**REPORT EX 4 NLP**

For this assignment, we need to implement a relation extraction system.

To do so, we received several sentences, with their tree parse, and the relation extracted from them.

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 same in the future. When we ran and checked the entities pairs we got, we found out that we succeeded to identify only 60% of them for the relation « Work\_For ».

To solve this problem, we analysed 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 ‘-‘. We want to take into consideration the whole entity and not like two different entities. After analysing the entities we 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, then, decided to get rid of them.

After several manipulations wih entitites we got to 96/109 entities in dev and train.

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

Example :

entity : Home Loan Bank of San Fransisco

gold\_entity : Home Loan Bank

In both case, it is suppose to identify because our mission is to find the relation between entities and to do so, we used spacy.

GRAPH

**Building the model**

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

We played with different features 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. It didn’t identify well the relations between entities.

After figuring out that it didn’t work well, we have found that it got confused between the relation « Work\_For » and the « Kill » one. 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…

To get the best model, we played with two different models and with their parameters : Logistic regression and SVM

We want our model to deal with the non balanced data, we added : Class\_weight ‘balanced’ that balances the data by changing the weights according to the frequence of the label. We used the SKlearn Package.

The best model in Logistic regression is wih solver = ‘liblinear’, with penalty = ‘l1’ (regularization).

GRAPH

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.

GRAPH

We also tried to give different Kernels but it didn’t give us better results.

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 non balanced 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 get sentence. It extracts all the possible pairs of entities. And if the sequence For every entity, it checks : does this sequence exist in the possible sequences ?

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