# Introduction

In this assignment, we aimed at performing a sequence labeling task using annotated text data from the W-NUT dataset. The primary objectives include data preprocessing, hyperparameter optimization, training a sequence labeling model using Hugging Face Transformers, and evaluating the model using suitable evaluation metrics. We used of libraries such as transformers, seqeval, evaluate, and accelerate, which are essential for working with NLP models and datasets. We will discuss every step we took for implementation in the following section. Then we will showcase results from our baseline and optimized model and at last we will draw some conclusions.

# Methodology

**Data Preprocessing, dataset creation, and tokenizaion:**
The dataset consists of three IOB files: wnut17train.conll (train), emerging.dev.conll (dev), and emerging.test.annotated (test). Firstly, we converted the IOB data to the correct data structure for token classification in Hugging Face (words and labels like the conll2023 data in the tutorial).To implement this purpose, we read each line of every file, extracting every token and NER tag into a list object in a dictionary. Then we used $DatasetDict()$ and $Dataset.from\_dict()$ to change the format to the desired data structure. After creating the dataset object, we tokenized it and aligned the labels with the tokens. For tokenizing we used $AutoTokenizer.from_pretrained()$ and load a tokenizer from "bert-base-cased" model checkpoint. When we run tokenizer on our datasets the tokenizer adds the special tokens used by the model (e.g. [CLS] and [SEP]) so we need to align all the labels with the proper words now. We used $align\_labels\_with\_tokens()$ function from tutorial for this purpose.

**Statistic about data:**
To have a clear overview of the dataset's content and tag distribution in the train, validation, and test datasets. We print out some sample sentences to get a sense about the overall data. We also investigate which labels we have and for each dataset (training, test, and validation), our code counts the occurrences of different NER tags and displays the counts. We found the following statistcs for the count of each tag in each dataset:

| **Tag** | **Train** | **Test** | **Validation** |
|---|---|---|---|
| O | 59570 | 21654 | 14475 |
| B-corporation | 221 | 66 | 33 |
| I-corporation | 46 | 22 | 11 |
| B-creative-work | 140 | 142 | 105 |
| I-creative-work | 206 | 218 | 133 |
| B-group | 264 | 165 | 39 |
| I-group | 150 | 70 | 25 |
| B-location | 548 | 150 | 74 |
| I-location | 245 | 94 | 33 |
| B-person | 660 | 429 | 470 |
| I-person | 335 | 131 | 117 |
| B-product | 142 | 127 | 114 |
| I-product | 203 | 126 | 94 |

Table 1: Counts of Each Tag in Datasets

As we can observe from this table we have an imbalanced datasets. For example, we see that the "O" tag is

the most frequent, while some specific entity types like "B-corporation" have fewer occurrences.

**Challenges of the task and the data:**
1. Our dataset is imbalanced, with some entity types such as O Tages having significantly more examples than others. This can affect generalization of the model over different class of labels and the model can become biased.

2. The W-NUT focuses on noisy text which was found in social media it contains slangs (like "jk") and creative use of language(using "tonite" instead of "tonight"). This makes it harder to fine-tune a model, which was initially trained on more formal and well-structured text. The model may also encounter many Out-of-Vocabulary (OOV) Words.

3. Many words can have multiple meanings, and named entities may be mentioned in various contexts. Handling this kind of ambiguities can be challenging.

**Set up the evaluation:**
The evaluation was set up correctly for the W-NUT test set, following the tutorial. For this porpus we have to convert both labels and predictions from integers to strings. We remove all the values where the label is -100, then pass the results to the *metric.compute*() method exactly the same with tutorial. It returns overall precision, recall, f1-score, and accuracy.

