Tiago Cumetti (s3897907)
Van Nguyen (s3726266)

Assignment 2: Sequence labelling

Text mining

November 8, 2023

1 Introduction

The goal of this assignment is to work with W-NUT dataset for "emerging and rare entity recognition".
With the help of Hugging face platform and libraries, such as Transformers for NLP, we are supposed to pre-process the provided data and train properly a Named Entity Recognition classifier on them, performing a fine-tuning through hyperparameter optimization to apply and evaluate correctly the sequence labelling.

The datasets we are working with are from the WNUT 17 Emerging Entities task. A task that "focuses on identifying unusual, previously-unseen entities in the context of emerging discussions." In the context of this assignment we are working with 3 datasets. The first one (3394 rows) is the training set called *wnut*17*train.conll* which is sourced from formerly know platform called Twitter (now X). The second one (1009 rows) is the validation (dev) set called *emerging.dev.conll* which contains data taken from presumably YouTube comments and lastly the test set (1287 rows) with tags named *emerging.test.annotated* from either news outlet or some reddit-subreddits.
The entity types present in the datasets are 'person', 'location', 'corporation', 'group' (subsuming music band, sports team, and non-corporate organisations), 'creative-work' (song, movie, book, and so on) and 'product' (tangible goods, or well-defined services).

1.1 Challenges of this task

One of the main challenges of this task is detecting novel and emerging entities. Meaning the ability to detect and classify previously-unseen entities in noisy text. The example given in source web page of this task can be really explanatory, it is about the word "kktny" in a context like "so.. kktny in 30 mins?", where *kktny* entity is not well-defined and can be hard to identify even as a human being. (kktny stands for Kourtney & Kim take New York). Defining emerging and rare entities can be difficult, especially in this task case where the data belong to the same domain of social networks, but each set of them comes from a different platform (Twitter, Youtube and Reddit), with a different public and different way of expressing concepts.

Another challenge coming from the original data representation is the different way in which labels are expressed, the structure of the data needs to be properly correct to fit with the Hugginface model that we want to train and to do that we need to redefine the original labels, to map them to numerical values and to create an appropriate data collator to reverse this process at the right moment in the implementation of the model.

2 Baseline model and fine-tuning through the hyperparameters

In this section we will talk about the process of tuning the hyperparameters, its prerequisites and its results.

First we imported the training set, validation set (dev) and testing set. Then we set up a correct data structure, transforming the original labels in numerical labels, tokenizing the input data and aligning labels with tokens. After that we also create a proper data collator and we finally set up the correct evaluation process to asses the results. At this point we imported and trained our classification model from Hugginface, combining all the previous actions to let our model fit in the best way on our data structure and retrieving the baseline results.

For the actual tuning of the hyperparameters the first thought was to simply run grid search to get the results but for some reason we kept running out of memory so we opted to manually testing each possible combination. The baseline model has learning rate of 2e-5 and batch size of 8. We decided to search through combination of learning rates of (1e-4, 5e-5, 2e-5 and 1e-5) with batch sizes of either 4 or 12. All the hyperparameter combinations are shown in Table 1.

Table 1: Hyperparameter Combinations and Evaluation Metrics on Test Set

Model	**Learning Rate**	**Batch Size**	**Precision**	**Recall**	**F1 Score**
Baseline	2e-5	8	0.5648	0.1878	0.2819
arg_com1	1e-4	4	**0.5814**	0.1895	0.2858
arg_com2	1e-4	12	0.5558	**0.2106**	**0.3055**
arg_com3	5e-5	4	0.5706	0.1944	0.2900
arg_com4	5e-5	12	0.5412	0.2098	0.3024
arg_com5	2e-5	4	0.5289	0.1973	0.2874
arg_com6	2e-5	12	0.5827	0.1783	0.2730
arg_com7	1e-5	4	0.5740	0.1687	0.2608
arg_com8	1e-5	12	0.5787	0.1828	0.2779

From the results displayed in Table 1 we can see that the highest achieved precision was 0.5814, in correspondence of **arg_com1** where the learning rate was 1e-4 and batch size of 4. While the highest recall and F1 Score was achieved with **arg_com2** where the learning rate was also 1e-4, but with a batch size of 12. At this point we can consider the learning rate 1e-4 as the most suitable.

