Q1: Tokenizing

1. **Create a text file with, 50 words in it and make sure it has items that challenge the tokenizer(e.g., I.B.M and other weird things). But, you need to make these up yourself, so you think about what it is doing.**

At first, I created the following text based on these criteria, (Hyphenation, Number, acronyms, etc).

“Tom& jerry was the first broadcast on March 20, 1940, in the USA. It has been popular all over the world since then and is still a really popular TV series today. In fact, It has been translated into multiple languages and was also shown on the BBC in the past.”

Load the file in and use nltk.word_tokenize() on it. Report the list of tokens that are produced from it and note any oddities that arise. Comment on these oddities and how they might be handled.

Tokens = ['tom', '&', 'jerry', 'was', 'the', 'first', 'broadcast', 'on', 'march', '20', ',', '1940', ',', 'in', 'the', 'usa', '.', 'it', 'has', 'been', 'popular', 'all', 'over', 'the', 'world', 'since', 'then', 'and', 'is', 'still', 'a', 'really', 'popular', 'tv', 'series', 'today', '.', 'in', 'fact', ',', 'it', 'has', 'been', 'translated', 'into', 'multiple', 'languages', 'and', 'was', 'also', 'shown', 'on', 'the', 'bbc', 'in', 'the', 'past', '.']

In the tokens, there are a large number of punctuations that can be stripped in the pre-processing section. In addition to this, there are some abbreviations of proper nouns that should be better to be completed in the pre-processing section. Furthermore, there are some numbers and it is also beneficial to convert numbers into English words for the next steps.

2. **Now, take this output from the tokenizer and do the normalization step.**

As mentioned above. In the normalization, It will be divided into three steps. Lowercase the text, remove punctuations and Convert numbers to English words.

Tokens = ['tom', 'jerry', 'was', 'the', 'first', 'broadcast', 'on', 'march', 'twenty', 'one', 'thousand', ' nine', ' hundred', 'forty', 'in', 'the', 'usa', 'it', 'has', 'been', 'popular', 'all', 'over', 'the', 'world', 'since', 'then', 'and', 'is', 'still', 'a', 'really', 'popular', 'tv', 'series', 'today', 'in', 'fact', 'it', 'has', 'been', 'translated', 'into', 'multiple', 'languages', 'and', 'was', 'also', 'shown', 'on', 'the', 'bbc', 'in', 'the', 'past']

3. **Now, take the output from normalization step and run it through a pos-tagger. Report this output as your answer and highlight any inaccuracies that occur at this stage.**

Pos_tag = [('tom', 'NN'), ('jerry', 'NN'), ('was', 'VBD'), ('the', 'DT'), ('first', 'JJ'), ('broadcast', 'NN'), ('on', 'IN'), ('march', 'NN'), ('twenty', 'NN'), ('one', 'CD'), ('thousand', 'NN'), (' nine', 'NN'), (' hundred', 'VBN'), ('forty', 'NN'), ('in', 'IN'), ('the', 'DT'), ('usa', 'NN'), ('it', 'PRP'), ('has', 'VBZ'), ('been', 'VBN'), ('popular', 'JJ'), ('all', 'DT'), ('over', 'IN'), ('the', 'DT'), ('world', 'NN'), ('since', 'IN'), ('then', 'RB'), ('and', 'CC'), ('is', 'VBZ'), ('still', 'RB'), ('a', 'DT'), ('really', 'RB'), ('popular', 'JJ'), ('tv', 'NN'), ('series', 'NN'), ('today', 'NN'), ('in', 'IN'), ('fact', 'NN'), ('it', 'PRP'), ('has', 'VBZ'), ('been', 'VBN'), ('translated', 'VBN'), ('into', 'IN'), ('multiple', 'JJ'), ('languages', 'NNS'), ('and', 'CC'), ('was', 'VBD'), ('also', 'RB'), ('shown', 'VBN'), ('on', 'IN'), ('the', 'DT'), ('bbc', 'NN'), ('in', 'IN'), ('the', 'DT'), ('past', 'NN')]

We can see from the output above that there are some incorrect categories. The majority of them is that it divides numbers into singular nouns. And it also mistakenly identifies the number hundred as a verb. Besides, It also classifies proper nouns BBC and USA, and the adverb of time today, as singular nouns, and plural nouns series as singular nouns.

