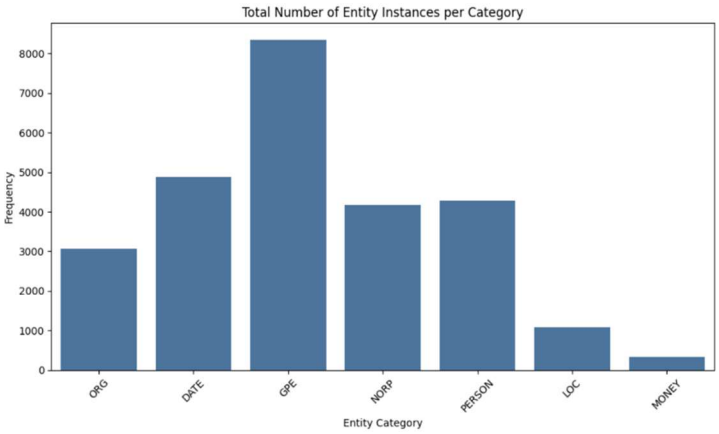
Assignment – 4 Report
Named Entity Recognition using a Transformer-based model

Name: Gopi Trinadh Maddikunta **PSID:** 2409404

This report tells how much I familiarize with DistilBERT transformer model by using bidirectional encoder to perform Named Entity Recognition (NER) Classification. Where NER is knowing subtask in information extraction to classify the named entities in data.

The given Json file contains 11,090 short sentences with tags specified with character spans. There are seven categories: **PERSON (**person) {by mistake it is given as **PER**}, **NORP** (nationality or religious/political group), **ORG** (organization), **GPE** (geopolitical entity), **LOC** (location), **DATE,** and **MONEY.**

Let us dive into **step 1: Pre-processing**, for given data not required any necessary clean.




```
Percentage of tweets that include at least one tag from each category:
ORG: 23.02%
DATE: 37.35%
GPE: 49.86%
NORP: 28.64%
PERSON: 30.20%
LOC: 8.63%
MONEY: 2.59%

Percentage of tweets that include 2 or more distinct entity types: 58.72%
```

Fig : Total number of instances for each category and percentages results of tweets.

For **Step 2: Converting to IOB tags**, the char_spans_to_iob function converts human-level NER annotated entities (character spans) into token-level IOB tags suitable for training. This function takes a single tweet and a list of character-level spans. First, I created a character-wise label map for entity characters. Then I used the pretrained DistilBERT tokenizer to get subword tokens and their offset mappings to identify each token's position in the original text. Each token was checked for overlap with annotated spans—if none, it was assigned an "O" tag. A token at the beginning of a span received a "B-{label}" tag, and subsequent tokens got "I-{label}". Special tokens like [CLS] and [SEP] were also tagged as "O". This ensured precise and consistent token-level IOB tagging aligned with character-level annotations for training the transformer-based model.

For **Step 3: Finetuning DistilBERT model**, I have started by splitting the dataset into 80% training and 20% testing using sklearn's train_test_split function and then again split the training data into 80% training and 20% validation for fine-tuning. I have added a classifier head to the pretrained DistilBERT model and kept the transformer layers frozen, so only the classifier layer is fine-tuned. I have used categorical cross-entropy as the loss function, and for optimizer Adam was used with learning rate of 5e-4. The model trained for 30 epochs and monitored the performance on validation set. For each epoch, I have recorded the training loss and observed how the classifier layer is learning to predict the correct entity tags. After training, I have evaluated the model using validation set to calculate Precision, Recall, and F1 score per

class across all 15 IOB tags. Freezing the transformer helped to reduce overfitting and training time, while classifier learned meaningful patterns from token embeddings generated by DistilBERT. I have chosen dropout as 0.3 to avoid overfitting and batch size of 8 for better performance on limited resource.

