dlnd_tv_script_generation

January 3, 2021

1 TV Script Generation

In this project, you'll generate your own Seinfeld TV scripts using RNNs. You'll be using part of the Seinfeld dataset of scripts from 9 seasons. The Neural Network you'll build will generate a new ,"fake" TV script, based on patterns it recognizes in this training data.

1.1 Get the Data

The data is already provided for you in `./data/Seinfeld_Scripts.txt` and you're encouraged to open that file and look at the text. >* As a first step, we'll load in this data and look at some samples. * Then, you'll be tasked with defining and training an RNN to generate a new script!

In [2]:
```
"""
DON'T MODIFY ANYTHING IN THIS CELL
"""
# load in data
import helper
data_dir = './data/Seinfeld_Scripts.txt'
text = helper.load_data(data_dir)
```

1.2 Explore the Data

Play around with `view_line_range` to view different parts of the data. This will give you a sense of the data you'll be working with. You can see, for example, that it is all lowercase text, and each new line of dialogue is separated by a newline character `\n`.

In [3]:
```
view_line_range = (0, 5)

"""
DON'T MODIFY ANYTHING IN THIS CELL THAT IS BELOW THIS LINE
"""
import numpy as np

print('Dataset Stats')
print('Roughly the number of unique words: {}'.format(len({word: None for word in text.s

lines = text.split('\n')
```

```
        print('Number of lines: {}'.format(len(lines)))
        word_count_line = [len(line.split()) for line in lines]
        print('Average number of words in each line: {}'.format(np.average(word_count_line)))

        print()
        print('The lines {} to {}:'.format(*view_line_range))
        print('\n'.join(text.split('\n')[view_line_range[0]:view_line_range[1]]))
```

```
Dataset Stats
Roughly the number of unique words: 46367
Number of lines: 109233
Average number of words in each line: 5.544240293684143

The lines 0 to 5:
jerry: do you know what this is all about? do you know, why were here? to be out, this is out...

jerry: (pointing at georges shirt) see, to me, that button is in the worst possible spot. the se

george: are you through?
```

1.3 Implement Pre-processing Functions

The first thing to do to any dataset is pre-processing. Implement the following pre-processing functions below: - Lookup Table - Tokenize Punctuation

1.3.1 Lookup Table

To create a word embedding, you first need to transform the words to ids. In this function, create two dictionaries: - Dictionary to go from the words to an id, we'll call `vocab_to_int` - Dictionary to go from the id to word, we'll call `int_to_vocab`

Return these dictionaries in the following **tuple** `(vocab_to_int, int_to_vocab)`

```
In [4]: import problem_unittests as tests
        from collections import Counter


        def create_lookup_tables(text):
            """
            Create lookup tables for vocabulary
            :param text: The text of tv scripts split into words
            :return: A tuple of dicts (vocab_to_int, int_to_vocab)
            """
            # TODO: Implement Function
            counts = Counter(text)
            vocab = sorted(counts, key=counts.get, reverse=True)
            vocab_to_int = {word: i for i, word in enumerate(vocab, 0)}
```