Q2: Stemming and lemmatizing

1. **Tokenize a new-text-file(50 words) and the stem it using Porter Stemming. Report your answer and some of weird things that Porter Stemming does.**

I took a snippet of text from the stock news as follows. This text also contains hindrances when analyzing text.

“The losses were historic by any measure. The pound shed 3.2%. In fact Ten-year gilt yields seeing their biggest one-day surge on record in Bloomberg data through 1989, closing 33 basis points higher on the day at 3.83%. The (FTSE) 100 Index sank 2%.”

Using_Porter = ['the', 'loss', 'were', 'histor', 'by', 'ani', 'measur', '.', 'the', 'pound', 'shed', '3.2', '%', '.', 'in', 'fact', 'ten-year', 'gilt', 'yield', 'see', 'their', 'biggest', 'one-day', 'surg', 'on', 'record', 'in', 'bloomberg', 'data', 'through', '1989', ',', 'close', '33', 'basi', 'point', 'higher', 'on', 'the', 'day', 'at', '3.83', '%', '.', 'the', 'ftse', '100', 'index', 'sank', '2', '%', '.']

The output shows its rules as follows.

a. Removing all plurals and the -ed or -ing suffix (marked in red above).

b. Turning terminal y to i when there is another vowel in the stem (marked in blue above).

c. Removing a final -e (marked in green above).

But it still removes the s from the non-plural noun as well as third-person singular verbs (marked in yellow above).

2. **Tokenize the new-text-file and then lemmatize it using the WordNetLemmatizerl; note you may have to pos-tag the sentences first and then convert the tags to make this work. Report the result of these steps and point out some of the things that look wrong.**

Using_Wordnet = ['The', 'loss', 'be', 'historic', 'by', 'any', 'measure', '.', 'The', 'pound', 'shed', '3.2', '%', '.', 'In', 'fact', 'Ten-year', 'gilt', 'yield', 'see', 'their', 'big', 'one-day', 'surge', 'on', 'record', 'in', 'Bloomberg', 'data', 'through', '1989', ',', 'closing', '33', 'basis', 'point', 'high', 'on', 'the', 'day', 'at', '3.83', '%', '.', 'The', 'FTSE', '100', 'Index', 'sank', '2', '%', '.']

The words marked in red look incorrect which are all turned into its prototypes.

3. **Compare the outputs from Porter Stemming and the Lemmatization of the same file.Which do you think is the best to use and why?**

By removing punctuation, and numbers, both PoterStem and Wordnet are compared with Rawtext as shown below.

Rawtext	PoterStem	Wordnet
The	the	The
losses	loss	loss
were	were	be
historic	histor	historic
by	by	by
any	ani	any
measure	measur	measure
The	the	The
pound	pound	pound
shed	shed	shed
In	in	In
fact	fact	fact
gilt	gilt	gilt
yields	yield	yield
seeing	see	see
their	their	their
biggest	biggest	big
surge	surg	surge
on	on	on
record	record	record
in	in	in
Bloomberg	bloomberg	Bloomberg
data	data	data
through	through	through
closing	close	closing
basis	basi	basis
points	point	point
higher	higher	high
on	on	on
the	the	the
day	day	day
at	at	at
The	the	The
FTSE	ftse	FTSE
Index	index	Index
sank	sank	sank

I think it will depend on what kind of text is being analyzed and whether we want to go for speed of processing or higher accuracy. When I compare the process of running PoterStemmer with the process of running WordNet, I can feel that the processing time of the former is much less than the latter. As to the accuracy, PoterStemmer will produce a huge amount of errors in the processing, which we can see from the table above, due to its rules. WordNet has higher accuracy but it takes time longer which will do more processes in handling the text. So in my view, I will choose WordNet as the best. It does not matter especially when we use WordNet to deal with such a large volume of text data that accuracy is more essential than speed. The more accuracy we get, the less time we spend.