COMP 4211: Machine Learning Group Project Report

Contributors::

- LAM, King Cheuk, SID: 20966303, email: kclambd@connect.ust.hk
- YAU, Ho Yin, SID: 20961248, email: hyyauaf@connect.ust.hk

Date: May 2, 2025

Introduction

The COMP 4211 Spring 2025 group project challenges students to solve a named entity recognition (NER) task on Kaggle, predicting entity types (e.g., organization, person, location) for words in sentences. The dataset includes 40,000 labeled training sentences and 5,000 test sentences, evaluated via weighted F1 score on public and private leaderboards. With a baseline private leaderboard score of 0.68, our goal was to develop a model with fewer than 1 billion parameters that maximizes weighted F1 while improving macro F1 for rare classes. We implemented a DeBERTa-base model with a CRF layer, augmented with synonym-based data augmentation, achieving a private leaderboard weighted F1 of 0.856870. This report outlines our methodology, experiments, results, insights, and team contributions, emphasizing the rationale behind our design choices.

Data Processing Methods

Data Loading and Preprocessing

We loaded the training dataset (40,000 sentences with NER tags) from train.csv. Sentences and tags were stored as string representations of lists, which we converted to Python lists using ast.literal_eval. We identified 19 unique NER tags (e.g., B-org, I-per, O) and created label2id and id2label mappings for training.

We noticed some classes (B-art, B-eve, B-nat) were rare, which guided our augmentation strategy.

Data Augmentation

To handle rare classes, we used the nlpaug library with WordNet to augment sentences containing B-art, B-eve, or B-nat tags. We replaced words with synonyms, keeping the original NER tags if the sentence length stayed the same. This added about 10% more examples, saved in augmented_train.csv for efficiency. The augmented dataset was split into 80% training and 20% validation sets with a random seed of 42.

Research and try out

After doing some research, we found out that the BERT base models are commonly used and suitable for solving NER problems. We have tried several models, each training for 3-5 epochs for standard comparison. The macro F1 score is for rare case detection, which adds performance other than generalization:

1. DistilledBERT
    - A smaller, faster version of BERT, created by distilling BERT-base to retain 97% of its performance with 40% fewer parameters (~66M). It's lightweight, efficient.
    -
    - Observation from testing it:
        i. Converges after 3 epochs. The performance gets to ~0.836 F1
        ii. Overfits quickly and cannot be enhanced further

2. BERT-base
    - A transformer model with ~110M parameters, pretrained on large text corpora using masked language modeling. It captures bidirectional context, making it versatile for tasks like NER

Epoch	Training Loss	Validation Loss	Weighted F1	Macro F1
1	0.099400	0.100480	0.967382	0.606701
2	0.077300	0.091728	0.970723	0.620341
3	0.058900	0.093587	0.971428	0.632101

    -
    - Observation from testing it:
        i. Converges after 3 epochs. The performance gets to ~0.845 F1 in kaggle.
        ii. Overfits quickly in epoch 2-3 and cannot be enhanced further in local training

iii. Overall a promising, yet quickly converged macro f1 score

3. RoBerta
   - An optimized version of BERT (~125M parameters) with improved pretraining (more data, longer training, dynamic masking). It outperforms BERT
   - Epoch Training Loss Validation Loss Weighted F1 Macro F1

     1 0.224200 0.208958 0.927320 0.495322

     2 0.187800 0.190753 0.934304 0.542363

     3 0.159500 0.182284 0.938525 0.577342

     4 0.144600 0.176825 0.941162 0.597323

     5 0.126600 0.180562 0.941537 0.597312

   - Observation from testing it:
     i. Overall converges after 4-5 epochs. The performance gets to ~0.85 F1 in kaggle.
     ii. Overfits quickly in epoch 3-4 and cannot be enhanced further in local training
     iii. worse F1 score and quickly converged

4. DeBerta
   - An advanced transformer model (~183M parameters for DeBERTa-base) that enhances BERT with disentangled attention and an improved pre-training process.
   -

Epoch	Training Loss	Validation Loss	Weighted F1	Macro F1
1	0.114500	0.108604	0.966705	0.543059
2	0.096200	0.099488	0.968887	0.587714
3	0.086000	0.095559	0.970216	0.619405
4	0.075800	0.093843	0.970435	0.630588
5	0.067400	0.094438	0.971427	0.644746

- Observation from testing it:
    i. Continuously improving after 5 epochs, the gradient shows that it is yet to be converged
    ii. Macro f1 has huge improvements in each epoch.
    iii. Potential to train further

Decision: Continue to improve using DeBerta Model

Machine Learning / Deep Learning Methods

Model Architecture

We used ‘microsoft/deberta-base‘ (183M parameters), which employs disentangled attention for enhanced context modeling. To address NER’s sequential nature, we developed 2 a custom ‘DebertaCRF‘ class by adding a CRF layer to ‘AutoModelForTokenClassification‘. The CRF models tag sequence dependencies (e.g., ‘B-org‘ → ‘I-org‘), preventing invalid transitions (e.g., ‘B-org‘ → ‘I-per‘). During training, the CRF computes negative log-likelihood loss over valid attention mask regions, and during inference, it outputs the most likely tag sequence via Viterbi decoding. This was chosen because initial trials with DistilBERT and BERT-base showed inconsistent tag sequences for rare classes, which CRF mitigates by enforcing global sequence constraints.