**Fine-tune a baseline model:**
We chose "bert-base-cased" pretrained model to fine tune for this task. We used the default hyperparameter settings on *TrainingArguments*() and initialized a trainer with it. We fine-tuned trainer on train set and evaluated it on the test set. The results of the task according baseline are shown in the following table:

| Epoch | Training Loss | Validation Loss | Precision | Recall | F1 | Accuracy |
|---|---|---|---|---|---|---|
| 1 | No log | 0.265 | 0.516 | 0.301 | 0.380 | 0.942 |
| 2 | 0.174 | 0.302 | 0.608 | 0.303 | 0.404 | 0.945 |
| 3 | 0.057 | 0.322 | 0.566 | 0.333 | 0.420 | 0.946 |

Table 2: Train and Evaluation (on test dataset) Process with baseline

**Hyperparameter optimization:**
For hyperparameter optimization, we tried three different learning rates (1e-6, 1e-4, 5e-5) and two different values for batch size (8, 16). We try different combinations of these parameters manually using the TrainingArguments class. The best performing model (according f1) was trained with a learning rate of 1e-6 and a batch size of 8. This model achieved a F1-score of 56.40% on validation dataset.
**Evaluation of best model on the test set:** In this section we provide a table with results of the optimized model and our baseline model on test dataset. The results show that our optimized model is working a bit better than the baseline in terms of percision, recall, and f1-score.

| Model | Precision | Recall | F1 |
|---|---|---|---|
| Baseline | 0.566 | 0.333 | 0.420 |
| Optimized model | 0.580 | 0.349 | 0.436 |

Table 3: Result of evaluation of models on test dataset

**Extend the evaluation function:**
We extend the evaluation function so that it shows the Precision, Recall and F-score for each of the entity types (person, location, etc.) on the test set. For this purpose we rewrite the *compute_metrics*() function and used $classification\_report()$ from seqeval.metrics to calculate metrics per entity. The output of evaluation the best model on test dataset has provided in following table and shows that the scores are highest for the "person" entity type and lowest for the "product" entity type. This suggests that the model is able to identify person mentions better than product mentions.

| Category | Precision | Recall | F1 Score | Support |
|---|---|---|---|---|
| corporation | 0.351 | 0.287 | 0.316 | 66 |
| creative-work | 0.450 | 0.253 | 0.324 | 142 |
| group | 0.485 | 0.200 | 0.283 | 165 |
| location | 0.538 | 0.466 | 0.500 | 150 |
| person | 0.769 | 0.482 | 0.593 | 429 |
| product | 0.244 | 0.094 | 0.136 | 127 |

Table 4: Evaluation results for different entity types

**Micro and Macro F1 Scores:** The micro F1 score is calculated by averaging the F1 scores for all entity types, regardless of the number of instances of each entity type. In micro F1 we calculate metrics globally by counting the total true positives, false negatives and false positives of the model for all classes, and then using these sums to calculate F1 Score. Micro F1: 0.436
The macro F1 score is calculated by unweighted averaging the F1 scores for each entity type separately. Macro F1: 0.359
However, the macro F1 score may not be a good measure of overall model performance, as it does not take into account the prevalence of each entity type in the dataset.

# Conclusion

In summary, the hyperparameter optimization that we have done on our model showed that the model performs best with learning rate = 1e-06 and batch size = 8. Then the evaluation on test set showed that the optimized model has actually improved (in terms of precision, recall, and F1 score) compared to the baseline model when evaluating on test dataset.

Differences in scores for different entity types demonstrate how well the model handles specific categories. For example our model performance on some entity types (like product category) was lower than on others (such as person and location categories). This might happened because of the imbalanced datasets we used for training the model. If some entity types are more frequent in the training data than others, the model may be biased towards recognizing those entities better due to more examples to learn from. Less frequent entity types might not be learned as effectively, resulting in lower F1 scores.

The difference between macro- and micro-averaged F1 scores come from the impact of label weighting in evaluation process. Micro-averaging gives more weight to labels with a higher number of instances, when it aggregates across all labels. In contrast, macro-averaging treats all labels equally which results in an equal contribution for different labels in the final score. We observe a difference between the micro and macro f1 scores in our evaluation and that comes from our imbalanced test dataset. When the macro-averaged F1 score is significantly lower than the micro-averaged F1 score, it suggests that the model performs poorly for some low-support labels which is true in our case and we can observe it in tabel 4 (e.g. comparing f1 score of "person" and "coorporation").