Assignment 3: Text normalization

Adil Rhoulam

arhoulam@ens-paris-saclay.fr

1 The principales of the system

1.1 The libraries and functions used:

In a python notebook, I have implemented a method to normalize twitter texts, I have used the following libraries:

- *numpy*
- from $\_future\_$, I have imported division in order to have a float as a result when dividing two ints
- *nltk, nltk.download('punkt')*: I have used *word_tokenize* in order to separate sentences into tokens.
- *context2vec*: in order to predict words according to context
- *re*: in order to return the words in a dowloaded dictionary
- *Collections*: Imported *Counter* in order to construct the dictionary from collected words by re.

The implemented functions:

- *find_most_similar*: inspired from the evaluation function in the context2vec library that takes a line where the word to predict is between brackets '[ ]' and the number of results wanted and it returns a list containing the words and their cosinus similarity sorted decreasingly.
- *candidates(word)*: that take a word and return all the possible words that are 1 or 2 distant from it w.r.t edit distant
- *known(words)*: a function that take a set of words and return the ones that exists in the dictionary
- *edits1(word)*: that returns all possible words with an edit distance of 1 either by replacement, removing or insertions of letters from 'a' to 'z'.
- *edits2(word)*: returns possible words of an edit distance of 2 from a given word
- *edit_distance(s1,s2)*: a function that calculate the edit distance between two strings and that is programmed using Dynamic Programming.
- *correction_context(word,candidates)*: a function that take a word and some candidates, calcul the edit distance between the word and the candidates, return the first one with an edit distance of 1 if exists, otherwise the first one with an edit distance of 2 otherwise use the function *correction_spelling(word)*.
- *correction_spelling(word)*: take a word and return the first ranked candidate from the ones given by the function *candidates(word)* using as a metric the frequency of the word in the dictionary (basic approach).
- *join_text(tokens,path)*: that write the normalized twitts in a text file path

1.2 The approach used to normalize the twitters data:

First, we open the CorpusBataclan and read its lines. Then each line is pre-processed by filtering the tokens beginning with @or # that are related to twitts (hashtags), filtering the URLs, the string 'RT' that means 'retweet' from the beginning of each line or twit and also whitespaces. Then we filtre non ascii codes (that have and order superior than 128) and transform the characters to lower chars. After that, each line or sentence is cut into tokens. After that, we create a list 'nouns' containing only the alpha characters so it's filters all the numbers and punctuation symbols.
After the first cleaning, we parse the nouns in each line and verify if they exists in the dictionary, if it's so we don't have to correct them, otherwise if there exist a context (i.e words after and/or before the word to correct), we correct it bu recovering the *find_most_similar* function that find the best 10 or more similar words according to the context as candidates and then use the function *correction_context(word,candidates)*, otherwise if there is no context, we return the spelling correction if it exists using the *correction_spelling(word)* function.

2 System evaluation

2.1 Strengths and weaknesses points

The algorithm that I used handles the misspelled or incorrect words using its context or spelling with a high accuracy, I also recovered the twitts with the same punctuation and with corrected texts. The algorithm handles also some named entities as $'Francois, Paris...'$ but not holland since it replaced it by hollander, so the algorithm is not very efficient for named entity. It also doesn't know what to do for one-to-many words as RT so I was obliged to remove it since I used the pre-trained context2vec model and it's not very efficient to those type of words. It doesn't work on those two type because I used only two functionalities: the context correction and the spelling one while Named Entities and one-to-many words doesn't depend on those two functionalities and we must add other functionalities using word embedding in order to handle those cases.
I have tested the algorithm over 50 000 twitts and not all the 1M twitts since the time for parsing and correcting each line is in average 6s/100twitts and will take a long time to correct all the twitts using only CPU. The first 50 000 twitts after normalizing will be joined with the python code.

2.2 Remark:

I have not used the shell script since I didn't use classes and I didn't notice this recommendation earlier. Please used the python notebook. It's easier to compile by only changing the paths of the corpus and the writing file. Thank you for your understanding.