hangman

October 9, 2017

```
In [4]: with open('hw1_word_counts_05.txt') as f:
            content=f.readlines()
        words, word_freq = zip(*(s.split(" ") for s in content))

In [5]: word_freq = [freq.strip() for freq in word_freq]

In [6]: word_freq = map(float, word_freq)

In [7]: words[:6]

Out[7]: ('AARON', 'ABABA', 'ABACK', 'ABATE', 'ABBAS', 'ABBEY')

In [8]: word_freq[:6]

Out[8]: [413.0, 199.0, 64.0, 69.0, 290.0, 213.0]

In [9]: prior_pblty = [freq/sum(word_freq) for freq in word_freq]

In [10]: prior_pblty[:6]

Out[10]: [5.3882283779071155e-05,
          2.59626500533539e-05,
          8.349797002083667e-06,
          9.002124892871452e-06,
          3.783501766569161e-05,
          2.77891681475597e-05]

In [11]: corpus_dict = dict(zip(words, prior_pblty))

In [12]: len(corpus_dict)

Out[12]: 6535

In [13]: sorted(corpus_dict, key=corpus_dict.get, reverse=True)[:15]

Out[13]: ['THREE',
          'SEVEN',
          'EIGHT',
          'WOULD',
```

```
 'ABOUT',
 'THEIR',
 'WHICH',
 'AFTER',
 'FIRST',
 'FIFTY',
 'OTHER',
 'FORTY',
 'YEARS',
 'THERE',
 'SIXTY']
```

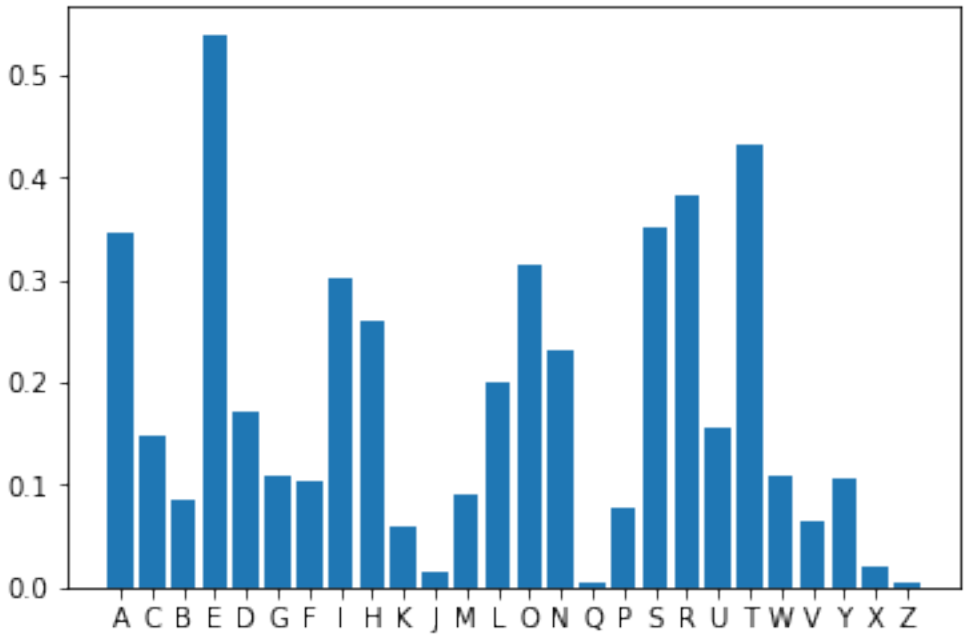
```
In [14]: sorted(corpus_dict, key=corpus_dict.get, reverse=False)[:14]
```

```
Out[14]: ['TROUP',
 'MAPCO',
 'CAIXA',
 'OTTIS',
 'BOSAK',
 'NIAID',
 'YALOM',
 'SERNA',
 'CLEFT',
 'CCAIR',
 'FOAMY',
 'PAXON',
 'TOCOR',
 'FABRI']
```

```
In [15]: import operator
         import re
         import matplotlib.pylab as plt

         def get_pattern(word_regex, incorrect_guess):
             regex_match_all='[ABCDEFGHIJKLMNOPQRSTUVWXYZ]'
             exception_string = word_regex.replace(".","")
             exception_string += ''.join(incorrect_guess)
             for exception_letter in exception_string:
                 regex_match_all = regex_match_all.replace(exception_letter,"")
             word_regex = word_regex.replace(".",regex_match_all)
             return word_regex

         def next_guess(word_regex, incorrect_guess=[]):
             pd=0.0
             evidence_pattern = re.compile(get_pattern(word_regex, incorrect_guess))
             # calculate once and there's no need to calculate again.
             # It's the heaviest calculation.
             for pd_word,pd_prob in corpus_dict.iteritems():
```