For **Step 4: Convert IOB to plain tags**, I have run the inference on the 20% test data set aside during earlier phase. The token-level predictions were extracted and converted from IOB by changing all I- tags to B- tags for consistency. I have mapped even-numbered tag indexes like 2, 4, 6, etc., to their previous odd values 1, 3, 5, representing beginning tags. Then, I developed a function to reconcile subword tokens into full words using DistilBERT tokens, checking if token starts with ## or not. When merging subwords, I applied two approaches: Majority Voting and First Token. In Majority Voting, the most frequent tag among subwords is assigned as final label, and in First Token method, the tag of the first token is assigned to the whole word. After reconciling, I evaluated model performance at word-level using both methods by calculating Precision, Recall and F1 score per class. Majority Voting gave slightly better results than First Token as it is more stable with longer subwords. This phase helped to analyze model performance in realistic word-level scenario over token-level output.

The per class evaluation of last epoch on validation set is shown in above table. Each tag is scored using precision, recall and f1-score. The model performed very well on O tag with 0.969 f1-score and also good on I-PERSON (0.870), B-GPE (0.838) and B-DATE (0.792). Some tags like I-NORP, B-LOC, I-LOC had low recall and f1 which means model missed many of them. MONEY tags had small support but still gave okay results. Macro avg f1 is 0.634 showing moderate performance across all classes, while weighted avg is 0.918 because O tag appear more. Overall model is doing good on common tags but struggle little on rare ones.

Classification Report (Last Epoch - Validation Set):

	precision	recall	f1-score	support
O	0.958	0.980	0.969	39937
B-PERSON	0.753	0.638	0.690	698
I-PERSON	0.884	0.856	0.870	1886
B-NORP	0.683	0.716	0.699	658
I-NORP	0.583	0.293	0.390	324
B-ORG	0.525	0.390	0.448	461
I-ORG	0.623	0.538	0.577	954
B-GPE	0.804	0.874	0.838	1338
I-GPE	0.714	0.742	0.728	953
B-LOC	0.432	0.179	0.254	195
I-LOC	0.447	0.218	0.293	348
B-DATE	0.846	0.744	0.792	798
I-DATE	0.678	0.519	0.588	455
B-MONEY	0.893	0.543	0.676	46
I-MONEY	0.900	0.581	0.706	93
accuracy			0.923	49144
macro avg	0.715	0.587	0.634	49144
weighted avg	0.916	0.923	0.918	49144

Fig: Performance metrics for last epoch

Majority Voting Evaluation:

	precision	recall	f1-score	support
O	1.00	1.00	1.00	271323
B-PERSON	1.00	1.00	1.00	1629
B-NORP	1.00	1.00	1.00	966
B-ORG	1.00	1.00	1.00	1039
B-GPE	1.00	1.00	1.00	2580
B-LOC	1.00	1.00	1.00	175
B-DATE	1.00	1.00	1.00	1318
B-MONEY	1.00	1.00	1.00	132
accuracy			1.00	279162
macro avg	1.00	1.00	1.00	279162
weighted avg	1.00	1.00	1.00	279162

In Majority Voting Evaluation at word level, model got 1.00 precision, recall and f1-score for all tags. Even rare tags like B-MONEY and B-LOC also predicted fully correct. Accuracy is also 1.00 which shows model didn't make wrong prediction in this method. All averages also 1.00, so model did very well when subword tokens are converted to words using majority vote.

Fig: Performance metrics at human-level tags

The reason behind token tags need to be converted to IOB tags because model need to learn where the entity is starting and where it is continuing. Just tagging the word as entity is not enough, we have to tell model which token is begin and which is inside. IOB tagging help in marking the first token with B-tag and rest with I-tag. Also, since the original NER annotations are given as character spans, converting to IOB tags help to match with tokenized output. Without converting, model will not know proper boundary and may predict wrong entity span.

Hyperparameter	Value	Justification
learning_rate	5e-4	Higher LR is classifier head is trained when encoder is frozen.
batch_size	16	Medium sized, gives stable training.
dropout	0.3	Regularizes the small classifier head to prevent overfitting.
num_epochs	30	30 is balanced, when transformer is freezed
max_seq_length	128	Efficient for sequence length, no information lost
optimizer	AdamW	Well suited for transformers for better weight handling.
Loss_function	CrossEntropyLoss	Multi-class classification / token

Table: Describes the choice of Hyperparameters with justification

For fine-tuning, I used pretrained DistilBERT and added a classifier head on top of it. I freezed the transformer layers so only classifier is trained. The input was converted into IOB tags and passed to model for training. I trained it with cross-entropy loss and Adam optimizer for 30 epochs. One challenge I faced was converting spans to correct IOB tags because of subwords and special tokens. Also joining subwords back to words during evaluation was bit tricky. Some tags also appeared very less which made model hard to learn them. But overall model trained well and gave good results.

