Homework 1: Named Entity Recognition

Florin Cuconasu 1835605

Abstract

Named Entity Recognition (NER) is a sequence labeling task, whose purpose is to identify for each token in a text its semantic type, such as person, location, corporation etc. I have considered several approaches to build the model, however it seems that a slight improvement of the baseline, with the use of CRF, leads to the best performance in terms of F1 score and time to converge.

1 Introduction

The analysis was performed considering six possible labels, without considering the no entity label O: Person (PER), Location (LOC), Corporation (CORP), Product (PROD), Group (GRP) and Creative Work (CW) (see Table 3). Actually, the number of labels to be classified are 13, as we consider the BIO tagging.

For this task, one of the most suitable models for dealing with sequential input and also for capturing long-range token dependencies is the LSTM architecture. Therefore, I chose it as the core architecture of all my attempted models.

The main models I built use POS tags (section 5.1), character-level embeddings (section 5.2) and token dependencies (section 5.3), along with the word embeddings.

2 Dataset and Preprocessing

The following paragraph describes all the choices and considerations I made regarding the data.

The first point is that the dataset is very unbalanced (see Table 3), as there are about 80% of O labels out of all the labels. If we do not consider the O labels, the most common labels are PER (23%), GRP (20%) and CW (19.6%) (see Table 4). This observation is important to understand the final results, since one of the most misclassified labels is CW.

The dataset is made up of lowercase tokens only, hence we could not take advantage of having additional morphological features like capitalization.

I noticed that the dataset contains some noise, so it may happen that a line contains also punctuation beyond the token of interest. For instance, it usually happens that a line contains a question mark at the end of the actual word, i.e., the question mark is not a new-line-token. Therefore, when constructing the word vocabulary we can consider the same word twice: one is the word itself and the other is the word with a punctuation mark at the end. I tried to train my models with a vocabulary that considered both cases and there were almost no differences; so, I decided to keep only the single word in order to reduce the vocabulary size.

Moreover, there are non-Latin words, such as Russian, Chinese and Arabic words, but I decided to substitute all of them with a special character because they never appear as a named entity (NE). Thus, this will reduce the number of out-of-vocabulary (OOV) tokens regarding word embeddings.

3 Word Embeddings

In all the architectures, I have used pretrained word embeddigs, which lead to a fast convergence with respect to the randomly initialized embeddings. I have tried Word2Vec (Mikolov et al., 2013), FastText (Grave et al., 2018) and GloVe (Pennington et al., 2014). Word2Vec did not performed as good as the other two, probably because it was trained with a much cleaner corpus, where also many types of punctuation have been removed. Since I keep punctuation, FastText and GloVe work better. In particular, FastText is really powerful due to its capability to handle OOV; the advantage of GloVe is that relies on global statistics (word co-occurrence) to obtain word vectors.

Therefore, after the tuning procedure I realized that the best configuration is achieved using the concatenation of the 300d FastText word embeddings with **frozen** weights and the 100d GloVe but continuing to **update** the weights. By freezing also the GloVe weights the final results could be similar, but more epochs are required to train the model (generally 3-5 epochs).

4 Bi-LSTM-CRF Architectures

The core module of the neural network architectures is the Bidirectional LSTM (Bi-LSTM), which is the composition of two LSTMs: one considers the context from left to right and the second in the opposite direction.

4.1 CRF

A state-of-the-art approach is to use LSTM in combination with CRF (Ma and Hovy, 2016) which helps a lot the network in understanding the BIO annotation. For instance, CRF should consider very unlikely to classify I-CORP after B-PER. So, the main advantage of using a Conditional Random Field (CRF) is that each label is not independent from the each other as in the traditional Categorical Cross Entropy.

I also used Cross Entropy as a loss function, but CRF generally performs better.

5 Experiments

This section describes the principal attempts I have done. I was inspired by many papers (Li et al., 2018) and even if the following approaches are part of state-of-the-art architectures, I could not achieve results as I expected; perhaps due to non optimal hyper-parameters tuning.

5.1 Part of Speech Embeddings

The POS tag of a word can be really useful in training the nerual network, since it adds more information about the context and the ”role” of the word in the sentence.

I used the Spacy library (Honnibal and Montani, 2017) to retrieve the POS tags of each token in input. I have tried two different architectures: the first one inputs the POS embeddings to a Bi-LSTM and then concatenates the output with the word Bi-LSTM output; the second one, considers the concatenation of the word and of the POS embeddings and next uses it as the input of the Bi-LSTM.

