Munier

Antoine

Homework 3

Introduction:

The goal of this exercise is to create Named Entity Recognition (NER) from the BERT model. To do this, we'll use the DNRTI dataset, which is divided into three parts: train, test, and valid.

Model :

In an initial version, I started with a classic BERT model. To improve performance, I first added a BiLSTM layer, as suggested in the brief, which allowed for better processing of sequential context. Subsequently, I replaced BERT with SecBERT, a model pretrained specifically on cybersecurity-related data, better suited to our DNRTI dataset. This change immediately led to a significant improvement in results, particularly for certain business entities. However, despite these adjustments, rare classes were still poorly recognized compared to the results obtained by the professor. To address this, I decided to introduce a weighted loss function based on class frequency to give more weight to minority entities. This strategy significantly improved the recall and F1-score of these underrepresented classes.

The SecBERT model comes from Huggingface: https://huggingface.co/jackaduma/SecBERT

*I took the first one found on the site

Results :

Classic BERT :

	precision	recall	f1-score	support
Area	0.82	0.90	0.85	251
Exp	0.99	1.00	0.99	691
Features	0.74	0.92	0.82	152
HackOrg	0.78	0.88	0.83	628
Idus	0.82	0.79	0.80	142
OffAct	0.79	0.85	0.81	208
Org	0.58	0.71	0.64	153
Purp	0.55	0.79	0.65	140
SamFile	0.87	0.93	0.90	811
SecTeam	0.88	0.93	0.90	339
Time	0.75	0.90	0.82	186
Tool	0.75	0.79	0.77	725
Way	0.83	0.92	0.87	214
micro avg	0.81	0.89	0.85	4640
macro avg	0.78	0.87	0.82	4640
weighted avg	0.82	0.89	0.85	4640

BERT + BiLSTM :

	precision	recall	f1-score	support
Area	0.85	0.86	0.85	252
Exp	0.99	1.00	0.99	476
Features	0.93	0.99	0.96	118
HackOrg	0.72	0.81	0.76	477
Idus	0.88	0.87	0.88	142
OffAct	0.78	0.70	0.74	179
Org	0.66	0.63	0.65	149
Purp	0.76	0.85	0.80	123
SamFile	0.89	0.90	0.90	593
SecTeam	0.86	0.83	0.85	179
Time	0.79	0.83	0.81	186
Tool	0.71	0.74	0.73	465
Way	0.88	0.97	0.92	164
micro avg	0.83	0.85	0.84	3503
macro avg	0.82	0.84	0.83	3503
weighted avg	0.83	0.85	0.84	3503

SecBERT + BILSTM :

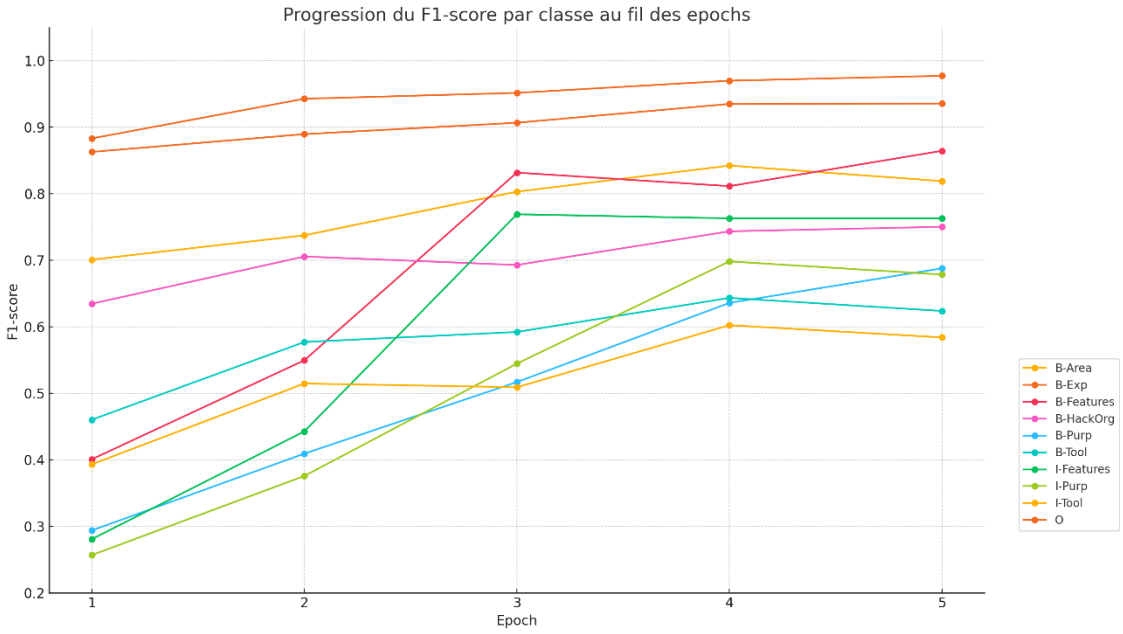
	precision	recall	f1-score	support
Area	0.84	0.82	0.83	252
Exp	0.99	1.00	0.99	476
Features	0.81	0.98	0.89	118
HackOrg	0.73	0.81	0.77	477
Idus	0.82	0.89	0.85	142
OffAct	0.78	0.64	0.70	179
Org	0.61	0.63	0.62	149
Purp	0.70	0.83	0.76	123
SamFile	0.93	0.87	0.90	593
SecTeam	0.85	0.84	0.84	179
Time	0.88	0.89	0.88	186
Tool	0.77	0.68	0.72	465
Way	0.91	0.97	0.94	164
micro avg	0.84	0.83	0.84	3503
macro avg	0.82	0.83	0.82	3503
weighted avg	0.84	0.83	0.83	3503

