**REPORT EX 4 NLP**

For this assignment, we implement a relation «work\_for» extraction system. To do so, we received file with sentence and another one which contains the labels for the sentences.

We need, first of all, to read the files and to extract features that are important to us.

We chose to treat the relation : *Work\_For*.

**Use of Spacy**

To do so, we used the Spacy package with model « *en\_core\_web\_lg* ».

**Analysis and amelioration**

We create a function which extract entities from the sentences. Afterward, we create pairs of those entities to use them in the future. When we ran and checked the entities pairs we got, we found out that we succeeded to identify only 60% of the « Work\_For » relations.

To solve this problem, we analyzed which entities we did not recognize. And we used multiple approaches to increase the percentage of entities we get.

1. We tried different spacy models to get the maximum entities of the data.
2. We tried to ameliorate the parser, adding constraints. Such as decomposition ‘-‘, to take into consideration the whole entity and not like two different entities. After analyzing the entities we also saw that things like : dot  ‘ . ’, ‘ ’s ’ at the end of entity ,‘the’ at the beggining of the entity or ‘the the’ in the beggining also reduce the results. We decided to clean the entities to get rid of them.

All those things gave us better identification of good labels we increased the number of good labels from 62/109 to 95/109.

1. To calculate precision and recall, we used two different options hard and soft evaluations.
2. Hard : we checked if gold\_entity relation = entity relation in the same sentence.
3. Soft : In the same sentence, does the gold\_entity include the entity or the opposite.

Example :

Entity target: Home Loan Bank of San Fransisco

Gold entity target: Home Loan Bank

In both case, it is supposed to identify because our mission is to find the relation between entities and probably with better parsing we could identify better them.

**Building the model**

We built the model, based on the form of the features and convert it into sparse vector.

We tried to use different features and its combination like distance between the sentences, the words around and their parameters. It worked great on the training, but in the DEV evaluation, we got a lot of failures, part of them was due to overfitting. It didn’t identify well the relations between entities.

After figuring we add the regulization parameter to the model. But It has low precision and recall, we have found that it got confused between the relation « Work\_For » and the « Kill ». Thus, we added a Key Word vector. In the Key Word vector, we put words that belongs to the « Kill » relation like « shoot », « assassinate », « death », « murder », « kill » etc… The model didn’t succeed to indentify them. And doing that, helped a lot !

We also add in the Key Words vector, words that can be useful for the « Work\_For » identification : « work », « head », « serve », « star », « perform » etc…

**Filtering the pair sentences**

We have « Work\_For » sequences with entities that are not relevent. Example : Source = Org, Label = Org, or Source = loc, Label = loc. It’s not helping to train our model. It turns out that it unbalances even more the data.

As a solution, we made a script with a description of what really should be sequences of « Work\_For » : the possible sequences for every label of « Work\_For ».

Why ? We have a unbalanced data, and also sequences that don’t exist so we want them to be ‘False’ because we don’t want to take them into consideration. We would like the model to learn only on the probably true ones. (The pairs that can be right).

The final model gets sentence. It extracts all of the possible pairs of entities. For every entity, it checks : does this sequence exist in the possible sequences ?

If the sequence of pair entities can’t be associated with « Work\_For », the model returns False.

**Model**

In our experiment we tried two different models with various parameters :

* Logistic regression and SVM

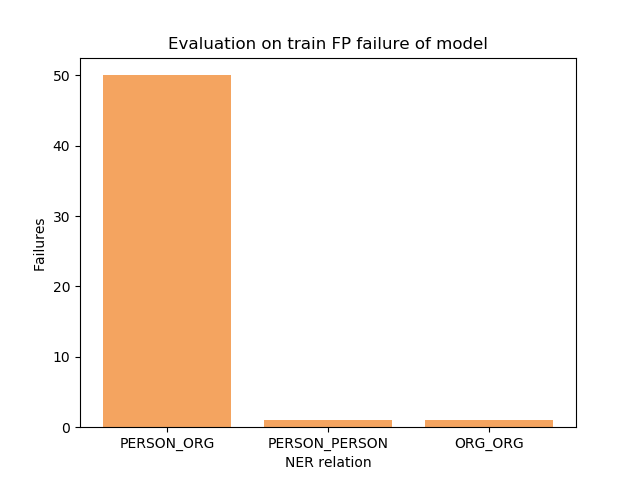
We add the weight class parameter with value « balanced data », to be able to work with unbalanced dataset according to the the frequence of the label.