Both the models gives similar results, yet they do not really improve the Baseline ones.

5.2 Character-level Representation

Several studies (Kuru et al., 2016), (Tran et al., 2017) incorporate character-based word representations with world-level representations as input of the Bi-LSTM. Nevertheless, this type of encoding requires much more computational power with respect to a simple word representation, and a lot of time to be trained. I decided to stop the hyper-parameter tuning when I saw that it slowed down the training by at least 15/20 times.

5.3 Dependency-Guided

Dependency trees capture long-distance and syntactic relationships between words in a sentence (Jie and Lu, 2019). Therefore, the syntactic relations (e.g., nominal subject, object. Figure 1) can potentially infer the existence of certain NEs.

In this case, the representation of a word becomes the concatenation of three embeddings:

$$u_i = [w_i; w_h; v_r]$$

the embeddings of that word (w_i), the embeddings of its head word in the dependency structure (w_h) and the embeddings of the dependency relation (v_r). I used the Spacy library to retrieve the dependency tree of a sentence.

Also in this case, I was unable to achieve comparable results with the improved Baseline.

6 Training

Table 1 shows a comparison of the different model proposed during the experiments. I used Weights and Biases (Biewald, 2020) to keep track of my runs and to choose the best hyper-parameters.

The most suitable configurations regarding limiting overfitting are the use of a Dropout layer before the LSTM and the use of gradient clipping.

7 Results

As we can see in the confusion matrix (Figure 2), the most misclassified labels are B-CW and B-PROD. By analysing some statistics (Figure 3, 4) I observed that those entities are frequently either rare words (e.g., dvd, bmw, nintendo etc.) or even stopwords/articles (e.g., the, a, my, etc.). So, the model easily confuses them with O. Perhaps, by tuning more, for example, the POS hyper-parameters or by adding more data I could reach better results.

References

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Zhanming Jie and Wei Lu. 2019. Dependency-guided LSTM-CRF for named entity recognition. *CoRR*, abs/1909.10148.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. 2016. CharNER: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 911–921, Osaka, Japan. The COLING 2016 Organizing Committee.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2018. A survey on deep learning for named entity recognition. *CoRR*.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *CoRR*, abs/1603.01354.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Googlenews-vectors-negative300.bin.gz - efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Quan Tran, Andrew MacKinlay, and Antonio Jimeno Yepes. 2017. Named entity recognition with stack residual LSTM and trainable bias decoding. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 566–575, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Model	**F1 DEV**	**Epochs**
Bi-LSTM + CCE	62.0	15
Bi-LSTM	66.5	14
Bi-LSTM + (F | G* | DEP)	70.3	10
Bi-LSTM + (F | G* | POS)	71.3	7
Bi-LSTM + (F | G)	72.0	9
Bi-LSTM + (F | G*)	**72.5**	12

Table 1: F_1 scores. CCE, F and G stands respectively for Categorical Cross-Entropy, FastText embeddings and GloVe embeddings. G* indicates that the weights are learned. If not specified the loss is CRF.

Hyper-parameter	**Typical values**
Dropout	[0.3, 0.5]
Initializing Linear Weights	[-0.5, 0.5]
Gradient Clipping	[0.5, 2.5]
StepLR Scheduler	(5, 0.3)
Early Stopping	Patience of 2 epochs
Optimizer	Adam
Activation Function	LeakyReLU

Table 2: Typical hyper-parameters configuration during tuning.

Entity	**Train**	**Dev**
O	192841	10240
PER	10895	629
GRP	9459	567
CW	9267	431
LOC	7154	396
CORP	5962	252
PROD	4480	236

Table 3: Entity Distribution.

Entity	**Train**	**Dev**
PER	23%	25%
GRP	20%	22.6%
CW	19.6%	17%
LOC	15.2%	16%
CORP	12.7%	10%
PROD	9.5%	9.4%

Table 4: Entity Frequency without O.

