NAMED ENTITY RECOGNITION

Università degli Studi di Pisa, Master Degree in Computer Science
Course of “Human Language Technologies”
Prof. Giuseppe Attardi

Report of the final Project

A.A. 2022/2023

GAETANO NICASSIO
Matricola 658073

Contents

1. Introduction

Named entity recognition (NER) is a form of natural language processing (NLP), which is a subfield of artificial intelligence. NER is the task of detecting key information in texts, a specific entity can be only a word or a group of words that refer to the same concept. A named entity is, roughly speaking, anything that can be referred to with a proper name: a person, a location, an organization. The task of named entity recognition (NER) is to find spans of text that constitute proper names and tag the type of the entity. Four entity tags are most common: PER (person), LOC (location), ORG (organization), or GPE (geo-political entity). However, the term named entity is commonly extended to include things that aren't entities per se, including dates, times, and other kinds of temporal expressions, and even numerical expressions like prices. Here's an example of the output of an NER tagger [1]:

Citing high fuel prices, $[_{ORG}$ United Airlines] said $[_{TIME}$ Friday] it has increased fares by $[_{MONEY}$ $6] per round trip on flights to some cities also served by lower-cost carriers. $[_{ORG}$ American Airlines], a unit of $[_{ORG}$ AMR Corp.], immediately matched the move, spokesman $[_{PER}$ Tim Wagner] said. $[_{ORG}$ United], a unit of $[_{ORG}$ UAL Corp.], said the increase took effect $[_{TIME}$ Thursday] and applies to most routes where it competes against discount carriers, such as $[_{LOC}$ Chicago] to $[_{LOC}$ Dallas] and $[_{LOC}$ Denver] to $[_{LOC}$ San Francisco].

This task can be adopted in a plenty of fields (Human resources, Customer support, Content classification, Healthcare and so forth). Named entity tagging is a useful first step in lots of natural language processing tasks. In sentiment analysis we might want to know a consumer's sentiment toward a particular entity. Entities are a useful first stage in question answering, or for linking text to information in structured knowledge sources like Wikipedia. And named entity tagging is also central to tasks involving building semantic representations, like extracting events and the relationship between participants.

As final project for Human Language Technologies (HLT) course, I developed a project that extracts knowledge from the MultiCoNER II competition [5]. I also compared the quality of the project's result with the results of MultiCoNER competition, available at this url: https://multiconer.github.io/results

Related Work

In the early state of NLP several tasks like NER are solved by Symbolic Artificial Intelligence (SAI). It uses human-readable symbols that represent real-world entities or concepts in order to create "rules" for the manipulation of those symbols, leading to a rule-based system. As AI has progressed, we have moved on techniques such as Conditional Random Fields (CRF) or Support Vector Machines (SVM) models provided by Machine Learning field.

The gold standard benchmark for NER was laid out in a 2003 academic challenge called CoNLL. The CoNLL data set consists of news articles with all the named entities hand-labeled by humans. (There is also a German-language CoNLL data set.) This established the four standard NER classes: person (PER), organization (ORG), location (LOC), and miscellaneous (MISC).

The state of the art of NER tasks are tracked [10] and the results of this task are now improved compered to 2003.

In these last years, a new Machine Learning framework has earned the supremacy in the NLP: it is called **BERT** (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers). BERT is a model which is composed of several transformer layers with a multi-Head attention module that allows to understand better (respect to previous models) the context of the sentences. BERT is a general pre-trained framework, so this allows us to use it as a baseline and to extend it in order to solve a specific task like NER.

2. The Model Architecture

The model architecture used in the NER Task for this project, is a three-layer architecture composed by:

- A transformer layer (a Bert Model)
- A feed forward linear layer and a SoftMax layer to compute the scores of the tags on top the transformer.
- A Conditional Random Field as final layer that outputs the best path

