

# **Assignment 3**

**Fall 2016**

**CS834 Introduction to Information Retrieval**

**Dr. Michael Nelson**

Mathew Chaney

November 6, 2016

## Contents

<b>1</b>	<b>Question 6.1</b>	<b>3</b>
1.1	Question . . . . .	3
1.2	Approach . . . . .	3
<b>2</b>	<b>Question 6.2</b>	<b>4</b>
2.1	Question . . . . .	4
2.2	Approach . . . . .	4
2.3	Results . . . . .	4
<b>3</b>	<b>Appendix</b>	<b>5</b>
<b>4</b>	<b>References</b>	<b>7</b>

## List of Figures

## Listings

1	spelling.py example output . . . . .	4
2	spelling.py . . . . .	5
3	stem.py . . . . .	6

## List of Tables

# 1 Question 6.1

## 1.1 Question

Using the Wikipedia collection provided at the book website, create a sample of stem clusters by the following process:

1. Index the collection without stemming.
2. Identify the first 1,000 words (in alphabetical order) in the index.
3. Create stem classes by stemming these 1,000 words and recording which words become the same stem.
4. Compute association measures (Dice's coefficient) between all pairs of stems in each stem class. Compute co-occurrence at the document level.
5. Create stem clusters by thresholding the association measure. All terms that are still connected to each other form the clusters.

Compare the stem clusters to the stem classes in terms of size and the quality (in your opinion) of the groupings.

## 1.2 Approach

The `stem.py` script, found in Listing 3, was used to solve this problem.

## 2 Question 6.2

### 2.1 Question

Create a simple spelling corrector based on the noisy channel model. Use a single-word language model, and an error model where all errors with the same edit distance have the same probability. Only consider edit distances of 1 or 2. Implement your own edit distance calculator (example code can easily be found on the Web)

### 2.2 Approach

The `spelling.py` script, found in Listing 2, was created using the Python programming language [1]. Downloaded `big.txt` to calculate an example language model. These words were counted and stored in a map that was compressed on disk using the pickle python library [2].

The process of determining a spelling correction is as follows:

1. Count all the words contained in the example text file. This count is used to determine the language model  $P(W)$  calculation.
2. Take the input word and determine all existing (correctly spelled) words with edit distance one and two.
3. With the assumption that shorter edit distances equate to a higher probability of being the correct spelling, select from the remaining set of (valid) words the one with the shortest edit distance and highest value for  $P(W)$ .

$P(W)$  is calculated with the following formula:

$$P(W) = \frac{C_W}{N}$$

where  $C_W$  is the word count for word  $W$  and  $N$  is the sum of all word counts.

### 2.3 Results

Here is some sample output from the `spelling.py` script.

```
1 [mchaney@mchaney-1 spelling]$ ./spelling.py werds
2 words
3 [mchaney@mchaney-1 spelling]$ ./spelling.py fleworz
4 flower
5 [mchaney@mchaney-1 spelling]$ ./spelling.py splling
6 selling
7 [mchaney@mchaney-1 spelling]$ ./spelling.py sweling
8 swelling
9 [mchaney@mchaney-1 spelling]$ ./spelling.py aacck
10 back
```

Listing 1: `spelling.py` example output

## 3 Appendix

```
1 #!/usr/bin/env python
2
3 import re
4 import sys
5 import cPickle
6 from collections import Counter
7
8
9 def get_words():
10     try:
11         return cPickle.load(open('words.p', 'rb'))
12     except IOError:
13         wordmap = Counter(re.findall(r'\w+', open('big.txt').read().lower()))
14         cPickle.dump(wordmap, open('words.p', 'wb'))
15         return cPickle.load(open('words.p', 'rb'))
16
17 words = get_words()
18 N = sum(words.values())
19
20
21
22 def exists(wordset):
23     return set([word for word in wordset if word in words])
24
25
26 def prob(word):
27     return float(words[word]) / float(N)
28
29
30 def edit1(w):
31     letters = 'abcdefghijklmnopqrstuvwxyz'
32     deletes = [w[:i]+w[i+1:] for i in range(len(w))]
33     transposes = [w[:i]+w[i+1]+w[i]+w[i+2:] for i in range(len(w)-1)]
34     replaces = [w[:i]+l+w[i+1:] for i in range(len(w)) for l in letters]
35     inserts = [w[:i]+l+w[i:] for i in range(len(w)+1) for l in letters]
36     return set(deletes + transposes + replaces + inserts)
37
38
39 def edit2(word):
40     e2 = [edit1(w) for w in edit1(word)]
41     return [item for sublist in e2 for item in sublist]
42
43
44 def parse(word):
45     return exists([word]) or exists(edit1(word)) or exists(edit2(word)) or [word]
46
47
48 def correct(word):
49     return max(parse(word), key=prob)
50
51
52 if __name__ == '__main__':
53     print correct(sys.argv[1])
```

Listing 2: spelling.py

```
1 #!/usr/bin/env python
```

Listing 3: stem.py

## 4 References

- [1] The Python Programming Language. Available at: <https://www.python.org/>. Accessed: 2016/09/17.
- [2] Python.org. Python object serialization. Available at: <https://docs.python.org/2/library/pickle.html>. Accessed: 2016/11/06.