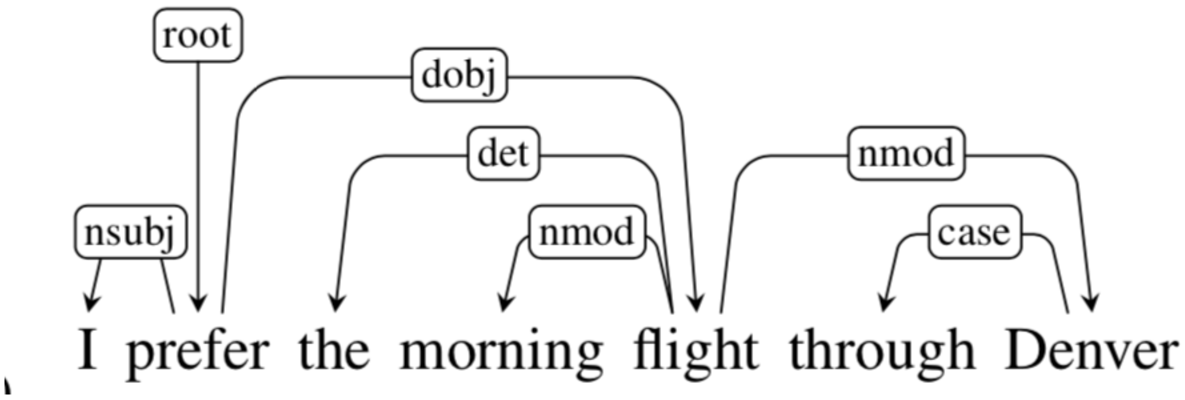
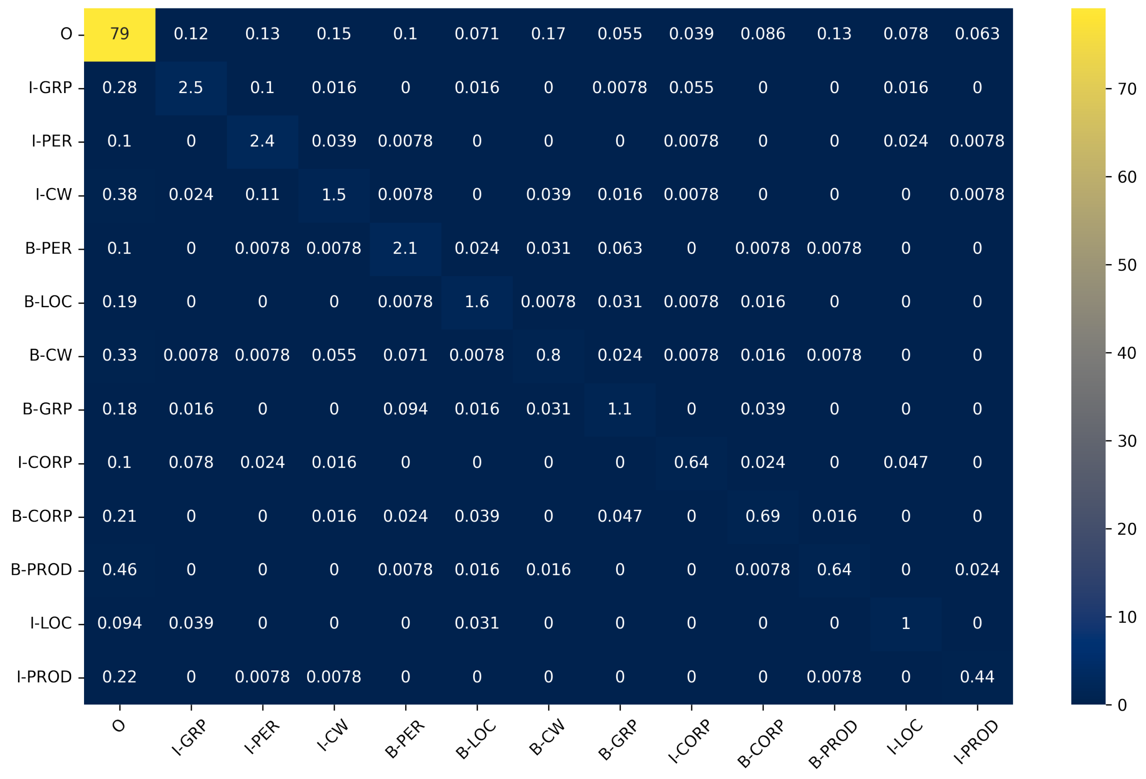


Figure 1: Dependency Tree taken from https://www.analyticsvidhya.com/blog/2021/12/dependency-parsing-in-natural-language-processing-with-examples/



