NLU Assignment 2

Dimitri Vinci (223960)
University of Trento
Via Sommarive, 9, 38123 Povo,Trento TN
dimitri.vinci@studenti.unitn.it

Abstract

This is the report for the second assignment of NLU. The document is divided in three part covering the implementation of three exercises of the assignment. In conclusion it will bereported some observations about the implemented exercises, about some details of implementation, some choices that were made and some conclusions.

1 First exercise implementation

the first exercise can be called from the test_spacy function. It takes as input

- a path in string format pointing to the test.txt file of conll2003 corpus

the output is a tuple that contains two pandas table

- the first one contains the statistics requested for the token evaluation (accuracy for class, O included, and total). They are made both counting and not counting the IOB tag
- the second one contains the statistics requested for chunk evaluations (precision. Recall, f1 etc) for class and total

the implementation is the following:

1. retrieve the data from test file. Create refs list of phrases as requested from evaluate function of conll.py. recreate the original phrases to parse it with spacy
2. parse each phrases with spacy and iterate for each token to extract text and tags to insert them in the list. This list will represent the phrase in refs list in the format requested by the evaluate function in conll.py
3. In this context you have to fix difference in tokenization when you encounter a "-" inside a phrase using the function restructure_tokenisation (token, doc, new_list, problem='-', compound_list=False), where problem parameter was inserted to account to potentially new differences in tokenization and compound_list is used in third exercise and explained after in document. New_list is instead the list of tuples (token, tags) that we want to create for each phrases. The function inserts the compound that spacy has separated with IOB and NER tag if present or O instead, and said to main where to jump in the token iteration
4. Also you have to adapt the tags of spacy to the tags of conll. For that there is the remap(value) function that remaps the values of the spacy tags to conll2003 tags. The ratio behind that is described in conclusion
5. After the list of tuples is put in a list of list as requested from conll.py function evaluate
6. call evaluate_tokes function to evaluate statistics on tokens and evaluate function from conll.py to get statistics for chunks using hyps and refs extracted before. We put the resulting values in pandas table and return them as a tuple.

2 Second exercise implementation

the second exercise can be called from the group_name_entities function. It takes as input

- a list of phrases as was used also in the exercise before

the output is an object from a custom class named statistics and containing a dictionary with as key the chunks name and as value requested statistics. The class statistics has some methods to extract the requested statistics

- get_list_of_lists(self) outputs all list of lists analysed in the format requested (a list of noun chunks and token with NER tag that are not part of noun chunks, in the order of the original phrase)
- get_freq_groups(self, reverse=False), obtains absolute frequencies for all keys in the dictionary (so all chunks or token with NER entities setted). If reverse Is setted the order is descending
- get_rel_freq_groups(self, reverse=False), same as before but with relative frequencies
- get_rel_freq_groups_wouttokens(self, reverse=False) same as before but with relative frequencies not counting in total the single token chunks

the implementation is the following:

1. create statistics object
2. for each phrase, parse the phrase
3. call the function extract_list_of_lists(doc) (the function extract_list of lists takes a doc as an argument and create a list of lists where there is the noun chunks and the individual chunks, for tokens with NER tag setted but not belonging to noun chunk, in the order of the original phrase)
4. send the resulting list of list to the statistics object (inside the statistics class, the method add_phrase(self, new_list) takes a parsed phrase and updates the total count with and without single token chunks, update the dictionary keys and values.)

3 Third exercise implementation

the third exercise can be called from the test_spacy_exercise3 function. The output and input are the same of the first one. The only implementation differences are the following:

- it calls exercise3_function(doc) for each phrase iteration. This function finds the compound relation and find noun chunks elements with no NERtag settled to update it with the tag of head of this element. Contextually it takes note if we have to change IOB tags (for example if these tokens precede other tokens of its compound with the same NER tags that was attributed to them). To obtain these, it returns for each phrases the doc with the ent_type_ modified if it was the case and a list of length equal to the length of the doc filled with either None or the new IOB tag If these must be changed.
- We call restructure_tokenisation with an extra parameter that is the list produced by the function quoted before. This is necessary to account to the problem of change tokenization while taking in account possible changes in IOB

4 Conclusion and Observation

Some general observation

- The code of utilities function used to implement the point of assignment is described in comments and documentations inside assignment.py
- The accuracies between class and the overall accuracy were computed using the formula learned in the machine learning course of PhD Farid Melgani in the University of Trento. So the accuracy for a class in a multiclass problem is defined as the correct prediction for that class divided by the sum of correct prediction for that class and the number of samples the was not categorized correctly in that class. The overall accuracy is defined as the sum of correct prediction for all classes divided by all

predictions. Obviously other definitions could be used for example scikit learn computes the Jacard score of order one .

