HSE UNIVERSITY

NLP

Homework 3: Distillation

2025-10-24

Authors:
Sergey Grozny

HSE
2025

Contents

1 | Introduction

Named Entity Recognition (NER) is a fundamental task in Natural Language Processing that aims to identify and classify named entities such as persons, organizations, and locations within text. In this project, we address the NER task using the CoNLL-2003 dataset and a pre-trained BERT model.

The main objective is to reduce the model size to approximately 20 million parameters while maintaining high performance. To achieve this, we apply several model compression techniques, including embedding factorization and knowledge distillation.

2 | Preprocessing

- Before training, the dataset was reorganized to obtain balanced and independent subsets for reliable evaluation. Since the dataset does not include a predefined validation split, we created it manually. Specifically, 20% of the original training data was reserved, and this subset was further divided equally into validation and test sets. This yielded a typical 80/10/10 partition ratio.
- During tokenization, a single word may split into multiple sub-tokens (for example, the word `Fischler`). This leads to a mismatch between the number of tokens and the number of labels, which must be resolved manually.

  To handle this, we align each label with its corresponding token sequence using the function below. It ensures that continuation sub-tokens receive the correct `I-` labels, while non-word tokens (like padding) are ignored. For batch processing, we use the `word_ids` provided by the tokenizer, which allows consistent alignment across examples.

```python
def align_labels_with_tokens(labels, word_ids):
    previous_word_idx = None
    aligned_labels = []
    for word_idx in word_ids:
        if word_idx is None:
            aligned_labels.append(-100)
        elif word_idx != previous_word_idx:
            aligned_labels.append(labels[word_idx])
        else:
            label_id = labels[word_idx]
            label_name = label_names[label_id]
            if label_name.startswith("B-"):
                label_name = label_name.replace("B-", "I-")
                label_id = label_names.index(label_name)
            aligned_labels.append(label_id)
        previous_word_idx = word_idx
    return aligned_labels
```

3 | Teacher training

For the teacher model, we fine-tuned the `bert-base-cased` architecture o using the Hugging Face Trainer API. Since the base model was not initially designed for token-level classification, a token classification head with nine output labels was added to handle the BIO tagging scheme.

Number of parameters: ~107M

The model was trained for three epochs with mixed precision (`bf16`) to accelerate convergence. However, training proved to be *highly sensitive to hyper-parameters*, particularly the **learning rate** and **weight decay**. Small deviations (e.g., increasing the learning rate from `3e-4` to `5e-4`) led to unstable loss dynamics or degraded validation F1 scores, indicating that fine-tuning BERT for NER requires careful optimization.

After tuning, the best configuration achieved:

Final test F1: **0.9452**

4 | Embedding factorization

The embedding layer of BERT is one of its most parameter-heavy components, containing approximately $V \times H = 28,996 \times 768 \approx 22.3$ million parameters — nearly one-fifth of the entire model. To reduce this overhead, we adopt the approach proposed in the ALBERT paper, where the large embedding matrix is factorized into the product of two smaller matrices.

This factorization replaces the single $V \times H$ matrix with two matrices of sizes $V \times E$ and $E \times H$, yielding a total of $V \times E + E \times H$ parameters. For example, setting $E = 64$ lowers the parameter count by roughly 20 million.

In practice, we implemented a custom embedding wrapper class that performs this factorization. Both matrices were initialized using Singular Value Decomposition (SVD) to preserve the structure of the original embeddings and enable faster convergence during fine-tuning.

```
class FactorizedEmbedding(nn.Module):
    def __init__(self, orig_emb, hidden_dim, factor_dim=64,
use_svd_init=True):
        super().__init__()
        self.vocab_size = orig_emb.num_embeddings
        self.hidden_dim = hidden_dim
        self.factor_dim = factor_dim

        self.emb_small = nn.Embedding(self.vocab_size, factor_dim)
        self.proj = nn.Linear(factor_dim, hidden_dim, bias=False)

        if use_svd_init:
            with torch.no_grad():
                W = orig_emb.weight.data.cpu()
                U, S, Vh = torch.linalg.svd(W, full_matrices=False)
                U = U[:, :factor_dim]
                S = S[:factor_dim]
                Vh = Vh[:factor_dim, :]

                W1 = U * S
                W2 = Vh.T

                self.emb_small.weight.copy_(W1)
                self.proj.weight.copy_(W2)

    def forward(self, input_ids):
        x = self.emb_small(input_ids)
        return self.proj(x)
```

Number of parameters after factorization: **87M**

However, it was observed that *training became more sensitive to optimization settings*: using the same learning rate as the original BERT led to unstable convergence and poor F1 scores. After reducing the learning rate and increasing number of epochs, the factorized model achieved

Final test F1: **0.8980**

5 | Student training

To compress the model further, we trained a lightweight student model (≈20M parameters) using knowledge distillation from the fine-tuned BERT teacher. The training objective combined a **soft loss** — the Kullback–Leibler (KL) divergence between the softened logits of the teacher and student — with a **hard loss** based on the true NER labels. The total objective followed the standard DistilBERT formulation, using `nn.KLDivLoss(reduction: "batchmean")`.

One key challenge was the correct handling of the KL divergence inputs: the student's predictions must be in log-probability space (`log_softmax`), while the teacher's outputs must use `softmax` with temperature scaling. Incorrect formatting or temperature usage led to unstable gradients and poor convergence.

Student's number of parameters: **10M**)

Additionally, the training proved highly sensitive to hyperparameters — especially the learning rate, weight decay, and warmup schedule. Without careful tuning, the model failed to surpass an F1 score of 0.6 or lower. After optimization, the student achieved over **0.7248 F1** on the test set, confirming effective knowledge transfer from the teacher.

6 | Results

Model	Parameters	Description	Test F1
BERT (teacher)	107M	Fine-tuned `bert-base-cased`	0.9452
Factorized BERT	87M	Embedding matrix factorized ($E = 64$)	0.8980
Student	10M	Distilled model trained with KL loss	0.7248

7 | Conclusion

The experiments demonstrate that significant model compression for Named Entity Recognition is achievable without catastrophic loss in quality. Fine-tuning the original BERT established a strong baseline with an F1 of 0.9452, serving as an effective teacher for distillation.

Applying embedding factorization reduced the parameter count by approximately 20 million while retaining 95% of the baseline performance, highlighting its efficiency for lightweight deployment.

Finally, knowledge distillation successfully transferred information to a compact 10M-parameter student, which achieved an F1 of 0.7248 after extensive hyperparameter tuning.