activations within each layer. This can mitigate the issues of vanishing or exploding gradients, leading to smoother training progress and faster convergence compared to the base model.
- **Post-Model**: Layer normalization applied after each training step further stabilizes the activations, potentially allowing the model to learn more efficiently. By normalizing the activations after each step, the model may experience less internal covariate shift, leading to faster convergence and more robust training progress compared to both the base and pre models.

Comparison

- Approach with Layer Combination:

    - DLCL dynamically integrates features from preceding layers. Rather than rigidly adhering to pre-norm or post-norm alone, it facilitates a more adaptable amalgamation of layer outputs.
    - In pre-norm Transformers, layer normalization precedes the residual connection, whereas in post-norm Transformers, it follows the residual connection. DLCL diverges from both of these conventions.
    - Through dynamic feature integration, DLCL can adjust to diverse contexts and tasks, potentially enhancing overall performance.
- Enhanced Depth Management:
    - DLCL enables the training of deeper models (e.g., 30/25-layer encoders) without compromising performance. These extended architectures surpass shallower Transformer baselines (e.g., Transformer-Big/Base) by 0.4∼2.4 BLEU points.
    - This advancement is notable because deeper models can capture intricate patterns and dependencies, albeit training them can be hindered by gradient vanishing issues. DLCL assists in mitigating this challenge.
- Reduced Computational Overhead:
    - Despite their increased depth, DLCL-based models exhibit smaller and faster training characteristics. This efficiency is pivotal for practical deployment and scalability.
    - The diminished computational burden is achieved by dynamically selecting pertinent features from preceding layers, thereby circumventing unnecessary computations.
- Generalization and Resilience:
    - DLCL's capacity to fuse features from diverse layers bolsters generalization. It can discern and prioritize relevant information across varying contexts, resulting in superior performance across diverse inputs.
    - Moreover, DLCL's adaptability to different layers' contributions enhances robustness against noise and input data variations.