Tweet Tokenizer and Normalizer

Introduction:

This is a tweet tokenizer which tokenizes tweets without using any NLP library. It depends heavily on regular expressions. I have referred Python Official Documentation to study re module.

Logic:

The tokenizer uses regular expressions to create tokens from a tweet. It reads an input file containing a tweet on each line and it calls tokenizeTweet function. The regular expressions are used in order. It checks for the URL, Email, Username, Hashtag, Date, Time, Hyphenated words, Clitics, Abbreviations, Words, Emojis and Punctuations in order. Twitter username starts with @ so each individual word starting with @ would be a user reference. Similarly each word starting with # would be a hashtag.
This program detects date in various formats like dd/mm/yyyy, mm/dd/yyyy, yyyy/mm/dd, dd mmm, yyyy etc. It converts these date to canonical format CF:D:year-month-date. Tokenizer finds dates using regex and passes it to normalizer which converts it to canonical format. Similar logic is used to convert time to canonical format CF:T:time:timezone. The timezones are stored in a file named timezones.txt in data directory. The program dynamically loads all the timezones and creates a regex.
Clitics are detected using regular expressions and normalizer converts them to standard words depending on the next word. Clitics like ‘m, ‘ve, ‘re are expanded to am, have and are respectively as there are no other expansions possible. The expansion of ‘ll depends on the attached word. If ‘ll is attached to I or we, it is expanded to shall. In other cases it is expanded to will. The clitics like ‘d and ‘s depend on current word as well as next word for expansion. It ‘s is attached to a word ‘let’ then it is expanded to ‘us’. If the word after ‘s ends with ‘ing’ then it is expanded to ‘is’. If the word after ‘s ends with ‘ed’, ‘en’, ‘un’, ‘wn’, ‘ht’ or is a irregular verb then ‘s is expanded to has. In all other cases it is kept as ‘s which is used to denote possession. Similarly if the word after ‘d ends with ‘ed’, ‘en’, ‘un’, ‘wn’, ‘ht’ or is an irregular verb then it is expanded to ‘had’. Otherwise, if ‘d is attached to I or we, then it becomes ‘should’. In all other cases ‘d is expanded to ‘would’.
Abbreviations are detected using regex and spaces between them are removed if present. Words and punctuations are relatively easy to detect using regular expression.
The file containing standard emojis is stored in data directory.
The tokenizeTweet function returns a list of tokens. It is used to print the output to the output.txt file.

Interesting datasets:

The tokenizer is not 100% accurate. It fails for some unusual results as no intelligence is used. Following are the examples I have come across in which the tokenizer fails:

1. He’s done. : Can be expanded as He is done and He has done.
2. Please click-https://www.coep.org.in : It fails to tokenize the URL as a whole and considers click-https as a hyphenated word.
3. May 13 Reasons Why become the bestselling novel : 13 Reasons Why is a name of the novel but the tokenizer considers ‘May 13’ as a date.

Resources:

1. Official Python Documentation of re module
2. Official Python Documentation of argparse module
3. Online sources for emojis and irregular verbs