Assignment 1 Tweet Tokenization and Normalization

Approach and Code Development

I created the tokenizer using regular expressions, built in string manipulation methods of Python and the libraries Enchant and NLTK as english dictionaries. I used the principles of regular expressions and morphemes taught in class for creating the tokenizer.

I begun by doing a complete analysis of all the conditions specified in the problem statement. Considering the requirements for tokenization on finding any punctuation marks or hashtags and smilies and the manipulations required for date and time, it was clear that I should develop the code incrementally. I divided the requirements into logical groups and developed code for them one by one as you could examine from each of the functions in the codebase as well.

Firstly, I solved the hashtags ('#') requirement. If any string contained a hashtag, the characters before that hashtag and after that were tokenized separately. If there were multiple hashtags each was tokenized separately as well as per the requirement. If a hashtag was found just before a blank space, it was ignored as suggested from one of the examples. A similar function was written for 'User References' as well considering the character '@' instead of the hashtag.

Before tokenizing for all the punctuation marks it was necessary to check for URL's since valid URL's could get tokenized into different tokens because of the punctuation marks they contain. I spent some time building a regular expression which could handle all test cases to check for any valid URL and return true or false. In case a valid URL was found no further processing is done on such a string.

I handled the special case of apostrophes, smilies along with the rest of the punctuation marks. In case of a finding a smilie those characters were immediately tokenized along with the characters before them in the string and further part of the string is evaluated. For an apostrophe the string is tokenized and a function to tokenize that further is called later in the code. For all the rest punctuation marks I kept tokenizing the string before the mark, then the punctuation mark and evaluate the rest of the string.

After this I wrote a function to check if there was any reference to a particular date, month or year made in the tweet. The function checks if there is any reference to any month using regular expressions to satisfy all cases irrespective of spelling or letter case. It also checks for any date or year reference around the month reference. Whenever a reference is found all the referred tokens are removed and a single token of the said format is added. A similar procedure using another function is carried out to check for any reference made to time in the given string.

Another function is written to check if a string (without spaces in between) is not by itself a valid english word but in fact is a combination of multiple valid english words. In such a scenario as specified in the test cases I create a token for each of the valid english words the string contains and remove the earlier token. The method to figure out all valid english words in the string is done by a right to left traversal in the string checking if any valid word gets formed at any stage. The enchant and NLTK library is used here to check the validity of the word in the english language. However, the libraries contain a lot of false positives especially of 2 lettered english words which yields to somewhat lower accuracy using them rather than not using them at all. Hence the code written for the same has been commented in the submission and could be used with any other suitable better library.

Datasets and Experiments

I used the various tweets in the large number of files I got on the twitter website itself to find out tougher and complicated test cases than the ones I had earlier thought of. The test cases I found there helped me not just in testing my code rigorously but also to get a more in depth understanding of the problem statement during the early stages of the development.

Along with using tweets from the above mentioned data sets, I ran a large number of experiments and test cases myself during the development of my code. A usual check that needs to be performed if code is being developed incrementally is to ensure that all previous modules work properly on addition of any new logical module. I ran many such experiments on addition of every new function to ensure satisfaction of all possible test cases.

Collaborators

During the early stages of the development of the tokenizer I discussed the problem statement and the possible approaches to solve the problem with my classmate Prathamesh Mane.