Training Setup

We trained using the Hugging Face Trainer API with these hyperparameters:

- Learning Rate: 1e-5
- Batch Size: 16 (effective batch size 32 with 2 gradient accumulation steps)
- Epochs: 8
- Weight Decay: 0.01
- Evaluation: Per epoch
- Save Strategy: Best model based on macro F1 score

Explanation:

- low rate ensured stable fine-tuning, preventing overfitting on frequent classes (‘O‘, ‘B-geo‘). Higher rates (e.g., 5e-5) caused instability, reducing macro F1.
- An effective batch size of 32 (16 × 2 gradient accumulation steps) balanced GPU memory constraints (NVIDIA Tesla P100) with gradient stability, reducing training time compared to smaller batches
- We trained for 8 epochs, as validation macro F1 improved steadily

- Regularized weights to handle the augmented dataset's diversity, preventing overfitting.

Training ran on Kaggle's GPU (NVIDIA Tesla P100) for around 1 hour and 15 minutes

Experiments and Results

Initial Trials

Our baseline DeBERTa-base model (no CRF, no augmentation) achieved a validation macro F1 of 0.62 and weighted F1 of 0.94 after 3 epochs. Class imbalance caused poor recall for 'B-art', 'B-eve', and 'B-nat', lowering macro F1. Adding the CRF layer increased macro F1 to 0.65, as it reduced invalid tag sequences. Without CRF, models like RoBERTa (Table 1) plateaued at 0.5973 macro F1, confirming CRF's necessity

Augmentation Impact

Augmenting rare classes added 4,000 examples, raising macro F1 to 0.65 and improving recall for 'B-art' and 'B-eve'. Over-augmentation (e.g., 2x) introduced noise, slightly 3 lowering weighted F1 (0.01 drop), so we limited augmentation to rare class sentences. This was chosen after testing back-translation, which disrupted tag alignment due to length changes.

CRF Integration

Adding the CRF layer boosted the macro F1 to 0.69. It ensured valid tag transitions, reducing errors in multi-token entities. We tested learning rates (5e-5, 2e-5, 1e-5), finding 1e-5 best for stability.

Hyperparameter Tuning

We tested learning rates (1e-5, 2e-5, 5e-5) and batch sizes (8, 16). The learning rate of 1e-5 with an effective batch size of 32 provided stable convergence, achieving a validation macro F1 of 0.5973 and weighted F1 of 0.9412 at epoch 4. Higher learning rates (5e-5) reduced macro F1 (0.58) due to overfitting on frequent classes. Smaller batch sizes (8) increased training time without significant gains.

Final Results

Epoch	Training Loss	Validation Loss	Weighted F1	Macro F1
1	0.114200	0.108080	0.966920	0.543634
2	0.095700	0.098866	0.969188	0.605110
3	0.085100	0.094665	0.970271	0.625935
4	0.073600	0.092817	0.970822	0.655846
5	0.066000	0.093060	0.971861	0.678463
6	0.062000	0.094144	0.971812	0.686928
7	0.052200	0.096590	0.972354	0.705032
8	0.051900	0.100226	0.972083	0.702111

Our final model scored ~0.858 weighted F1 on the public leaderboard and was competitive on the private leaderboard (exact score pending). The validation macro F1 was 0.70. Training logs showed steady improvement, with the best model saved based on macro F1.

Insights and Adjustments

Key Insights

- The CRF layer was critical for modeling tag dependencies, reducing errors in rare class sequences (e.g., ‘B-art‘ → ‘I-art‘). Without CRF, models produced invalid transitions, lowering macro F1 by 0.05. This was evident in BERTbase and RoBERTa trials, which converged early with lower macro F1.
- Augmenting only rare class sentences balanced the dataset without introducing excessive noise. Back-translation was rejected due to length mismatches, while synonym augmentation preserved tag alignment, boosting macro F1 by 0.03.
- A low learning rate (1e-5) and moderate batch size (32) prevented overfitting on frequent classes, allowing sustained macro F1 improvement. Higher learning rates caused instability, as seen in early trials (macro F1 drop to 0.58).

Adjustments Made

- Limited augmentation to rare classes after seeing diminishing returns.
- Increased effective batch size with gradient accumulation for stability.
- Used macro F1 instead of weighted F1 for model selection to focus on rare classes.

Division of Labor

Work was split evenly (50% each):

- LAM, King Cheuk: Built the DebertaCRF model, set up training, analyzed results, and contributed to the report and debugging.
- YAU, Ho Yin: Handled data preprocessing, augmentation, submission scripts, hyperparameter tuning, and report writing.

Conclusion

Our project delivered a strong NER model using deberta-base with a CRF layer and targeted augmentation. It exceeded the course baseline, performing well on public and validation metrics. We learned the value of addressing class imbalance and modeling tag dependencies. Future work could explore ensemble models or advanced augmentation for better rare class performance.