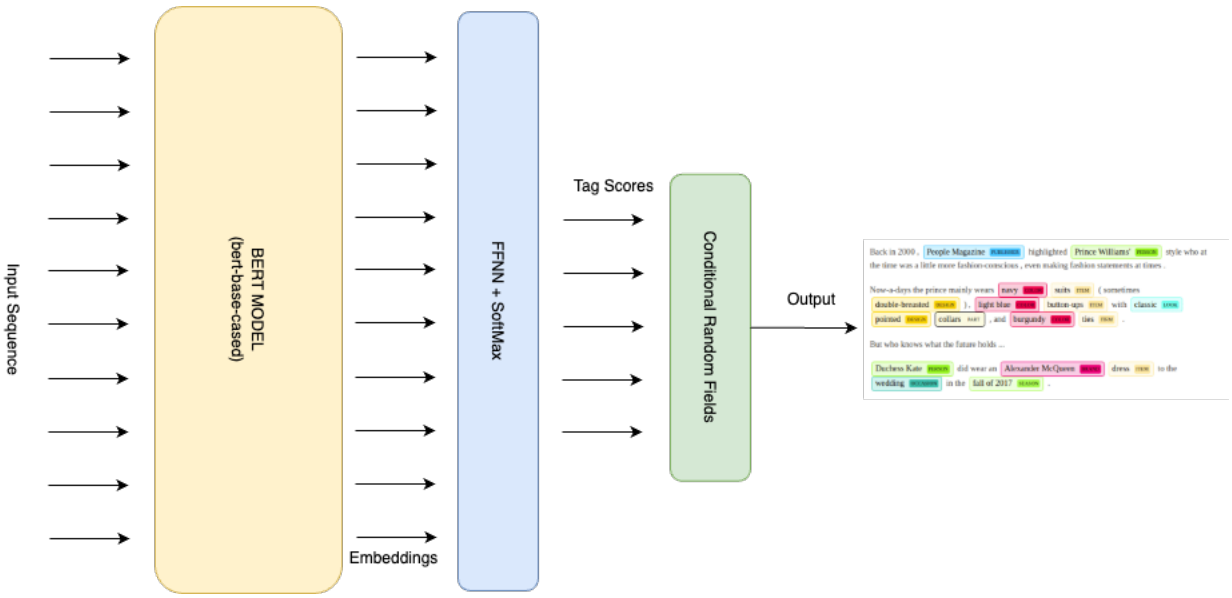


Figure 1 - Model Architecture

The parameters and the search procedure used for the model training and testing, are described in the paragraph [4]. In the following subsection I will explain a detailed description of the specific layer of the model.

Bert Model

BERT is a transformers model pretrained on a large corpus of English data in a self-supervised fashion. This means it was pretrained on the raw texts only, with no humans labelling them in any way (which is why it can use lots of publicly available data) with an automatic process to generate inputs and labels from those texts. More precisely, it was pretrained with two objectives:

- **Masked language modeling (MLM)**: taking a sentence, the model randomly masks 15% of the words in the input then run the entire masked sentence through the model and must predict the masked words. This is different from traditional recurrent neural networks (RNNs) that usually see the words one after the other, or from autoregressive models like GPT which internally mask the future tokens. It allows the model to learn a bidirectional representation of the sentence.
- **Next sentence prediction (NSP)**: the model concatenates two masked sentences as inputs during pretraining. Sometimes they correspond to sentences that were next to each other in the original text, sometimes not. The model then must predict if the two sentences were following each other or not.

This way, the model learns an inner representation of the English language that can then be used to extract features useful for downstream tasks.

For the aim of this project, the Bert model used is the *bert-based-cased* from Hugginface Hub libraries [3], a Pretrained model on English language using a masked language modeling (MLM) objective. It was introduced in this paper and first released in this repository. This model is case-sensitive: it makes a difference between "English" and "english".

Linear Layer

On top of the Bert model a layer of a neural network has been added with an input hidden size equal to the number of embeddings in output to the Bert Model, whose values are sent to a SoftMax layer which calculates the score of each tag.

Conditional Random Field

It's a discriminative sequence model based on log-linear models.
Assuming we have a sequence of input words $X = x1...xn$ and want to compute a sequence of output tags $Y = y1...yn$. In a Hidden Markov Model [1] to compute the best tag sequence that maximizes $P(Y|X)$ we rely on Bayes' rule and the likelihood $P(X|Y)$:

$$\begin{aligned}\hat{Y} &= \operatorname*{argmax}_Y p(Y|X) \\ &= \operatorname*{argmax}_Y p(X|Y)p(Y) \\ &= \operatorname*{argmax}_Y \prod_i p(x_i|y_i) \prod_i p(y_i|y_{i-1})\end{aligned}$$

In a CRF, by contrast, we compute the posterior $p(Y|X)$ directly, training the CRF linear chain CRF to discriminate among the possible tag sequences:

$$\hat{Y} = \operatorname*{argmax}_{Y \in \mathcal{Y}} P(Y|X)$$

For the aim of this project, I used the **linear chain CRF**, the version of the CRF most used for language processing.
CRFs find their applications in named entity recognition, part of speech tagging, gene prediction, noise reduction and object detection problems, and many other applications.

3. Corpus

The corpus used to train and test the model, is the one provided by MultiCoNER Competition [5]. It is composed of three different datasets, train, dev (validation) and test, each used for the respective model execution phase. The entire dataset is formatted by CoNLL [2], which means that the file contains one line for each token in the corpus with additional information such as IOS Tagging

Additional information such as POS tag, labels or general annotations are separated by a "_" character, instead different sentences are separated by a new line. For the experiment, I need a label for each token that identify if the token is associated to a specific tag (Person, Location, Facilities, and so on). Sometimes one tag could be associated to more than one single word, for example: "Mario Rossi is a researcher", "Mario Rossi" are two tokens but together the tag is "Person". To solve this, the annotations are coded by IOB format [11] (Inside–outside–beginning tagging), where the I- prefix before a tag (e.g. "I-PER") indicates that the tag is inside a group. An O tag indicates that a token belongs to no group. The B- prefix before a tag indicates that the tag is the beginning of a group that immediately follows another group without O tags between them.

```
# id  12 domain=trial
the         _   _     O
original    _   _     O
ferrari     _   _     B-PROD
daytona     _   _     I-PROD
replica     _   _     O
driven      _   _     O
by          _   _     O
don         _   _     B-PER
johnson     _   _     I-PER
in          _   _     O
miami       _   _     B-CW
vice        _   _     I-CW
```

Figure 2 - Example of CoNLL format

Fortunately, all the three datasets was formatted in the right way, so no more preprocessing was needed.