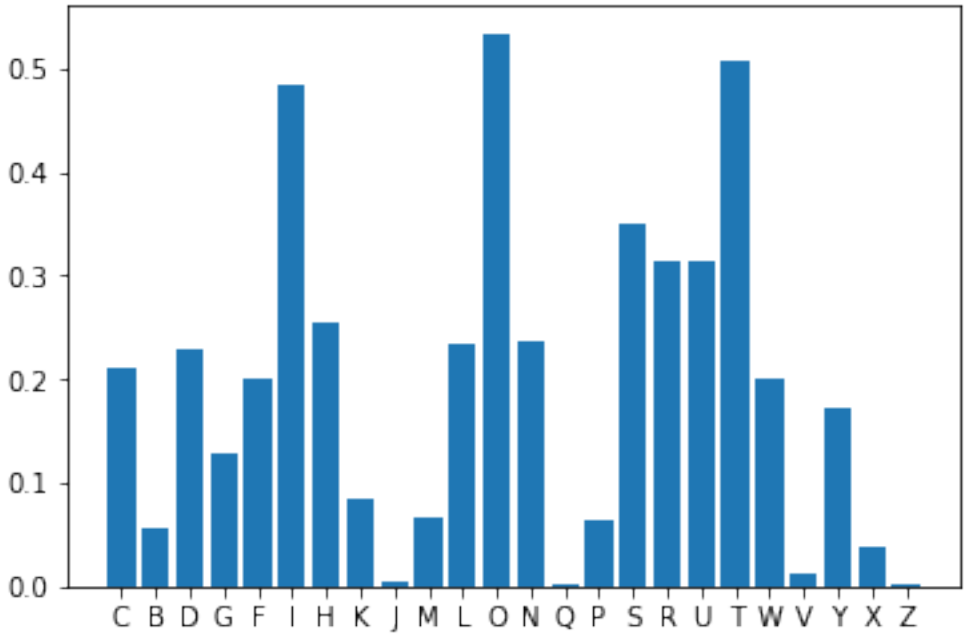
```
            pd = pd + (pd_prob if evidence_pattern.match(pd_word) else 0)
        next_guess_prob = {}
        all_letters='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
        exclude = word_regex.replace(".","")
        exclude += ''.join(incorrect_guess)
        for exception_letter in exclude:
            all_letters = all_letters.replace(exception_letter,"")
        for l in all_letters:
            sum_prob = 0;
            for w,prob in corpus_dict.iteritems():
                if l in w:
                    pn = corpus_dict.get(w) if evidence_pattern.match(w) else 0
                    sum_prob = sum_prob + (pn/pd)
            next_guess_prob[l] = sum_prob
        plt.bar(range(len(next_guess_prob)), next_guess_prob.values(), align='center')
        plt.xticks(range(len(next_guess_prob)), next_guess_prob.keys())
        plt.show()
        next_guess = max(next_guess_prob.iteritems(), key=operator.itemgetter(1))[0]
        return (next_guess, next_guess_prob[next_guess])
```