2.1 Extended evaluation metrics

In this section we extended basic function *compute_metrics* to a new function *compute_metrics_extended.*
This new function calculates micro avg, macro avg and weighted avg on top of before calculated metrics. We chose to perform this extra evaluations just with the models that use **arg_com1** and **arg_com2**, that result the most suitable settings for our model from the previous experiment, but we choose to show only **arg_comb2** as it has achieved the best results in the end.

Table 2: Full entity values

Label	**Precision**	**Recall**	**F1-Score**	**Support**
Corporation	**0.646**	**0.287**	**0.398**	571
Creative Work	0.529	0.116	0.191	232
Group	0.259	0.160	0.198	231
Location	0.386	0.340	0.362	150
Person	0.469	0.197	0.278	233
Product	0.255	0.094	0.138	127
Micro Avg	0.465	0.218	0.297	1544
Macro Avg	0.424	0.199	0.261	1544
Weighted Avg	0.486	0.218	0.294	1544

Table 3: Evaluation Metrics for B- and I- labels separately

Entity Type	Precision (B)	Recall (B)	F-score (B)	Precision (I)	Recall (I)	F-score (I)
O	0.760	0.340	0.480	-	-	-
Location	**0.570**	0.110	0.180	0.320	0.150	0.210
Group	0.530	**0.410**	**0.460**	0.230	0.050	0.080
Corporation	0.240	0.110	0.160	**0.570**	0.090	0.160
Person	0.310	0.060	0.100	0.200	0.100	0.140
Creative-Work	0.440	0.220	0.300	0.640	**0.220**	**0.330**
Product	0.280	0.070	0.110	-	-	-
Micro Avg	0.920	0.920	0.920	-	-	-
Macro Avg	0.470	0.230	0.280	-	-	-

In Table 2 we can notice the higher values are in label "Corporation" as it occurs the most, looking at the given support values. For location in Table 3 we can notice label "Location" having a relatively high precision(B) which means the model predicts label succesfully. But the recall is low meaning that it misses actual location entities quite often. Whereas label "Group" has comparable B- precision and recall meaning it identifies labels accurately and doesn't miss either. The "Corporation" label has high I- precision and low recall meaning that it's being conservative with its prediction.

3 Discussion

From the beginning of the tuning hyperparemeters implementation we can see that this procedure does little to improve the quality of the model as opposed to the based line model. Applying the hyperparameter optimization to the model increases the quality of it at most of 0.0228 for precision, recall and F1 score.

The scores for precision, recall and F1 can indicate few important things.

1. Precision for the entity is high to indicate that the model can successfully predict the type of this entity class. Also higher precision indicates the fewer false positives for given entity.

2. Recall measures correctly identify instances of a specific type. High recall indicates how many specific instances it identifies in its entirety. The higher the recall is the less likely the model is to miss instances of that entity type.

3. F1 is harmonic mean of above mentioned values and provides what the balance is between those two. The higher the value is the best the model is performing and the more the ratio between precision and recall is balanced.

Higher precision and low recall means the model makes correct prediction but misses the entity itself. On the other hand high recall and low precision means that the model doesn't miss the entities often but wrongly labels them. When these two measures are balanced the model correctly identifies labels and doesn't miss.

Looking at the results of Table 1 we can see how the model achieves better results for precision than for recall and after the hyperparameter optimization for each class label this difference between the two measures remains, slightly increasing for some label classes and slightly decreasing for some others, as shown in Table 2 & 3. From these tables we can also see how our model performs better on some label classes and worse on some others. On corportation and location classes the model perform well in all the measures, while for Group and Product classes it achieves bad results.

Micro and macro average F1 score differentiate in their calculation methods and in their different usage. Micro is calculated by taking the overall performance of all labels as one label. Aggregating in true positives, false positives and false negatives to calculate this value. Micro average is good to use for imbalanced datasets where there are more entities over the other ones. Macro average is calculated by taking different labels and considering the evaluating metrics separately and then computing the average of these metrics, as in the case of the F1 values.

Basically the micro average emphasizes the dominant classes and macro takes everyone equally. The higher the micro average is the better overall model performance if there are dominant classes present, but we aren't sure that the model performs well for low-represented entity classes. Higher macro average indicates the better performance if the classes are evenly distributed and it's also a proof that the model performs well on every type of entity class.

In our case we would have evenly distributed classes but the entity type "Corporation" overshadows other entities as shown in "Support" label. Looking again at Tables 2 & 3, the micro average values result always higher than the macro average ones, this can be explained looking at the evaluation metrics for the single entity types where we can see that the model performs better on Corporation, that's the main represented class, and obtains worse results on less represented class entities such as Product.