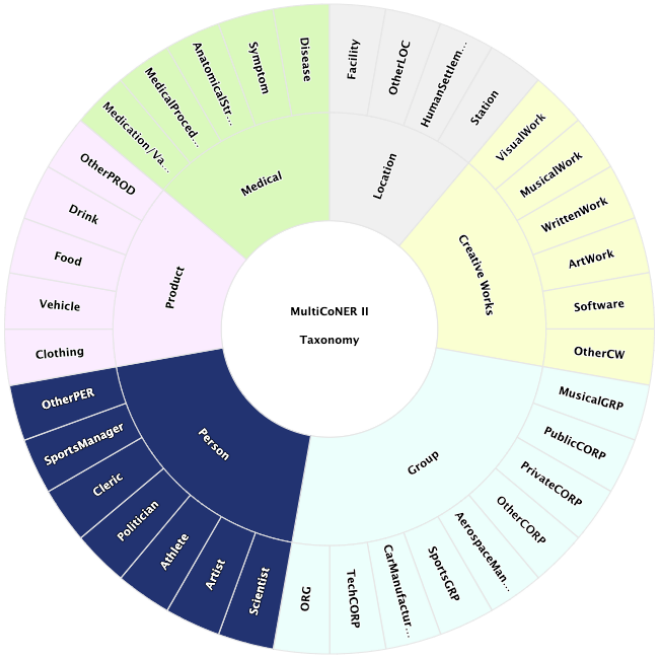
Dataset

The dataset of MultiCoNER Competion II (edition 2023) [5] is a fine-grained dataset.

The fine to coarse level mapping of the tags are as follows:

- Location (LOC): Facility, OtherLOC, HumanSettlement, Station
- Creative Work (CW): VisualWork, MusicalWork, WrittenWork, ArtWork, Software
- Group (GRP): MusicalGRP, PublicCORP, PrivateCORP, AerospaceManufacturer, SportsGRP, CarManufacturer, ORG
- Person (PER): Scientist, Artist, Athlete, Politician, Cleric, SportsManager, OtherPER
- Product (PROD): Clothing, Vehicle, Food, Drink, OtherPROD
- Medical (MED): Medication/Vaccine, MedicalProcedure, AnatomicalStructure, Symptom, Disease

The following figure shows the fine-grained taxonomy of the dataset.