In [16]:
```
next_guess(".....")
```



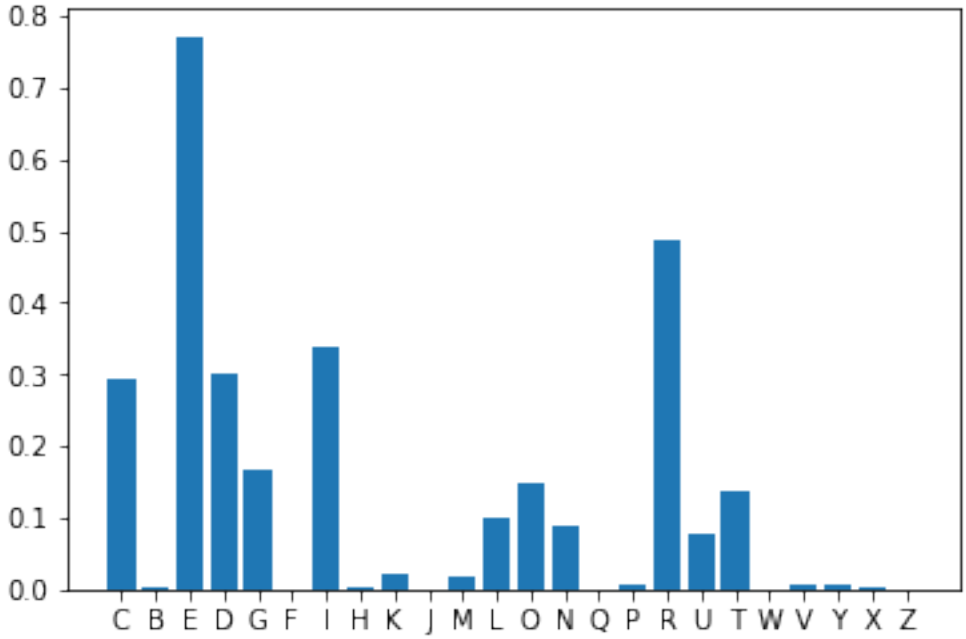
Out[16]:
```
('E', 0.5394172389647943)
```

In [17]:
```
next_guess(".....",['E','A'])
```



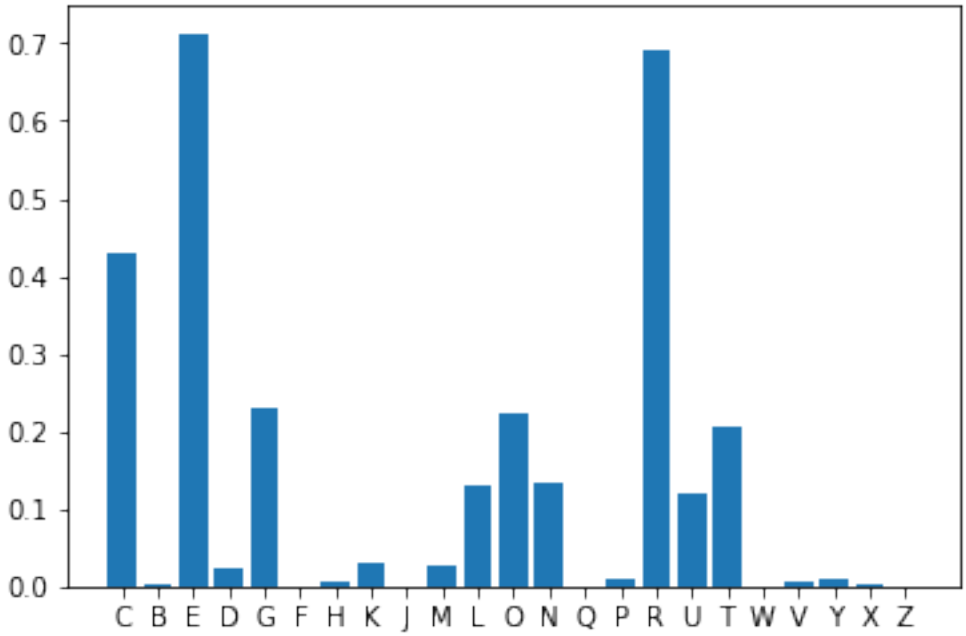
Out[17]: ('O', 0.5340315651557652)