SecBERT + BiLSTM + Weighting of the loss function:

	precision	recall	f1-score	support
B-Area	0.7196	0.9506	0.8191	162
B-Exp	0.9652	0.9898	0.9774	589
B-Features	0.7615	1.0000	0.8646	99
B-HackOrg	0.6523	0.8831	0.7504	599
B-Idus	0.8413	0.9138	0.8760	116
B-OffAct	0.4498	0.7833	0.5714	120
B-Org	0.5974	0.7077	0.6479	130
B-Purp	0.5276	0.9885	0.6880	87
B-SamFile	0.7235	0.8766	0.7927	397
B-SecTeam	0.7331	0.7936	0.7621	218
B-Time	0.8097	0.8883	0.8472	206
B-Tool	0.5788	0.6767	0.6239	467
B-Way	0.6566	0.8790	0.7517	124
I-Area	0.5833	0.9800	0.7313	50
I-Exp	0.6618	1.0000	0.7965	45
I-Features	0.6169	1.0000	0.7631	95
I-HackOrg	0.4409	0.7987	0.5681	154
I-Idus	0.6731	0.9459	0.7865	37
I-OffAct	0.4206	0.6618	0.5143	68
I-Org	0.5321	0.8056	0.6409	72
I-Purp	0.5135	1.0000	0.6786	133
I-SamFile	0.7739	0.9674	0.8599	92
I-SecTeam	0.6348	0.9733	0.7684	75
I-Time	0.6585	0.9205	0.7678	88
I-Tool	0.4864	0.7310	0.5841	171
I-Way	0.6621	0.8889	0.7589	108
O	0.9846	0.8913	0.9356	15110
accuracy			0.8865	19612
macro avg	0.6540	0.8850	0.7454	19612
weighted avg	0.9150	0.8865	0.8951	19612

* I cheated a little because there are more Epochs for the latest models (from 3 to 5).

Model progression at each Epoch:



We can see directly that small classes like B-Features, I-Purp or I-Features improve quickly thanks to the weighting of learning. For large classes they reach their maximum very quickly and stagnate (they don't need more Epochs). There are low average classes that progress well like B-Purp, I-Tool and I-Purp.

Conclusion:

By gradually improving our model from a simple BERT to a SecBERT + BiLSTM + Weighting by class, we significantly improved the results and were able to see what each element brought. On the majority classes, the model's F1 scores are close to 90%, which shows that our model learns very well. For the minority classes, it is mainly thanks to the deweighting that we were able to achieve good results (around 70%).

Comparison of my results with that of the professor:

Entity	My F1-score	Teacher F1-score	Gap
B-HackOrg	0,75	0,79	0,04
I-HackOrg	0,57	0,7	0,13
B-OffAct	0,57	0,77	0,2
I-OffAct	0,51	0,8	0,29
B-SamFile	0,79	0,92	0,13
I-SamFile	0,86	0,92	0,06
B-SecTeam	0,76	0,88	0,12
I-SecTeam	0,77	0,78	0,01
B-Time	0,85	0,93	0,08
I-Time	0,77	0,93	0,16
B-Tool	0,62	0,75	0,13
I-Tool	0,58	0,76	0,18
B-Idus	0,88	0,92	0,04
I-Idus	0,79	0,87	0,08
B-Org	0,65	0,6	-0,05
I-Org	0,64	0,75	0,11
B-Area	0,82	0,88	0,06
I-Area	0,73	0,75	0,02
B-Purp	0,69	0,97	0,28
I-Purp	0,68	0,96	0,28
B-Exp	0,98	0,99	0,01
I-Exp	0,8	0,9	0,1
B-Features	0,86	1	0,14
I-Features	0,76	0,98	0,22

I only beat the teacher's model (B-Org) once. Otherwise, it's better than mine everywhere, and sometimes by a large margin. The CRF seems to be the cause. It allows our models to learn links between tokens. For example, some tokens never follow each other; this information is very useful for a classification exercise. The entire sequence is taken into account, not just the token itself. It's like a language teacher correcting the classification if the sequence doesn't make grammatical sense. This significantly improves results for weak classes because here the BiLSTM doesn't have time to learn that. (The BiLSTM doesn't know "the grammatical rules" but learns the context linked to each token, which also allows for better classification).