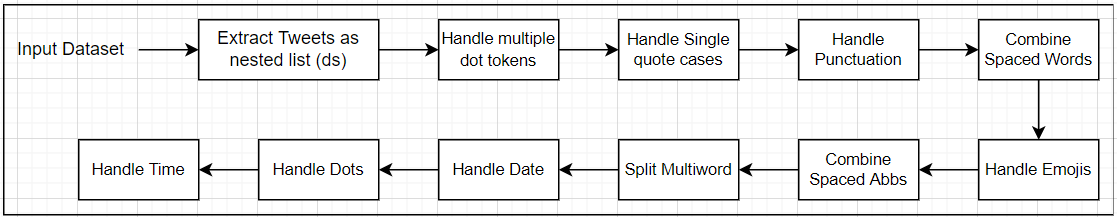
**NLP – Assignment 2: Tokenizer and Normalizer: Rohaan Advani (111903151)**

**Program Creation and Flow Diagram: (Each box represents a function)**



**General Flow of Functions:**

1. All Functions have the tweet nested list as an input parameter.
2. This nested list is a list of each tweet separately. Each tweet is tokenized on the spaces originally.
3. In each case we search for tweets /tokens with respect to specific patterns, characters, or strings in tokens and append such tweets /tokens to an external list. In doing so, we do not destroy the consistency of the original tweet nested list.
4. This external list is then processed as per case token by token to generate the new token the original token must be replaced by.
5. A list is created of the original token and new token.
6. We perform a search in the tweet nested list for the original token and replace it with the new token with the help of an index number.

**FOR THE CREATION OF ALL THE FUNCTIONS, THE FOLLOWING EXPERIMENTS HAVE BEEN CONDUCTED BY ME:**

**Specifications – Multiple Dot tokens:**

1. Extracted tokens have a multiple dot sequence, for example – ‘…’.
2. We break down strings into character lists.
3. We iterate over these lists to collect words and dot sequences.
4. These words and dot sequences are then replaced with the original token in the tweet nested list.

**Specifications – Handle Single Quote Cases:**

1. Single quote cases are tokens having ‘(single quote) in them.
2. These cases can be separated into 2 categories – the cases where the quote is at the start or end of a token or cases where the quote is in between the characters of the token.
3. The first case is handled by separating the quote into a different token and the word into a different token with the expected tokens having the pattern “ s’ ” at the end which is handled by converting it to a new token “ ‘s “ as mentioned in the assignment and separating the word.
4. The second case has 2 sub-cases. The first is “ ‘s “ wherein a new token “ ‘s “ is created and separated from the token as mentioned in the assignment. The second is handling clitics. To do this I created a dictionary of all possible clitics as the value and the tokens to replace them as keys.
5. Using the above methods was able to handle all single quote cases successfully.

**Specifications – Handling Punction:**

1. This function takes a second input parameter which is the character punctuation that we must handle.
2. This function has been run in a loop to handle punctuations added to the following punctuation list element by element.

['"', ',', '>', '<', ';', '+', '~', '|', '(', ')','&', '$', '^', '{', '}', '[', ']', '\*', '?', '!', ':', '=', '%']

1. I excluded punctuation like dot, /, and – as they affect other cases that follow.
2. While extracting tokens with punctuation always leave out emojis, hashtags, user mentions, and URLs.
3. Tokens are split on punctuation marks and each word and punctuation mark has been made a separate token before replacing it in the nested tweet list.

**Specifications – Combining Space Words (r e t w e e t): Follow the same for spaced abbreviations:**

1. Looked for tokens in a tweet where there was a sequence of single-character tokens.
2. Combined them and replaced them in the original list.
3. Remove excess tokens.

**Handling emotes has no specifications. We must break tweets before and after each emote is identified.**

**Specifications – Split Multi-word (iGuess):**

1. Look for words having an upper-case letter in between the token.
2. Split words before and from the letter into two different tokens from the replacement.

**Specifications – Handling Date: Similarly Handle Time Formats and convert to CFT:**

1. Date formats of dd-mm-yyyy or yyyy-mm-dd or dd-mm-yy or yy-mm-dd and dd/mm/yyyy or yyyy/mm/dd or dd/mm/yy or yy/mm/dd can be handled by splitting at – or / checking which part refers to the day, month, and year.
2. Note that in some cases the day or month might have a single character thus we must add a 0 prior to it.
3. This is then reformatted into CFD type as per question and replaced.
4. Date in word form first must be converted to yyyy-mm-dd or dd-mm-yyyy form then to CFD.
5. This is done by searching the month token in the months list.
6. Note that in case any part of the CFD is unavailable it can be replaced by ‘?’.

**Handling full stops can be done by ignoring tokens that are abbreviations, decimal numbers, URLs, or ‘a.m.’ or ‘p.m.’. Tokens are split into full stops and a full stop is made a separate token.**

**Handling / can now be done using the handle punctuation function since it will no longer affect the dates.**

**Specifications – Handling Hyphens:**

1. Hyphen tokens must be collected with the exception of Hashtags, User Mentions, URLs, CFDs, and CFTs.
2. They can now be separated where the hyphen is made part of the second token, for example – US-based = US -based. In order to handle the hyphen case like US-Japan, named entity recognition will be required to identify that the 2 tokens represent the same topic which in this case is countries.

**RESULT –** The cases were handled using the above methodology and sequence without disruption in other cases. Outfit file generated as per requirement by assignment.

**NOTE –** I was unable to find datasets that conducted tweet tokenization as per the requirements of the assignment. This assignment has been discussed with no one and it has been done solely using continuous coding and debugging.