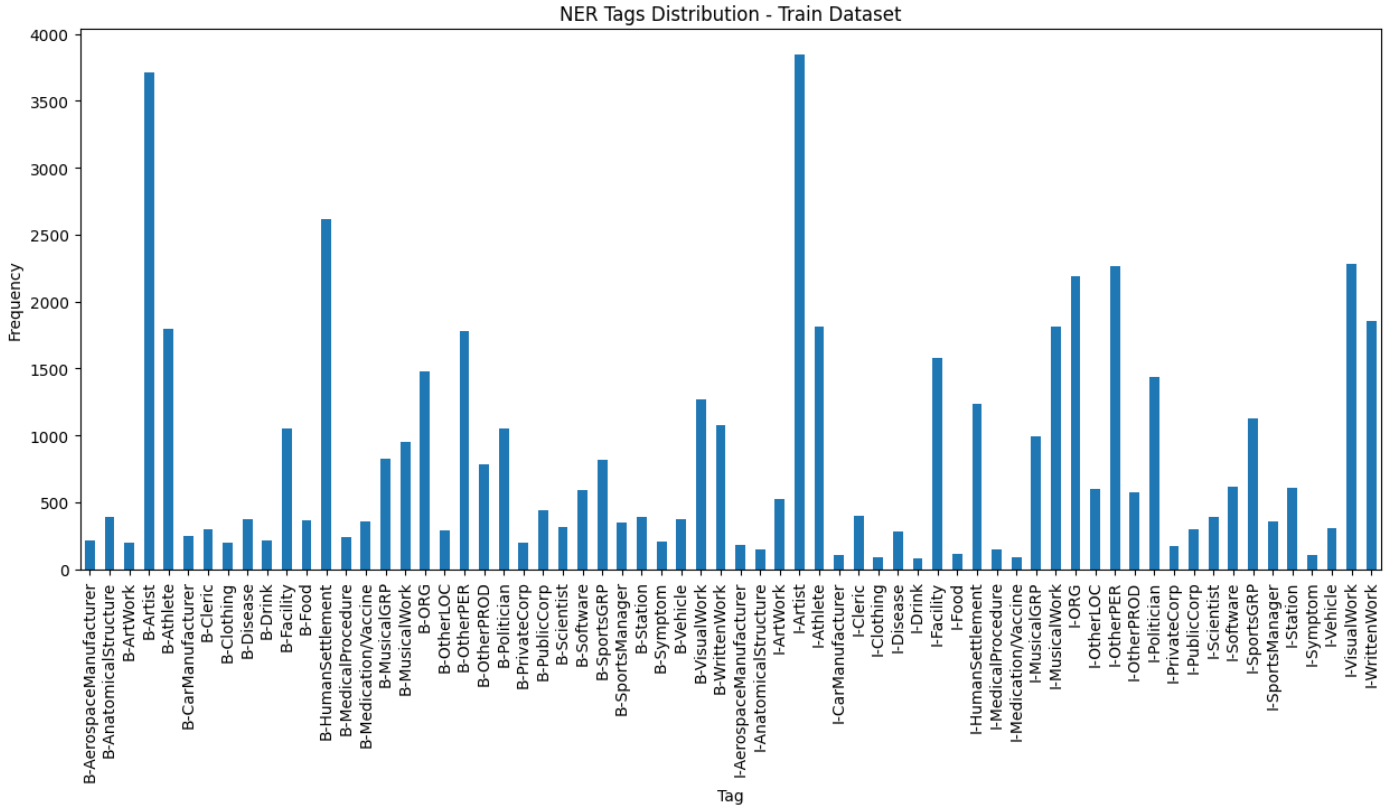


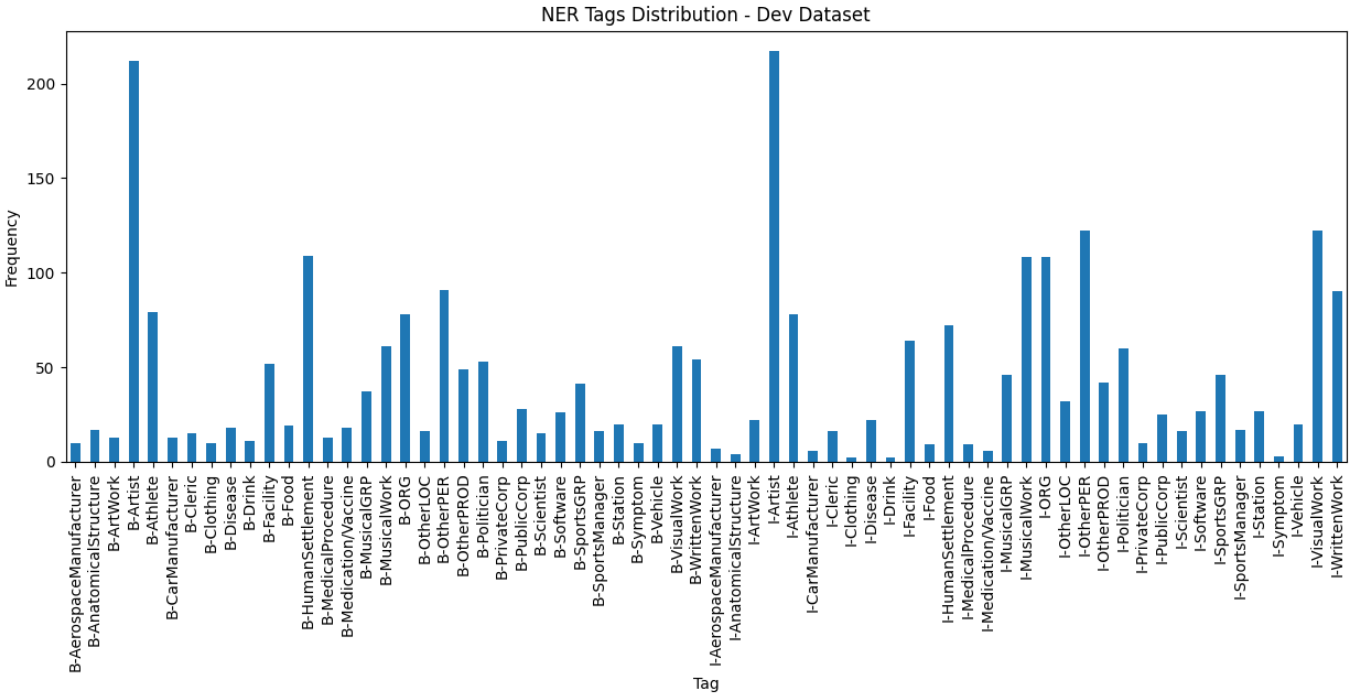
Sentences

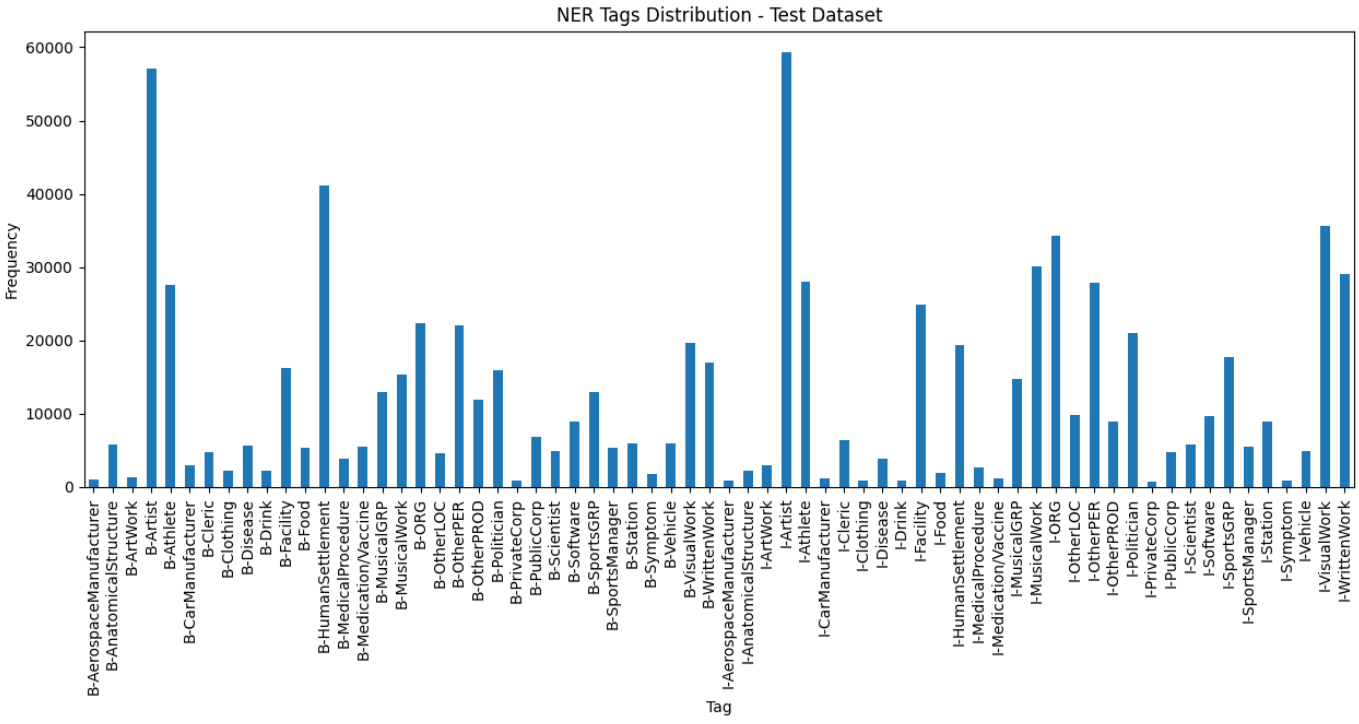
The three datasets used consist of several sentences divided as follows:

- **Train** corpus: 16.778 sentences
- **Dev** (Validation) corpus: 871 sentences
- **Test** corpus: 249.980 sentences

On each of the datasets, a check was carried out on the distribution of the individual tokens with respect to the tags present.
The distribution of tokens with respect to the different tags was divided as follows:




NER Tags Distribution - Dev Dataset
Frequency
Tag
0
50
100
150
200
B-AerospaceManufacturer
B-AnatomicalStructure
B-ArtWork
B-Artist
B-Athlete
B-CarManufacturer
B-Cleric
B-Clothing
B-Disease
B-Drink
B-Facility
B-Food
B-HumanSettlement
B-MedicalProcedure
B-Medication/Vaccine
B-MusicalGRP
B-MusicalWork
B-ORG
B-OtherLOC
B-OtherPER
B-OtherPROD
B-Politician
B-PrivateCorp
B-PublicCorp
B-Scientist
B-Software
B-SportsGRP
B-SportsManager
B-Station
B-Symptom
B-Vehicle
B-VisualWork
B-WrittenWork
I-AerospaceManufacturer
I-AnatomicalStructure
I-ArtWork
I-Artist
I-Athlete
I-CarManufacturer
I-Cleric
I-Clothing
I-Disease
I-Drink
I-Facility
I-Food
I-HumanSettlement
I-MedicalProcedure
I-Medication/Vaccine
I-MusicalGRP
I-MusicalWork
I-ORG
I-OtherLOC
I-OtherPER
I-OtherPROD
I-Politician
I-PrivateCorp
I-PublicCorp
I-Scientist
I-Software
I-SportsGRP
I-SportsManager
I-Station
I-Symptom
I-Vehicle
I-VisualWork
I-WrittenWork