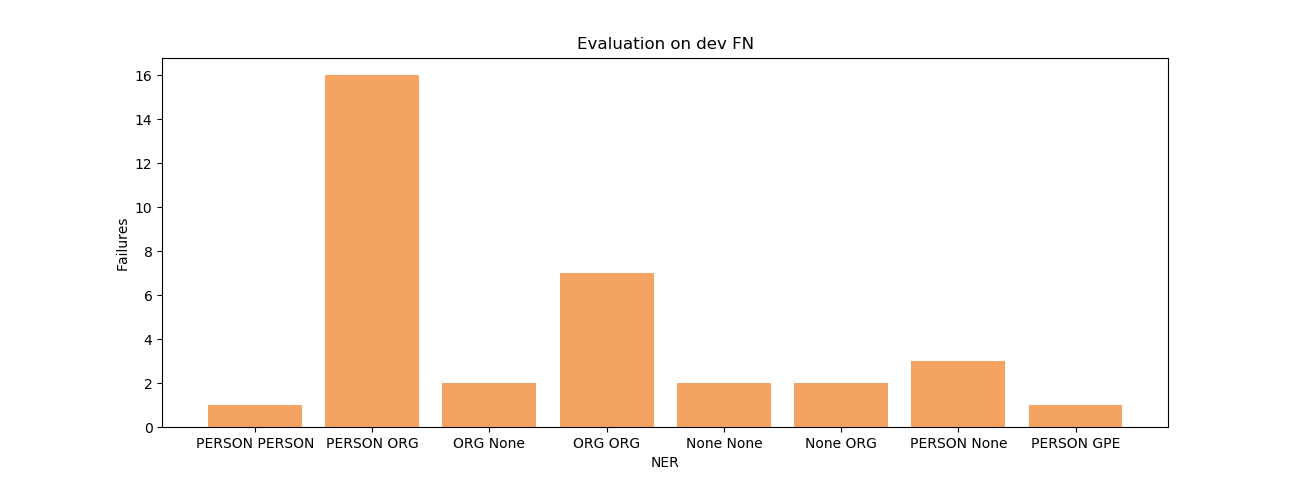
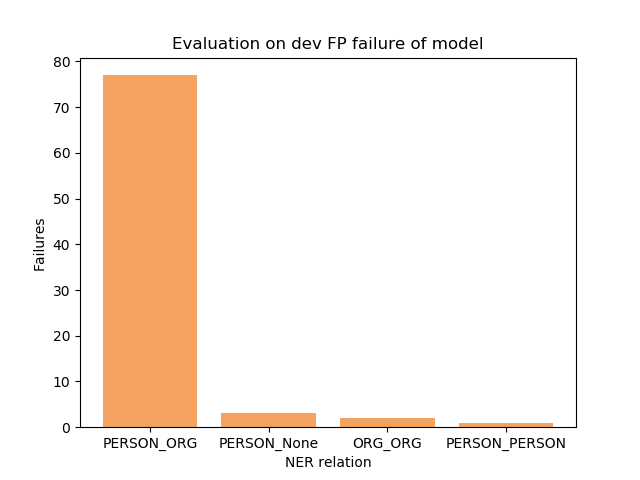
Model Logistic regression optimal parameters wih solver = ‘liblinear’, with penalty = ‘l1’ (regularization).

Train : **Precision**=0.72 **recall**=0.94 **F1**=0.81

Dev : **Precision**=0.57 **recall**=0.76 **F1**=0.65

\*Left graph false negative ner entities failure and right graph false positive wrong labels described. First row describes failures on train and the second on dev.