	O	I-GRP	I-PER	I-CW	B-PER	B-LOC	B-CW	B-GRP	I-CORP	B-CORP	B-PROD	I-LOC	I-PROD
O	79	0.12	0.13	0.15	0.1	0.071	0.17	0.055	0.039	0.086	0.13	0.078	0.063
I-GRP	0.28	2.5	0.1	0.016	0	0.016	0	0.0078	0.055	0	0	0.016	0
I-PER	0.1	0	2.4	0.039	0.0078	0	0	0	0.0078	0	0	0.024	0.0078
I-CW	0.38	0.024	0.11	1.5	0.0078	0	0.039	0.016	0.0078	0	0	0	0.0078
B-PER	0.1	0	0.0078	0.0078	2.1	0.024	0.031	0.063	0	0.0078	0.0078	0	0
B-LOC	0.19	0	0	0	0.0078	1.6	0.0078	0.031	0.0078	0.016	0	0	0
B-CW	0.33	0.0078	0.0078	0.055	0.071	0.0078	0.8	0.024	0.0078	0.016	0.0078	0	0
B-GRP	0.18	0.016	0	0	0.094	0.016	0.031	1.1	0	0.039	0	0	0
I-CORP	0.1	0.078	0.024	0.016	0	0	0	0	0.64	0.024	0	0.047	0
B-CORP	0.21	0	0	0.016	0.024	0.039	0	0.047	0	0.69	0.016	0	0
B-PROD	0.46	0	0	0	0.0078	0.016	0.016	0	0	0.0078	0.64	0	0.024
I-LOC	0.094	0.039	0	0	0	0.031	0	0	0	0	0	1	0
I-PROD	0.22	0	0.0078	0.0078	0	0	0	0	0	0	0.0078	0	0.44

Figure 2: Confusion Matrix with Percentages.

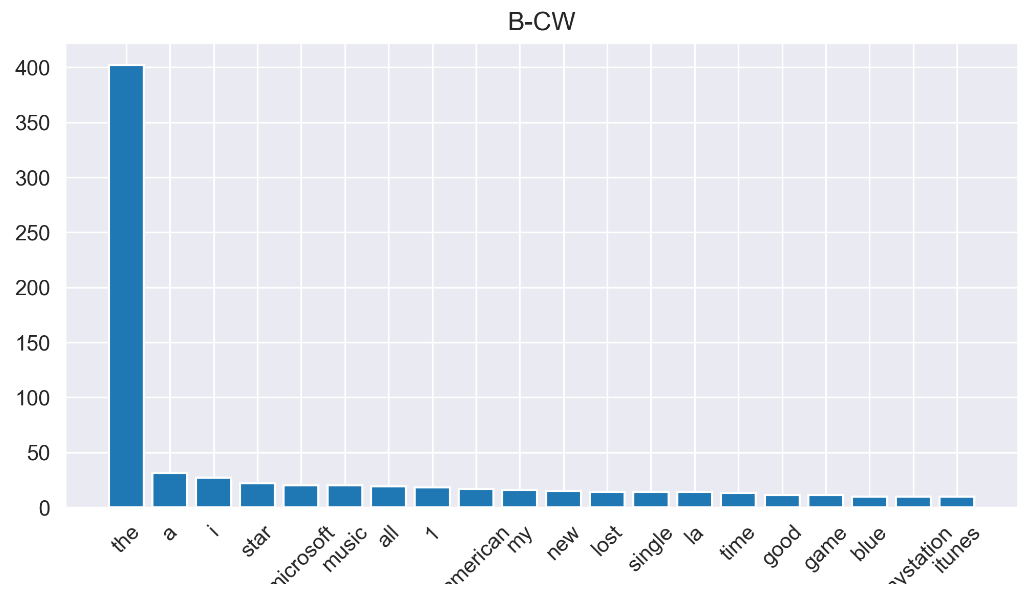


Figure 3: 20 Most common B-CW tokens.

