INLS 609-270 - Assignment 1

You name: _Jenna Gatley

* Please submit your answers as a PDF attachment via Canvas. Include your data and code as a separate zip file or pdf attachment.

* General discussion is allowed, but you should compose with your solution **independently**.

Question 1: Word Association (30 points)

Implement a program that finds the top 100 word associations in a given collection of Amazon reviews measured by pointwise mutual information we introduced in class. You may want to (1) remove some punctuations (e.g., comma, quotation marks, and so on); (2) consider only those ordered word pairs that appear within a fixed size of text windows (e.g., window size = 5 consecutive words) for at least a certain number of times (e.g., 50 times). You might want to start by taking the basic statistics, such as the frequency of each word (number of reviews that a word appeared in) and the frequency of each word pair (number of reviews a pair of consecutive words appeared in). Examine the word associations you've found. Do they all make sense? Can you see something interesting or undecipherable?

Most of the word pairings make sense but I found two pairs of interest. The specific pair of ('e', 'g') with a pmi of 11.035449516180822 was odd. I did not know what could have made it so these letters were separate and found together several times. Another interesting pair that made sense was ('yourself', 'favor') with a pmi of 11.443661882245157. It clearly seems to have found a correlation of the possible sentence 'do yourself a favor' but I would not have paired these words together on my own. Most of the other results seem intuitive but this find was interesting!

Question 2: Feature Selection (30 points)

Implement a program that finds 100 single words (so-called "unigrams") that are most associated with sentiment labels (label = 1: positive; label = 0: negative) in the given collection of Amazon reviews using Chi-square (χ^2) we introduced in class. Examine the words you've found. For each of these words, can you tell which sentiment (positive or negative) this word is strongly associated with? Do you see "mysterious" words that do not clearly associate with positive or negative sentiment?

For all but three words, it is pretty clear whether it has a positive or negative sentiment. When it comes to terms like trash, wasted, or ridiculous, I have no trouble identifying the negative association. The three words that showed up in the top 100 that were confusing are, “Would”, “Did”, and “same”. They could go either way yet came up within the top 100 list. I imagine that they may be all related to negative as they may relate to a customer’s expectations or previous experiences not being fulfilling.

Question 3: Spell Correction (40 points)

Spelling correction is a common functionality provided by most search engines. The basic idea can be simplified as matching an out-of-vocabulary string to a word in vocabulary that is the closest in spelling (for example, "carrolina" -> " carolina"). In other words, we need a function that measures the similarity between two strings. Of course, for those who knew it, a natural measure of string similarity/distance is the Levenshtein Edit Distance (http://en.wikipedia.org/wiki/Levenshtein_distance). You can find lots of implementations at http://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Levenshtein_distance. The problem in practice, however, is that the computation of edit distance is very costly, especially for long strings. Computationally it takes O(m*n) to find the edit distance between two strings with m and n letters (which can be pretty ugly when m and n are large). Spelling something like "*Parastratiosphecomyia sphecomyioides*" correctly is a challenge, and figuring out "*Parastratioschecomia*" responds to "*Parastratiosphecomyia*" isn't easier.

Alternatively, if we make use of a smart representation of a word/string, we can potentially reduce the time complexity with an approximation. One such approximation is to break a long string into **overlapping tri-grams**. That means you can represent a string with a **set** of tri-grams. That says, "*parastratioschecomia*" becomes {par, ara, ast, str, tra, rat, ati, tio, ios, osc, sch, che, hec, eco, com, omi, mia}, and "*Parastratiosphecomyia*" becomes {par, ara, ast, str, tra, rat, ati, tio, ios, osp, sph, phe, hec, eco, com, omy, myi, yia}. If we quickly compare the two sets, we can see they are highly similar, high enough to draw the conclusion that the two original strings are similar. Even sweeter, it takes a time complexity of O(m+n) to compute the similarity of two sets, instead of O(m*n).

You are provided with a dictionary (all words starting with "a" in wiktionary). Please implement a function so that for any input string, you can return the top 10 words in the dictionary that are the most similar to it. Use the Jaccard similarity that we introduced in class. Include the results for the following 5 strings:

abreviation

abstrictiveness

accanthopterigious

artifitial inteligwnse

agglumetation

Compare the results of your implementation with the results generated by the edit distance (you may use an existing implementation or package). Are you getting a good approximation?