NER Tags Distribution - Test Dataset
Frequency
Tag
0
10000
20000
30000
40000
50000
60000
B-AerospaceManufacturer
B-AnatomicalStructure
B-ArtWork
B-Artist
B-Athlete
B-CarManufacturer
B-Cleric
B-Clothing
B-Disease
B-Drink
B-Facility
B-Food
B-HumanSettlement
B-MedicalProcedure
B-Medication/Vaccine
B-MusicalGRP
B-MusicalWork
B-ORG
B-OtherLOC
B-OtherPER
B-OtherPROD
B-Politician
B-PrivateCorp
B-PublicCorp
B-Scientist
B-Software
B-SportsGRP
B-SportsManager
B-Station
B-Symptom
B-Vehicle
B-VisualWork
B-WrittenWork
I-AerospaceManufacturer
I-AnatomicalStructure
I-ArtWork
I-Artist
I-Athlete
I-CarManufacturer
I-Cleric
I-Clothing
I-Disease
I-Drink
I-Facility
I-Food
I-HumanSettlement
I-MedicalProcedure
I-Medication/Vaccine
I-MusicalGRP
I-MusicalWork
I-ORG
I-OtherLOC
I-OtherPER
I-OtherPROD
I-Politician
I-PrivateCorp
I-PublicCorp
I-Scientist
I-Software
I-SportsGRP
I-SportsManager
I-Station
I-Symptom
I-Vehicle
I-VisualWork
I-WrittenWork

4. Experiment

The training of the model described above, and the execution of the experiments were carried out on different machines. Due to the need for computational power, a first coarse grid search of the hyper-parameters was performed on a Colab notebook, with T4 GPU. A fine grid search and the final model testing was done on an M1 MacBook Pro with an 8-Core CPU and a 14-Core GPU, with 16MB of RAM.

The final model does not use a CRF library, but a version written from scratch and suitably modified. However, the first versions of the model are written and trained with a version of CRF provided by the pytorch-crf library [6] but comparing the results with the version written from scratch, no differences emerged. Therefore, I decided to leave the version from scratch as a part of the model, also for a better understanding of what happens during the train/test phase and easier debugging.

Hyperparameters and Grid Search

To train the model, a series of hyperparameters were selected, each of which contained different values.

A two-phase grid search was carried out:

- A first coarse phase in which I indicated a series of parameters at wider intervals.
- A second phase, fine grid search in which the parameters have been selected in a narrower range than the parameters of the first phase.

The following are the parameters of coarse grid search and fine grid search.

Parameter	Value
Batch size	16,32,64, 128
Learning rate	1e-2, 1e-3, 1e-4, 1e-5, 1e-6
Momentum	0.5, 0.7, 0.8, 0.9
Early stopping	3, 5
Max Epoch	10, 15, 20
Weight decay	1e-2, 1e-3, 1e-4, 1e-5
Weight decay for CRF	1e-3, 1e-4, 1e-5, 1e-6
Learning rate for CRF	1e-2, 1e-3, 1e-4, 1e-5, 1e-6
Optimizer	SGD, Adam

Table 1 - Coarse Grid Search Parameters

Parameter	Value
Batch size	32,64
Learning rate	9e-3, 1e-3, 5e-5
Momentum	0.9
Early stopping	3
Max Epoch	15
Weight decay	5e-5, 2e-3, 9e-4
Weight decay for CRF	1e-6, 5e-6
Learning rate for CRF	4e-3, 8e-5
Optimizer	SGD, Adam

Table 2 - Fine Grid Search Parameters

The code used for the grid search was implemented from scratch. This is a part of the Machine Learning exam that I re-used for this experiment.

Grid Search Result

The execution of the two grid search phases led to the values shown below on validation set, which made it possible to identify the final parameters with which the model was definitively trained:

#	Batch size	Learning rate	Momentum	Early Stopping	Epoch	Decay	Decay CRF	Learning Rate CRF	Optimizer	Accuracy	Recall	Precision	F1 Score
1	64	9e-3	0.9	3	16	5e-5	5e-6	8e-5	Adam	0.80	0.85	0.80	0.83
2	64	9e-3	0.9	3	16	2e-3	5e-6	4e-3	Adam	0.79	0.84	0.79	0.81
3	64	9e-3	0.9	3	16	2e-3	5e-6	8e-5	Adam	0.78	0.83	0.78	0.80

Table 3 - Grid Search's Top Results

So, the final parameter to use for testing the model are:

Parameter	Value
Batch size	64
Learning rate	9e-3
Momentum	0.9
Early stopping	3
Max Epoch	16
Weight decay	5e-5
Weight decay for CRF	5e-6
Learning rate for CRF	8e-5
Optimizer	Adam