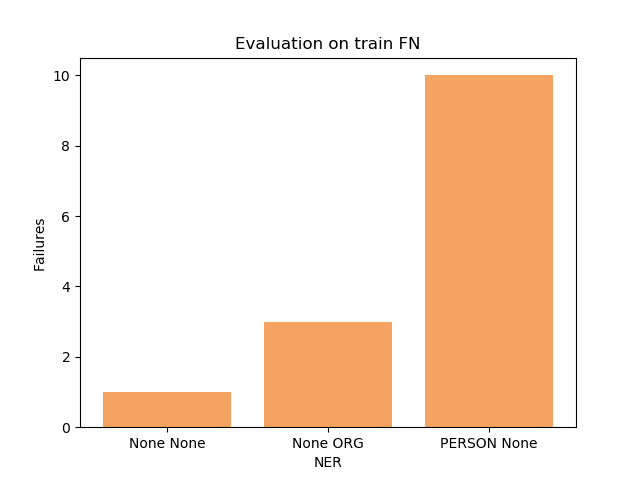


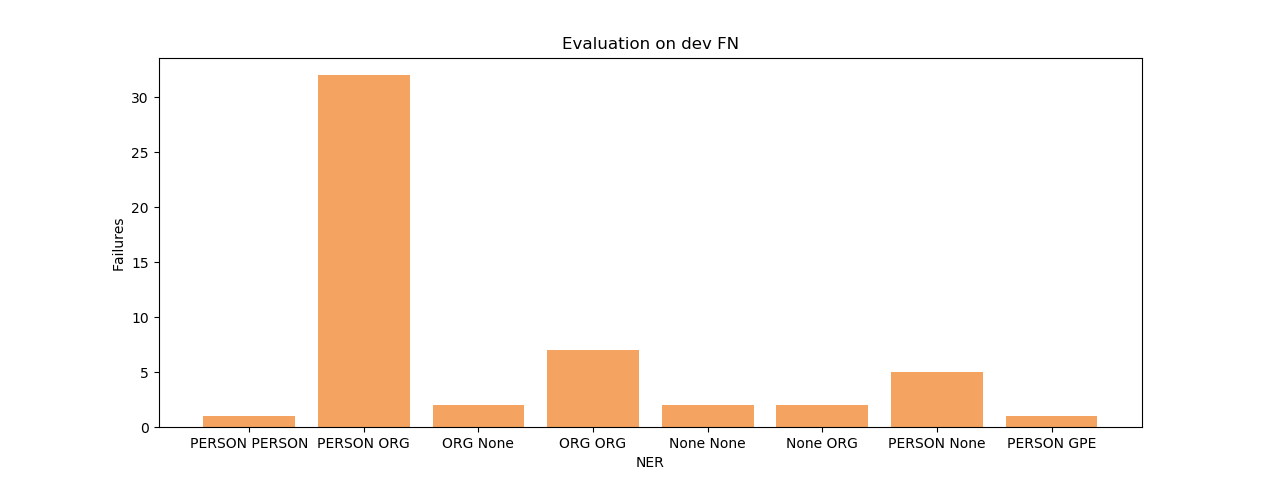
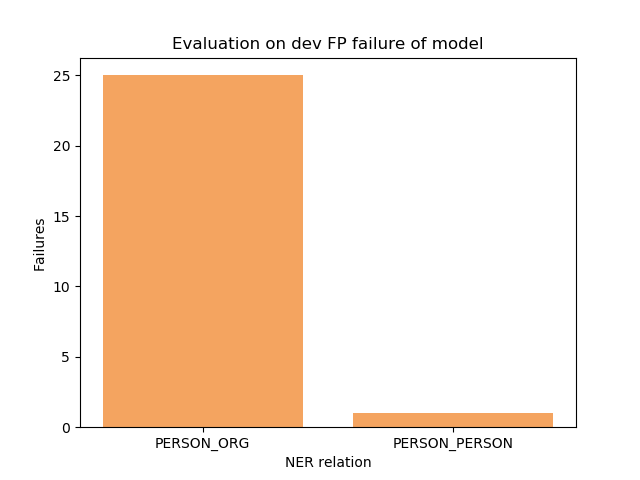
But, it gave us a lot of false positive examples.

The best model in SVM is using Kernel Linear, with dregree=3 and c= 0.3. It gave us the optimal resulst.

Train: **Precision=**0.95 **recall=**0.88 **F1=**0.92

Dev : **Precision**=0.73 **recall**=0.58 **F1**=0.65





**Final Results**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Relation | Train Recall | Train Prec | Train F1 | Dev Recall | Dev Prec | Dev F1 |
| Work\_For | 0.88 | 0.95 | 0.92 | 0.58 | 0.73 | 0.65 |

**Conclusion**

In our experiment we have learned that some failures in preprocessing can fail the process. For example, it identifis Person as a Date (« Winter ») or as an Organization. This could give some uncorrect answers and there can be many relations between Person and ORG which our model failed to identify. In the final architecture we used SVM because we prefer to get better precision than recall for our task.