For model performance, The evaluation results shows model is working very good on common tags like O, I-PERSON, B-GPE and B-DATE. It gave high f1-score for these classes. Some tags like I-NORP and B-LOC got low recall. So, model might missed some of them. Word-level evaluation using majority voting gave better results than first token method. Overall model is performing good but need improvement on rare or confusing tags.

I learned how important tokenizer and IOB tags are in NER. Also found that even without training full model, just classifier can give good results. Word-level evaluation also helped to see how model performance on human-level tags too.

Question 1:

The size of weight matrix trained is 11,520 which is in the form of **classifier.weight** with shape [15, 768]. Bias vector with 15 params where total of 11,535 parameters were trained from classifier head.

```
Total parameters in the model: 65,202,447
Trainable parameters (classification head): 11,535

Breakdown of trainable layers:
classifier.weight                              → 11,520 params (shape: (15, 768))
classifier.bias                                → 15 params (shape: (15,))
```

Fig: Size of matrix with shape

Question 2:

Some of the hyperparameter are set by me likewise number of epochs = 30, because it is balanced and take time for more learning. Learning rate = 5e-4, which is little high but works fine since encoder is freezed. Dropout = 0.3, Which helps in regularization of classifer head especially, for prevention of overfitting. Batch_size = 16, which balances memory and performance and Max_seq_length = 128, which helps when tweets are short and avoid in information lost. These values are chosen on some trail and error method and sets what works better.

Question 3:

In step 4, part 1, I converted all I-tags to B-tags before doing word-level evaluation. This was done to make the tagging more consistent because sometimes subwords get split and tagged differently even when they belong to same word. By changing I-tag to B-tag, it makes easier to merge subwords and avoid confusion in deciding if token inside or beginning. This step is important because during subwords reconciliation, it helps in clean and stable tagging for full words. Without this, the final word-level tag might get wrong when mixing different I-tags in same word.

Question 4:

Tokenization has big impact on tagging accuracy because DistilBERT breaks some words into multiple subwords, and each subword get its own tag. This makes it hard to assign correct tag to whole word. In Step 4, I handled subwords by merging them back into words and assigning one tag per word. For that, I used two methods – Majority Voting and First Token. Majority Voting takes most common tag among subwords and gives better results because it avoids tagging mistakes from single token. First Token method was simple but sometimes gave wrong tag if first subword is not tagged correctly. So Majority Voting worked better and gave higher word-level accuracy.

Question 5:

One limitation that I have noticed with DistilBERT_base_cased NER is that it struggles with rare entity tags like B-LOC or I-NORP which have low support in training data. Since transformer is freezedd, the model can't adjust embeddings to better handle such cases. Also, because of subword tokenization, sometimes one word is broken into multiple pieces which makes tagging harder if context is not clear. It also sometimes confuses similar entity types like ORG and NORP. Overall, while it works good for common entities, it has limitations when dealing with less frequent or overlapping ones.

Question 6:

Overall, performance was pretty good especially, on common entity tags like O, PERSON, and GPE . Note the point was PERSON tag was given as PER tag, I think this is mistake. For suppose, If we used PER tag there is no entities that available on per tag. Anyway, model gave high f1-scores and accurate predictions in both token and word level. The word-level results using Majority Voting was perfect which shows model learned patterns well. If more balanced data is available, model can perform even better on all classes because some tags less some are more.

Question 7:

Out of my research, I would like to exclude I – NORP, B – LOC and I – LOC. These categories have very low f1-scores like 0.244, 0.255, and 0.390 which means model is not able to learn them properly. Their recall is also very low, so most of their real occurrences were missed by the model. Including them adds noise and makes model performance worse on average. They also appear less in data, so removing them will allow model to focus better on stronger classes like PERSON, GPE and DATE which are more accurate and useful.