In [18]: next_guess("A...S")



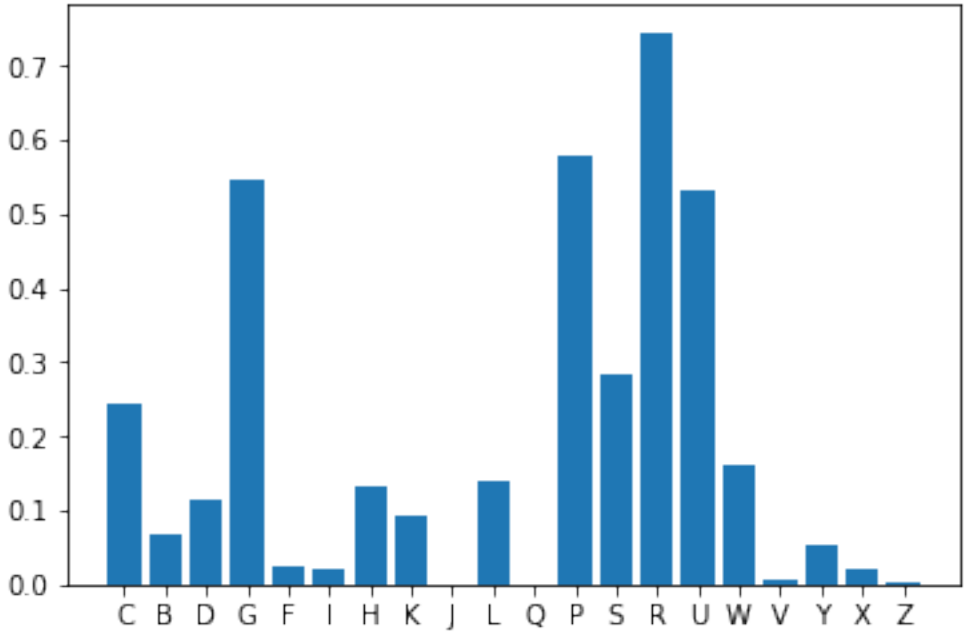
```
Out[18]: ('E', 0.7715371621621622)
```

```
In [19]: next_guess("A...S",['I'])
```

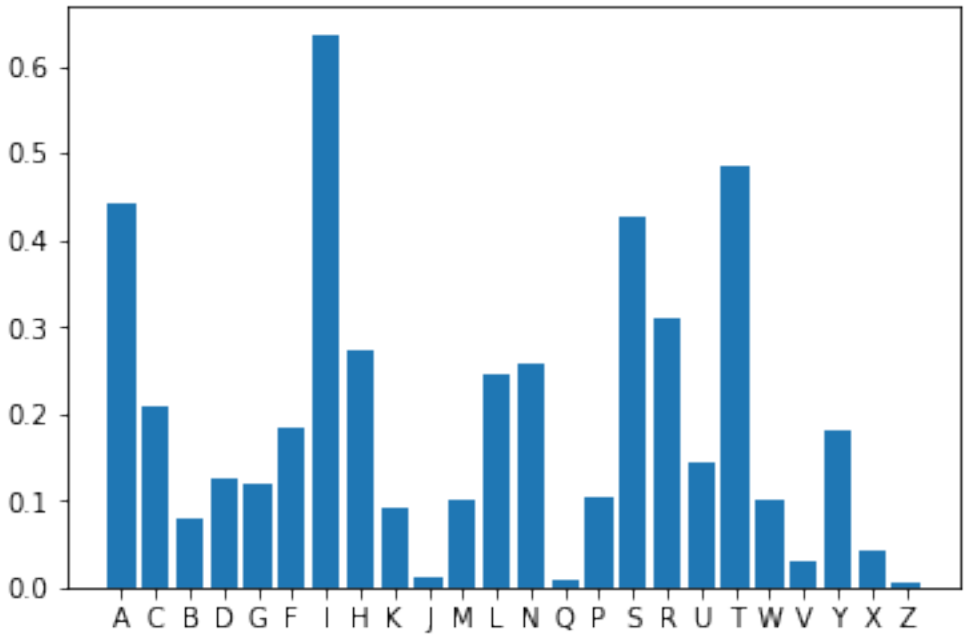



```
Out[19]: ('E', 0.7127008416220354)
```

