Davide Dalla Stella 223727

Natural Language Understanding
Second Assignment

To be sure that the program works it’s necessary to check that the file “test.txt” path in the “main” is correct.

Exercise 1:

To read the file the function “read_corpus_conll” of the given program conll.py is used and next only the words without the various tag were read to get each sentence for spacy. The function “get_sentence” return the spacy doc of the sentence and the original corpus of the file. The problem of evaluating spacy NER on conll is that the ent_type are different, so the function “spacytoconll” converts every ent_types in the correct format. The particularity is that, differently from text or iob, the ent_type_ attribute is writable so it is possible change it directly in the token. The function “align_entities” concatenates the ent_iob and the ent_type of the sentence to return a ListOfList in the correct format for the “evaluate” function of the conll.py file. The final step are the effective evaluations. To do them two 1D arrays with the tags of the sentence and the tags of the corpus are created for the sklearn.metrics “classification_report” function. Then for the evaluate conll function it’s necessary have the list of tuple returned by the “align_entities” function and create a list of tuple for the corpus in the same format of the other list of tuple. The results returned by “classification_report” and “evaluate” report the token-level performance and the conll chunk-level performance.

Exercise 2

The function “group_entities” groups the ent_type in a ListOfList for each chunks and return it. If a token is not recognized it’s ent_type is append at the end of the lists because for our purpose it doesn’t matter the order of the elements. The returned list is used by “freq_combination” that it counts each combination to see wich is more frequent. The lists with same elements but in different order are treated as the same combination for programming simplicity.

Exercise 3:

In this last exercise it’s necessary check all the tokens in each entities. If they have a child that is compound but it’s not part of the token’s entity it must be inserted. To do that it’s necessary to check if the child positions is out of the entity using the attributes “child.i” (that gives the child position) and the “entity.start/.end” attribute(that gives the start and the end position of the entity). Then the attribute “child.compound_” must be checked and when it’s sure that the token is a compound the starting point or the ending point of the entity has to be rewrited checking if the position of the child is minor of the starting point or major of the ending point. Considering that a tuple is unwritable the document must be casted in a list so as to succeed to replace the old entity with the new one created using the Span constructor. The last thing to do is substitute all the doc.ents using the list recasted in a tuple.