dlnd_tv_script_generation

April 2, 2022

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 [1]: """
        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 [2]: view_line_range = (0, 10)

        """
        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

        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('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 10:
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 

george: are you through?

jerry: you do of course try on, when you buy?

george: yes, it was purple, i liked it, i dont actually recall considering the buttons.
```

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 [3]: import problem_unittests as tests

        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
            from collections import Counter
            counts = Counter(text)
            vocab = sorted(counts,key=counts.get,reverse=True)

            vocab_to_int = {word:ii for ii, word in enumerate(vocab,1)}
            int_to_vocab = {ii:word for ii, word in enumerate(vocab,1)}
            # return tuple
            return (vocab_to_int, int_to_vocab)


        """
        DON'T MODIFY ANYTHING IN THIS CELL THAT IS BELOW THIS LINE
        """
        tests.test_create_lookup_tables(create_lookup_tables)

Tests Passed


In [4]: vocab_to_int, int_to_vocab=create_lookup_tables(text)

In [5]: vocab_to_int

Out[5]: {' ': 1,
         'e': 2,
         't': 3,
         'o': 4,
         'a': 5,
         'r': 6,
         'i': 7,
         'n': 8,
         'h': 9,
         's': 10,
         '\n': 11,
         'l': 12,
         'y': 13,
         'g': 14,
         'u': 15,
         '.': 16,
         'd': 17,
         'm': 18,
         'w': 19,
         ':': 20,
         'c': 21,
         'k': 22,
         ',': 23,
         'p': 24,
         'f': 25,
         "'": 26,
         'b': 27,
         'j': 28,
         '?': 29,
          'v': 30,
          '!': 31,
          '(': 32,
          ')': 33,
          '-': 34,
          'x': 35,
          '"': 36,
          'z': 37,
          'q': 38,
          '*': 39,
          '[': 40,
          ']': 41,
          '0': 42,
          '1': 43,
          '2': 44,
          '5': 45,
          '3': 46,
          ';': 47,
          '4': 48,
          '9': 49,
          '8': 50,
          '#': 51,
          '7': 52,
          '&': 53,
          '6': 54,
          '/': 55,
          '$': 56,
          '`': 57,
          '%': 58,
          '<': 59,
          '>': 60,
          '_': 61,
          '\x92': 62,
          '=': 63,
          '}': 64,
          '+': 65,
          '{': 66,
          '\x93': 67,
          '\x94': 68,
          '': 69,
          '\\': 70,
          '~': 71,
          '\x91': 72}
```

```
In [6]: int_to_vocab
```

```
Out[6]: {1: ' ',
         2: 'e',
3: 't',
4: 'o',
5: 'a',
6: 'r',
7: 'i',
8: 'n',
9: 'h',
10: 's',
11: '\n',
12: 'l',
13: 'y',
14: 'g',
15: 'u',
16: '.',
17: 'd',
18: 'm',
19: 'w',
20: ':',
21: 'c',
22: 'k',
23: ',',
24: 'p',
25: 'f',
26: "'",
27: 'b',
28: 'j',
29: '?',
30: 'v',
31: '!',
32: '(',
33: ')',
34: '-',
35: 'x',
36: '"',
37: 'z',
38: 'q',
39: '*',
40: '[',
41: ']',
42: '0',
43: '1',
44: '2',
45: '5',
46: '3',
47: ';',
48: '4',
49: '9',
50: '8',
```