HW1-prob4_dataAnalytics2

March 12, 2015

```
In [56]: import re
         import numpy as np
         import random
         from __future__ import division

         class BaseMarkovModel(object):
             def __init__(self, order=1):
                 """
                 Base Markov Sequence Model
                 params:
                 order - Number of history to use
                 """
                 self.order = order

             def learn(self, data):
                 """
                 Learn Markov Sequence Model for the given Data.
                 params:
                 data - array or any iterator of states
                 """

                 self.conditional_prob = self._calc_cpd(data)
                 return self

             def _calc_cpd(self, data):
                 """
                 Calculate the conditional probability .
                 Returns dict of type {prev_token : {next_token1: #freq,
                                       next_token2: #freq2}}
                 """

                 order = self.order
                 cpd = {}
                 for i in xrange(len(data) - order):
                     current_state = tuple(data[i : i + order])
                     next_state = data[i + order]
                     if current_state not in cpd:
                         cpd[current_state] = {}

                     if next_state not in cpd[current_state]:
                         cpd[current_state][next_state] = 0

                     cpd[current_state][next_state] += 1
```

```
            for i in cpd:
                marginal = sum(cpd[i].values())
                for l in cpd[i]:
                    cpd[i][l] = cpd[i][l] / marginal

            return cpd


        def walk(self, runlength=100):
            """
            Perform a random walk of length given by runlength
            """
            init_state = random.choice(self.conditional_prob.keys())
            gen_states = list(init_state)
            for i in xrange(runlength -1):
                next_state = self._choose_next(self.conditional_prob[init_state])
                gen_states.append(next_state)
                init_state = init_state[1:] + (next_state, )
            return " ".join(gen_states)

        def _choose_next(self,  probdist):
            """
            Perform multinomial sampling based on given probdist
            """
            states, pval = zip(*probdist.items())
            s = np.where(np.random.multinomial(1, pval, size=1)[0])[0]
            return states[s]

    class DataHandler(object):
        def get_states(self, fname):
            """
            Return lost of states in data
            params:
            fp - filepointer
            """
            if isinstance(fname, basestring):
                with open(fname) as fp:
                    states = self._tokenize(fp.read())
            else:
                states = self._tokenize(fname.read())
            return states

        def _tokenize(self, ss):
            """
            TODO: Implement nltk tokenizer and stemming and stop word removal
            """
            return re.split("\s+", ss)
```

```
In [60]: with open("/home/vicky/Downloads/conanDoyle.txt") as infile:
             t = DataHandler().get_states(infile)
         print len(t)
         model = BaseMarkovModel(order=2).learn(t)
         model.walk(100)
         with open("conan_gen.txt", "w") as out:
             out.write(model.walk(1000))
```

```
107534
```

```
Out[60]: 'your reasons," I remarked, "the thing always appears to tell heavily against our home product
```

```
In [63]: with open("/home/vicky/Downloads/janeAustent.txt") as infile:
             t = DataHandler().get_states(infile)
         print len(t)
         model = BaseMarkovModel(order=2).learn(t)
         print model.walk(100)

         with open("austen_gen.txt", "w") as out:
             out.write(model.walk(1000))
```

```
780226
```

```
Out[63]: "reasonable to expect that simply growing older ten years older than myself which makes celiba
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```