Table 4 - Final Model Parameters

MultiCoNER

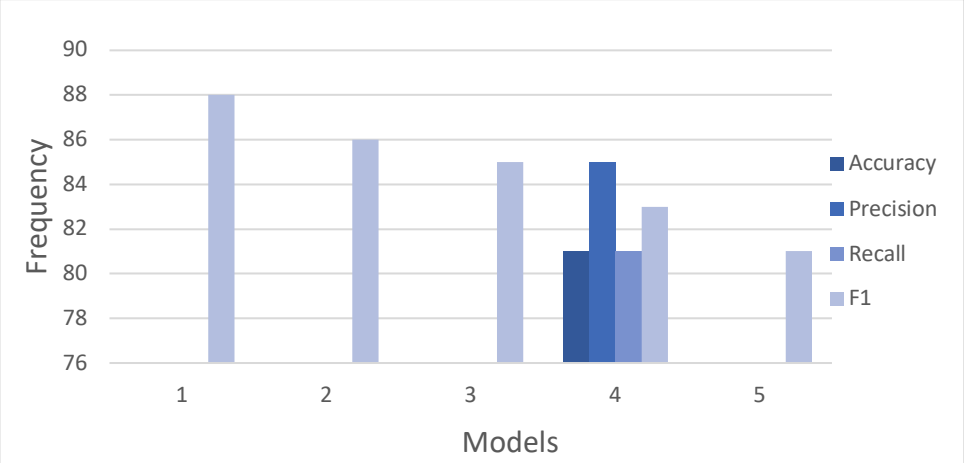
The *Multilingual Complex Named Entity Recognition* (MultiCoNER) [5] is a project developed by Amazon that represents the state of the art for NER task. I adopted the MultiCoNER [4] as reference to check the quality of my project. Amazon provides a code baseline as a starting point on which to build the final model. There are multiple differences between these two projects both in the framework exploited and in model implementation. Regarding the framework the mainly difference is that the MultiCoNER use Pytorch-Lightning for training and testing. Concerning the model implementation, the MultiCoNER deploys a further CRF layer over the BERT. There are others small differences regarding the implementation, but they do not influence the results and thus they will not be mentioned.

As final decision, despite the provided code base, for this project I decided to implement all the code from scratch (except for the Bert Model).

5. Results

The comparison of the results of the model object of this project was made with the results of the MultiCoNER Competition, made public at this link https://multiconer.github.io/results. The only available metric F1 was considered, therefore, the accuracy, precision and recall values for the competition models are not reported. The competition also evaluated the model on a noise dataset. The reported values refer only to the execution of the model on a non-noisy dataset, as the experiment object of this project was also performed on a non-noisy dataset.
The line highlighted in **blue** represents the test result of my model.

#	Accuracy	Precision	Recall	F1
1	n.a	n.a	n.a	88.13
2	n.a	n.a	n.a	86.16
3	n.a	n.a	n.a	85.36
4	**81.76**	**85.34**	**81.26**	**83.14**
5	n.a	n.a	n.a	81.29



The MultiCoNER does not report the subdivision of the evaluation parameters with respect to the single class, which instead was done within the project. Below is the accuracy, precision, recall and F1 values calculated by the model for each class:

TAG	ACCURACY	PRECISION	RECALL	F1
B-AerospaceManufacturer	0,78	0,85	0,75	0,8
B-AnatomicalStructure	0,78	0,84	0,81	0,8
B-ArtWork	0,8	0,85	0,89	0,8
B-Artist	0,79	0,81	0,89	0,81
B-Athlete	0,78	0,80	0,87	0,82
B-CarManufacturer	0,79	0,90	0,88	0,83
B-Cleric	0,78	0,81	0,83	0,79
B-Clothing	0,78	0,81	0,84	0,8
B-Disease	0,8	0,86	0,8	0,79
B-Drink	0,78	0,90	0,86	0,83
B-Facility	0,8	0,87	0,82	0,8
B-Food	0,79	0,88	0,81	0,83
B-HumanSettlement	0,78	0,90	0,89	0,81

B-MedicalProcedure	0,79	0,82	0,83	0,83
B-Medication/Vaccine	0,79	0,89	0,86	0,81
B-MusicalGRP	0,79	0,85	0,89	0,8
B-MusicalWork	0,79	0,81	0,81	0,82
B-ORG	0,78	0,88	0,88	0,82
B-OtherLOC	0,79	0,83	0,86	0,79
B-OtherPER	0,79	0,89	0,88	0,8
B-OtherPROD	0,78	0,90	0,89	0,79
B-Politician	0,8	0,87	0,89	0,83
B-PrivateCorp	0,78	0,89	0,81	0,8
B-PublicCorp	0,8	0,90	0,89	0,79
B-Scientist	0,78	0,83	0,83	0,81
B-Software	0,79	0,84	0,88	0,81
B-SportsGRP	0,79	0,86	0,81	0,83
B-SportsManager	0,79	0,80	0,87	0,79
B-Station	0,79	0,85	0,81	0,81
B-Symptom	0,79	0,87	0,83	0,82
B-Vehicle	0,79	0,89	0,83	0,81
B-VisualWork	0,8	0,86	0,84	0,79
B-WrittenWork	0,78	0,87	0,85	0,82
I-AerospaceManufacturer	0,79	0,83	0,84	0,83
I-AnatomicalStructure	0,8	0,86	0,87	0,83
I-ArtWork	0,78	0,80	0,83	0,83
I-Artist	0,79	0,80	0,83	0,8
I-Athlete	0,79	0,82	0,87	0,83
I-CarManufacturer	0,78	0,88	0,88	0,81
I-Cleric	0,8	0,81	0,85	0,79
I-Clothing	0,79	0,82	0,87	0,81
I-Disease	0,79	0,87	0,84	0,83
I-Drink	0,8	0,88	0,82	0,8
I-Facility	0,78	0,87	0,85	0,83
I-Food	0,78	0,90	0,88	0,81
I-HumanSettlement	0,79	0,80	0,89	0,79
I-MedicalProcedure	0,79	0,83	0,89	0,8
I-Medication/Vaccine	0,8	0,81	0,89	0,83
I-MusicalGRP	0,8	0,84	0,87	0,8
I-MusicalWork	0,8	0,90	0,83	0,8
I-ORG	0,78	0,80	0,87	0,81
I-OtherLOC	0,79	0,85	0,81	0,81
I-OtherPER	0,78	0,88	0,89	0,79
I-OtherPROD	0,8	0,86	0,8	0,79
I-Politician	0,78	0,83	0,84	0,79
I-PrivateCorp	0,78	0,88	0,85	0,81

