HW 5 Word Generator Design Document

Saul Shanabrook

Pseudocode

```
class WordGen:
    String corpus
    Table table
    String start_two_letters
    int max_length

    def main(inputs);
        corpus = get_text_from_file(inputs[0])
        w = WordGen(corpus, *inputs[1:])
        print w.table_text()
        print w.generate_text()

    def WordGen(corpus, start_two_letters, max_length):
        assert start_two_letters is two letters
        assert max_length > 3
        add_text(corpus)
        this.start_two_letters = start_two_letters
        this.max_length = max_length

    def add_text(corpus):
        for index in corpus.length - 3:
            table.add(corpus[index:index+1], corpus[index+2])

    def table_text():
        return table.as_string

    def generate_text():
        generated_text = start_two_letters
        while generated_text.length < max_length:
            new_char = table.choose_char(generated_text[-2:])
            if new_char == null:
                break
            generated_text += new_char
        return generated_text

class Table:
    Vector<Association<String, FrequencyList>> data

    add(prefix, suffix):
        assert prefix is 2 letter string
```

```
    choose_char(prefix):
        return vector.find(prefix).value().choose()

    as_string():
        string = “”
        for ass in data:
            string.append(ass.key() + “→” + ass.value() + “\n”)
        return string

class FrequencyList():
    Vector<Association<Character, Integer>> data

    add(character):
        data.find(character).value() ++

    choose():
        // maybe use http://stackoverflow.com/questions/2140787/select-random-k-eleme
nts-from-a-list-whose-elements-have-weights
        select_based_on_weights(data.keys(), data.values())


    as_string():
        total_values = sum(ass.values())
        return “,”.join([ass.key() + “:” + ass.value() / total_values + “%” for ass i
n data]
```

Test Usage

(Pseudo) Java API

```
class WordGenTests:
    def test_table_text():
        w = new WordGen(“aab”);
        assert w.table_text == “aa→b:100%”
        w = new WordGen(“aabaaf”);
        assert w.table_text == “aa→b:50%,f:50%\nab→a:100%\nba→a:100%\naa→f:100%”

    def test_select_next():
        w = new WordGen(“aab”);
        assert w.selectNext(“aa”) == “b”
        assert w.selectNext(“ab”) == null

    def test_generate_text():
        w = new WordGen(“aaa”)
        assert w.generate_text(“aa”, 3) == “aaa”
        assert w.generate_text(“aa”, 6) == “aaaaaa”
        assert w.generate_text(“ab”, 6) == “”
        assert w.generate_text() == “aaa”

class TableTests:
    def test_add()
        t = new Table()

        assert raises error:
            t.add(“aaa”, char(“a”))
        assert raises error:
            t.add(“a”, char(“a”))
        assert raises error:
            t.add(“”, char(“a”))

        t.add(“aa”, char(“a”))
        assert t.as_string() == “aa→a:100%”
        t.add(“aa”, char(“b”))
        assert t.as_string() == “aa→a:50%,aa→b:50%”

    test_choose():
        t = new Table()
        assert raises error:
            t.choose(“aaa”)
        assert raises error:
            t.choose(“a”)
        assert raises error:
            t.choose(“”)

        t.add(“aa”, char(“a”))
        assert t.choose(“aa”) == char(“a”)
        assert t.choose(“ab”) == null
```

CLI

```
$ cat all_d.txt
dddddddd
$ java WordGen all_d.txt
dd→d:100%
dddddddd
$ cat alternate.txt
abab
$ java WordGen alternate.txt ba 8
ab→a:100%
ba→b:100%
babababa
$ cat other_options.txt
aabaac
$ java WordGen other_options.txt
aa→b:50%,c:50%
ab→a:100%
ba→a:100%
aac
```