- To calculate single token accuracies was reuse some code of conll.py, contextually the file conll.py was commented to understand its logic
- To remap the tags from Spacy to Conll it was used as references the papers on the same topic and corpus (retrievable http://nlpprogress.com/english/named_entity_recognition.html), the references quoted in the next sections and spacy documentations. To corroborate the matching we used some statistics that are about which tag was attributed by spacy in contrast to CONLL to the same tokens contained in test.txt. the result of this operation is contained in extra_result. Obviously it was not a real significant statistic analysis because some maps could be errors of spacy but it strengthen the ratio behind the chosen mappings. The list of the mapping (when necessary, so when they were not the same or they were used in the test file) is the following
    - PERSON to PER (straightforward)
    - DATE to O (around 7 thousand token that spacy map to date were mapped from conll to O. the others were errors of spacy. Connl does not recognize Date).
    - Same as DATE for ORDINAL, CARDINAL, TIME, PER, MONEY, QUANTITY, PERCENT, PRODUCT
    - GPE meaning countries, cities or states is mapped to LOC(in fact conll map GPE spacy tagged token to LOC at least four times higher than other tags)
    - EVENT meaning named event is mapped to MISC(in fact conll map EVENT spacy tagged token to MISC at least ten times higher than other tags)
    - FAC meaning buildings, airport, highways etc is mapped to LOC
    - NORP meaning religious or political group is mapped to MISC(in fact conll map NORP spacy tagged token to MISC at least ten times higher than other tags)
    - GPE meaning countries, cities or states is mapped to LOC(in fact conll map GPE spacy tagged token to LOC at least four times higher than other tags)
    - LAW meaning named law is mapped to MISC
    - LANGUAGE meaning name of language is mapped to MISC (in fact conll map LANGUAGE spacy tagged token in MISC)
    - WORK_OF_ART meaning name of artwork is mapped to MISC (in fact conll map WORK_OF_ART spacy tagged token to MISC at least three times higher than other tags

About second exercise

- It was decided to take as parameter a list of phrases because the implementation of a system to extract them from a conll2003 file was implemented also in the first point. However a function called obtain_phrases(path) that take a path to the conll data and extract the phrases and put them in a list was implemented to call it in test.py and test the second point. It is also that in assignment.py

About third exercise

- It is divided in two main functions. One that is test_spacy_exercise3(doc) do as described pretty much the same thing of the first exercise. The exercise3_function(doc) implement the real logic of extending NER tags and modify IOB tags if necessary .

Some conclusions (the data are in the result file in the repo)

- As said in spacy documentation the performance on conll were degraded in part due to different NER tags used in conll, in part because spacy uses different tokenization and obviously due to different type of texts that were used to train spacy. The performance with single tokens are generally higher but they lowered with chunk performances because of the difficulties of segmentation. at token level we have higher performances for O class (that was predictable because they are the majority elements). Taking in to account the IOB tags, the I tags marked classes showed lower performance obviously because of the difficulty of finding semantic correlated elements. In both cases LOC and MISC have higher performances in contrast with ORG that has the lowest (It was checked that spacy tend to attribute often to a name the tag ORG or the other way around). At the chunks level best precision is also showed in LOC and MISC and worst in ORG but far away from a baseline around 0.8/0.9. the recall is very low on ORG this is due as I said before to the fact that spacy tend to give to an ORG a different named tag. Good recall is achieved for locations instead.
- It was not found a noun chunks that it is particularly overrepresented (maybe the test set was created to use as much different name chunks NER tags combos as possible). However obviously single token chunks are the majority especially for DATE, ORG, PERSON (due to the content of the test file obviously). About chunks longer than one we have that tags associated with PERSON are the majority.
- Extend the NER tags to token of a compound worsened generally the performances because probably if you do that after training process you can extend these tags unduly to not semantic relevant element of the chunks. So maybe do these during training as done in paper[1] could be profitable. We can however find that some single token performances slightly increase, in particular the performances of PER and ORG tag, especially for I tagged NER labels.

References

[1] Alt, C., Henning, L., Hu, C., Wang, C., Meng, Y., Zhang, H. *Bootstrapping Named Entity Recognition in E-Commerce with Positive Unlabelled Learning*

[2] Bird, S., Klein, E., Loper, E. (2009), *NaturalLanguage Processing with Python*

[3] Jurafsky, D., Marin, J.H.(2000), *Speech andLanguage Processing*