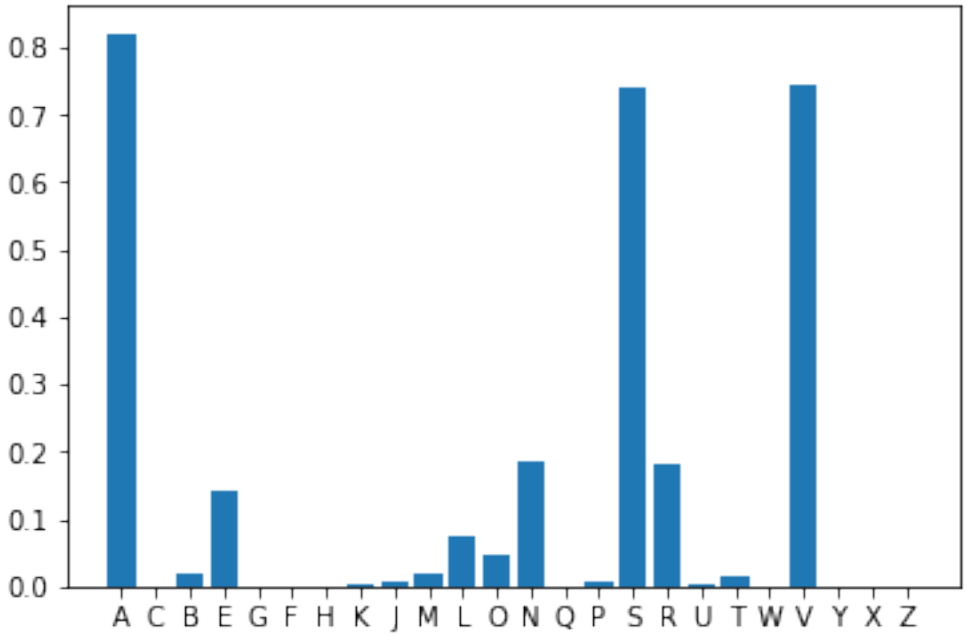
```
In [20]: next_guess("..O..",['A','E','M','N','T'])
```



```
Out[20]: ('R', 0.7453866259829716)

In [21]: ##############################################################################################
         #                                        VALIDATION HERE                                     #
         ##############################################################################################

In [22]: next_guess(".....",['E','O'])
```