I-PublicCorp	0,78	0,83	0,87	0,81
I-Scientist	0,8	0,85	0,81	0,82
I-Software	0,8	0,81	0,83	0,83
I-SportsGRP	0,8	0,85	0,83	0,8
I-SportsManager	0,79	0,84	0,84	0,8
I-Station	0,8	0,87	0,85	0,81
I-Symptom	0,8	0,82	0,8	0,81
I-Vehicle	0,79	0,85	0,82	0,79
I-VisualWork	0,78	0,85	0,89	0,8
I-WrittenWork	0,79	0,84	0,88	0,81

6. Conclusion

I would like to conclude this report with some considerations.
Compared to the competition results, my model has lower values in all the evaluation parameters. I don’t know how the participants' models were built and what the underlying architectures are. However, according to the literature for this type of task [1], I think that the choise I’ve made was in the right direction and the differences found in terms of results may depend on several factors. Also can be applied further improvements such as:

- Model training can be further refined in the grid search phase by exploring ranges of parameters other than those indicated.
- A dropout step with a fixed value of 20% has been added in the Feed Forward layer. Exploring different values (even no dropout) could give rise to different performances.
- Different architectures could be explored, changing the layers on top of the Bert Model and using, for example, LSMT in combination with Feed Forward Neural Network and CRF.

7. References

1. *Speech and Language Processing 3rd edition – Daniel Jurafsky, James H. Martin - Stanford Press*
2. *CoNLL. URL: https://universaldependencies.org/format.html.*
3. *Huggingface. URL: https://huggingface.co/*
4. *Multiconer. URL: https://github.com/amzn/multiconer-baseline.*
5. *Multiconer Competition. URL: https://multiconer.github.io/*
6. *Pytorch-crf. URL: https://pytorch-crf.readthedocs.io/*
7. *MultiCoNER: a Large-scale Multilingual dataset for Complex Named Entity Recognition. 2022. Shervin Malmasi, Anjie Fang, Besnik Fetahu, Sudipta Kar and Oleg Rokhlenko.*
8. *GEMNET: Effective Gated Gazetteer Representations for Recognizing Complex Entities in Low-context Input. 2021. Tao Meng, Anjie Fang, Oleg Rokhlenko and Shervin Malmasi. In Proceedings of NAACL.*
9. *Gazetteer Enhanced Named Entity Recognition for Code-Mixed Web Queries. 2021. Besnik Fetahu, Anjie Fang, Oleg Rokhlenko and Shervin Malmasi. In Proceedings of SIGIR.*
10. *NER Evolution: https://paperswithcode.com/sota/named-entity-recognition-ner-on-conll-2003*
11. *IOB Tagging: https://en.wikipedia.org/wiki/Inside%E2%80%93outside%E2%80%93beginning_(tagging)*

8. Appendix A - Conditional Random Field

Let's introduce the CRF more formally, again using *X* and *Y* as the input and output sequences. A CRF is a log-linear model that assigns a probability to an entire output (tag) sequence *Y* , out of all possible sequences Y, given the entire input (word) sequence *X*. We can think of a CRF as like a giant version of what multinomial logistic regression does for a single token. Recall that the feature function *f* in regular multinomial logistic regression can be viewed as a function of a tuple: a token *x* and a label *y* (page 89). In a CRF, the function *F* maps an entire input sequence *X* and an entire output sequence *Y* to a feature vector. Let's assume we have *K* features, with a weight *wk* for each feature *Fk*:

$$p(Y|X) = \frac{\exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)}{\sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)}$$

It's common to also describe the equation by pulling out the denominator into a function Z(X):

$$p(Y|X) = \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)$$

$$Z(X) = \sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)$$

We'll call these *K* functions *Fk*(*X*,*Y*) global features, since each one is a property of the entire input sequence *X* and output sequence *Y* . We compute them by decom- posing into a sum of local features for each position *i* in *Y*

$$F_k(X,Y) = \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)$$

Each of these local features fk in a linear-chain CRF is allowed to make use of the current output token yi, the previous output token yi−1, the entire input string X (or any subpart of it), and the current position i. This constraint to only depend on the current and previous output tokens yi and yi−1 are what characterizes a linear chain CRF. As we will see, this limitation makes it possible to use versions of the efficient Viterbi and Forward-Backwards algorithms from the HMM. A general CRF, by contrast, allows a feature to make use of any output token, and are thus necessary for tasks in which the decision depend on distant output tokens, like yi−4. General CRFs require more complex inference, and are less commonly used for language processing