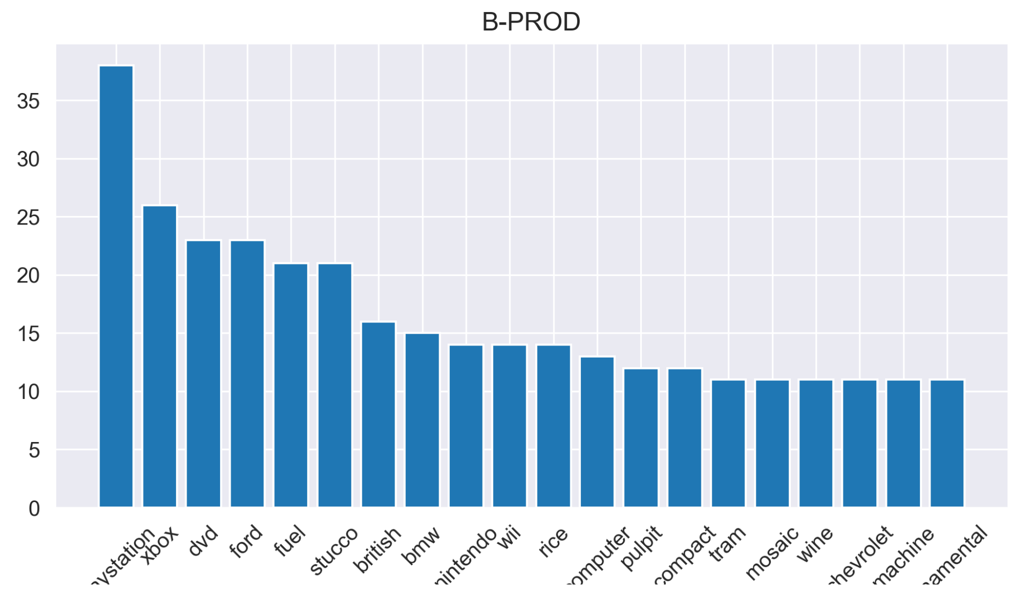


Figure 4: 20 Most common B-PROD tokens.

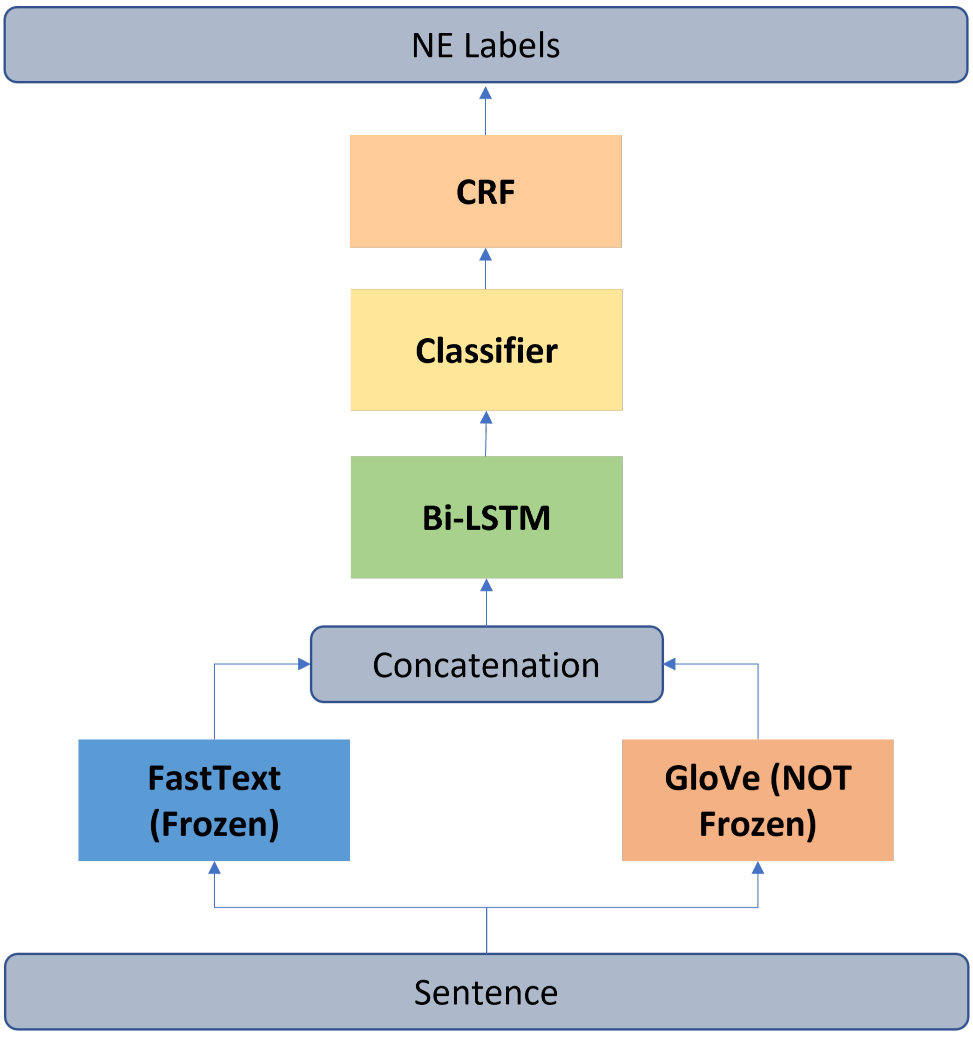


Figure 5: Architecture that gives the best results.