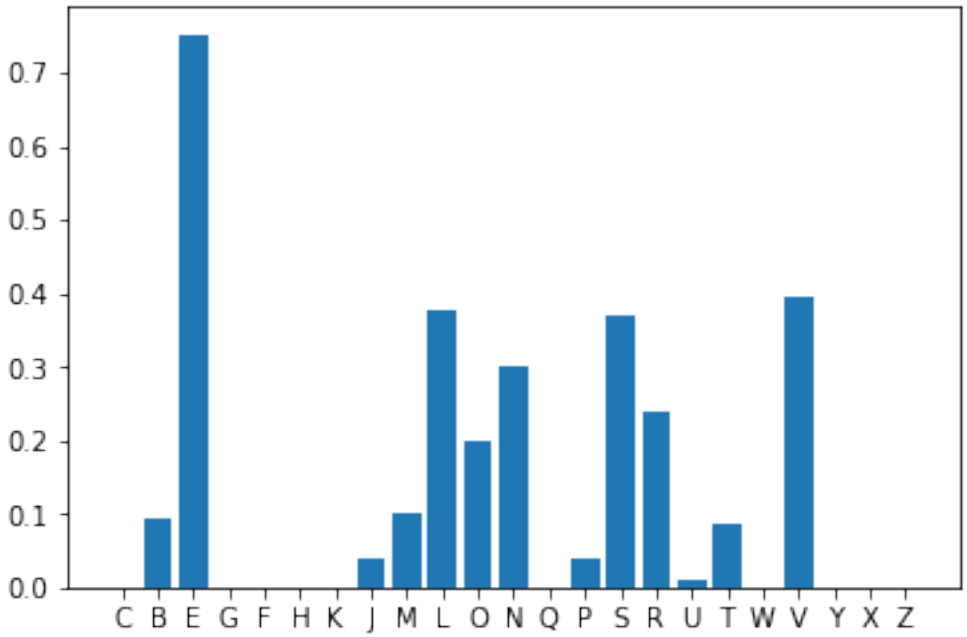



```
Out[22]: ('I', 0.6365554141009607)

In [23]: next_guess("D..I.")
```



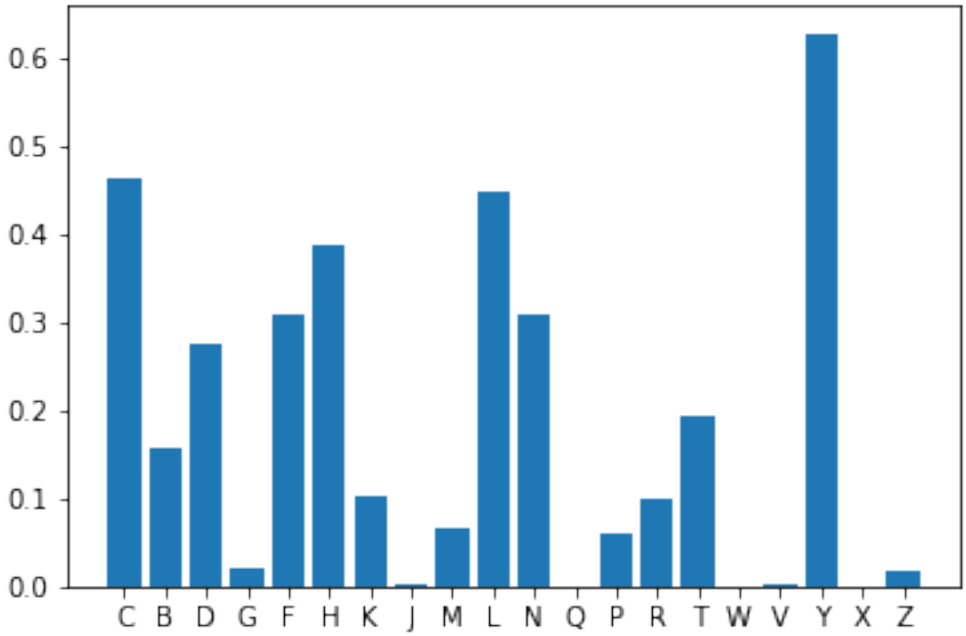

```
Out[23]: ('A', 0.8206845238095238)
```

```
In [24]: next_guess("D..I.",['A'])
```



```
Out[24]: ('E', 0.7520746887966804)
```

```
In [25]: next_guess(".U...",['A','E','I','O','S'])
```




```
Out[25]: ('Y', 0.6269651101630525)
```