Natural Language Processing(MSAI 337), Winter 2022

Homework #1: Text Cleaning & Processing

[Group 4: Bodhisatta Maiti, Shreyas Lele, Tejul Pandit, Vinit Todai]

The source text used for the assignment was generated using the Wikidata Query Service. The data was cleaned and processed as per the instructions mentioned in Q1 to Q4 of the assignment hand-in. The results and findings are as follows -

Q1. Tokenize the text using NLTK and convert all text to lowercase.

The entire text corpus was present in .txt format. The NLTK package was used to extract all the sentences. Using the sentence level information, preprocessing was performed using the function 'text_tokens'. The function would take a sentence as input (tokenized by NLTK), loop through each word in the sentence and convert the word to lowercase, remove the escape character '\n' and finally apply NLTK's word tokenizer.

This would create a list of lists consisting of tokenized words and special characters in each sentence.

Sample text corpus:
"He was Speaker of the House of Representatives of Thailand 1973–1974. He was the thirteenth Prime Minister of Thailand, serving in office from 1975 to 1976 between Seni Pramoj, his brother\'s, terms."

Processed output:
```
[['he', 'was', 'speaker', 'of', 'the', 'house', 'of', 'representatives', 'of', 'thailand', '1973–1974', '.'],
['he', 'was', 'the', 'thirteenth', 'prime', 'minister', 'of', 'thailand', ',', 'serving', 'in', 'office',
'from', '1975', 'to', '1976', 'between', 'seni', 'pramoj', ',', 'his', 'brother', "'s", ',', 'terms', '.']]
```

Q2. Replace (i) years, (ii) decimals, (iii) date days, (iv) integers and (v) all other numbers with <year>, <decimal>,<days>,<integer>and<other>tags,respectively.

A new function num_tokens is created to process the numeric tokens. A list of tokens is taken as an input to the function and each token present in the list passes through a series of regex expressions and conditions.

(i) To identify if the token signifies a year, the regex expression check is the token either starts with 1 followed by 3 digits or the token starts with 20 followed by 2 digits. If the condition matches, the token is substituted with '<year>' tag.
(ii) To identify if the token signifies a decimal number, the regex expression checks if the token has digits separated by '.'. To ensure that the period is taken as a decimal point and is not confused with the regex character '.' which matches any character except line-break, '\' is added

before the period character. This matches the character '.'. The token which matches the regex condition is replaced with <decimal> tag.
(iii) A list of month names with respective variations is created. For example, jan and january or feb and february and so on. Each token is then checked if it is numeric and is followed by or preceded by a token present in the list of month names. If the condition matches, the numeric token is replaced with the <days> tag.
(iv) A token that only consists of a numeric value is replaced with a <integer> tag. To ensure that tokens such as 12th that have a non-numeric character are not replaced with <integer> tag, the regex expression has a boundary character placed before and after the numeric identifier.
(v) All other numeric characters are replaced with the <other> tag. From the remaining tokens if a number is present in the token, it will be replaced with the <other> tag.

The function first loops through the entire list of tokens to identify year and decimal numbers. The list is then searched for date days. Lastly, the list of tokens is traversed for integer and other remaining numbers.

Q3. Split the corpus into training, validation and test sets to approximate an 80/10/10 distribution

The processed list consisting of the list of tokens is first randomized. The randomization ensures that similar data does not group together. This is essential in our case since we need to capture the maximum variety of words present in the corpus. Once the list has been randomized, the size of the training set is calculated which is 80% of the total list. Thus, training size equals the length of list of the list of tokens multiplied by 0.8. The list is split at the point of training size which means that the training corpus consists of the first n lines in the randomized set where n is equal to training size. The remaining corpus in the randomized set is split equally into test and validation. Each of the training, validation, and test lists are converted into a flat list of tokens.

Q4. Apply a frequency threshold of three to the corpus and report summary statistics. Summary statistics should include the (i) number of tokens in each split, (ii) the vocabulary size, (iii) the number of <unk> tokens, (iv) number of out of vocabulary words, (v) the number of types mapped to <unk>, (vi) the number of stop words in the vocabulary and (vii) two custom metrics of your choice.

The tokens that occur less than 3 times are dropped. The summary statistics for resultant training, validation and test corpus is calculated.

Indicator	Training set	Validation set	Test set
Number of tokens	413396	46297	47455
Vocabulary	10244	2275	2392
Number of <unk> tokens	-	116	138
Out-of-vocabulary words	0	9	10
Number of types mapped to <unk>	-	34	43
Number of stopwords	288	163	168
Number of punctuation tokens	57741	7221	7258
Number of types of punctuation	22	16	17

The two additional metrics are number of punctuation tokens and types of punctuation. Our motivation to select the metric was to assess the amount of information that is expressed in terms of punctuation marks. The use of punctuation marks varies with use cases. For instance, punctuation marks hold no significant value for performing text classification tasks. Hence, an assessment of such information provides insight into the distribution of worded data and data in form of punctuation marks. Additionally, types of punctuation helps to assess if there is one type of punctuation repeated throughout the corpus or there are different types of punctuation marks present in the corpus.

<u>Bonus Questions -</u>

Q8. Construct a word-piece tokenization of the source text using either the byte-pair encoding algorithm covered in class or a Huggingface word-piece tokenizer.

The Huggingface tokenizer is created using the source text. BertWordPieceTokenizer package is used wherein the tokenizer was set for minimum threshold of 3 and vocabulary size of 5000.

Q7. The query used to produce the source text.

The source text is generated using the following query conditions -
(i) Date of Birth is after 01 January 1909.
(ii) Date of Death is before 01 January 2010.
(iii) The person is considered erudite (Person has great knowledge and thus has an occupation which belongs to a subclass of erudite).
(iv) The person studied at a University.