In my code I tried to be modular in order to have easier times when reading or modifying the code. A lot of the things written here could be also found in the comments of the code, so I will try to be as concise as possible describing only the most relevant things:

1. **firstRequest(*corpus,mapping*)**: this is the function that evaluates the nlp pre-trained model both token-level and chunk-level. Instead of doing the normal pipeline, first I created a Doc object using the words from the dataset in order to make sure that spacy does not divide for example dates in different tokens; then I applied a sort of manual phrase division in order to preserve the begin and the end of a phrase as in the dataset; after all of this I used the normal pipeline of spacy₁. Then I took the references from the corpus and the hypothesis from the spacy predictions, and compared them using *sktlearn*. In order to compare them I used the function *label\_conversion()* that converts all the spacy NE labels into the ConLL ones using the dictionary *mapping* that contains all the conversions.
2. **ent\_grouping(*txt, is\_string*):** in this function I do all the logic behind the “entity-grouping” used in the second request. Takes in input a string or a Doc object and a Boolean indicating if the *txt* variable is a Doc or not. This function transform the input sentence into a list of lists containing the labels of the entities in order in which they compare in the sentence.
3. **secondRequest(*txt, is\_string*)₂:** this function simply uses the *ent\_grouping()* function, and then counts the frequencies of each type of entity, printing the frequency’s dictionary.
4. **fix\_segm(*txt*):** this is the logic behind the segmentation fix, if needed. Takes in input a string and returns a Doc object in which the *doc.ents* are fixed. The main logic behind this function is: if you find a token with a ‘compund’ dependency, check if it is in the same entity as it’s head; if not, create a new entity in order to fix this. If the token has not a compound dependency simply append it to the *result* list; in the end replace *doc.ents* with the *result* list.
5. **thirsRequest(*txt*)₂:** this function, taken in input a string, applies *fix\_segm()* and prints the output Doc.ents.

NOTES:

1. In the code I ignored the spacy ‘parser’ for the first request because it was not necessary for the request and gave me problems with my sentence-division (so with the shapes of refs and hyps).
2. By default I used an example phrase in order to test these methods.