One way to further tune the itemset representation of strings is to vary the length of the "*n*-grams" in your set. As you might already have guessed, an *n*-gram means *n* consecutive letters in a string. Try to use overlapping bigrams (2-grams), 4-grams, and 5-grams instead of tri-grams, and compare the results with the results using tri-grams. Which length of *n*-gram seems to work best for this task?

The Jaccard similarity and edit distance results seem to both have good approximations but the Jaccard results win out when it comes to the top words and their similarity. The edit distance results can vary pretty aggressively in similarity while the Jaccard words seem to be far more accurate letter wise.

I also tested the different n-grams and found that the bi-grams had better results. It may be due to the more detailed word breakdown that increased the accuracy. The 4 and 5-grams were not ideal in their accuracy results as it had to stick closer to the larger chunks for comparison. Based off of the suggested words and similarity statistics, I would say the bi-grams would work best for this task. They created the most accurate top ten words.

My Tri-gram results:

Input String: abreviation

Top 10 similar words using Jaccard similarity (tri-grams):

```
[('abbreviation', 0.7272727272727273), ('abbreviations', 0.6666666666666666), ('abbreviationi',
0.6666666666666666), ('abbreviatione', 0.6666666666666666), ('adbreviationi',
0.6666666666666666), ('adbreviatione', 0.6666666666666666), ('abbreviatio',
0.6363636363636364), ('adbreviatio', 0.6363636363636364), ('abreviativo',
0.6363636363636364), ('abreviativa', 0.6363636363636364)]
```

Top 10 similar words using edit distance:

```
[('abbreviation', 1), ('abbreviatio', 2), ('abbreviations', 2), ('alleviation', 2), ('abbreviationi', 2),
('abbreviatione', 2), ('abreviaron', 2), ('adbreviatio', 2), ('adbreviationi', 2), ('adbreviatione', 2)]
```

Input String: abstrictiveness

Top 10 similar words using Jaccard similarity (tri-grams):

```
[('abstractiveness', 0.625), ('activeness', 0.5), ('addictiveness', 0.5), ('astrictive', 0.5), ('abstriction',
0.4666666666666667), ('abstricting', 0.4666666666666667), ('astrictives', 0.4666666666666667),
('abstrict', 0.46153846153846156), ('abstrictions', 0.4375), ('activenesses', 0.4375)]
```

Top 10 similar words using edit distance:

```
[('abstractiveness', 1), ('absorptiveness', 3), ('attractiveness', 3), ('abortiveness', 4), ('abstersiveness', 4), ('abstractedness', 4), ('abstractness', 4), ('assertiveness', 4), ('abstrictions', 4), ('attributiveness', 4)]


Input String: accanthopterigious

Top 10 similar words using Jaccard similarity (tri-grams):

[('acanthopterygious', 0.55), ('acanthopteri', 0.5294117647058824), ('acanthopteri', 0.5294117647058824), ('acanthopterous', 0.47368421052631576), ('acanthopteran', 0.42105263157894735), ('acanthopterans', 0.4), ('acanthopterygii', 0.38095238095238093), ('acanthopterygian', 0.36363636363636365), ('acanthopt', 0.35294117647058826), ('acanthopterygians', 0.34782608695652173)]

Top 10 similar words using edit distance:

[('acanthopterygious', 2), ('acanthopterous', 4), ('acanthopterygians', 4), ('acanthopterygian', 5), ('acanthopterygii', 5), ('acanthopodious', 6), ('acanthophorous', 6), ('acanthopteri', 6), ('acanthopteri', 6), ('acanthopterans', 6)]


Input String: artifitial inteligwnse

Top 10 similar words using Jaccard similarity (tri-grams):

[('artificial intelligence', 0.41379310344827586), ('artificial intelligences', 0.4), ('artificial insemination', 0.28125), ('artificial art', 0.24), ('artificial anus', 0.2222222222222222), ('artificial life', 0.2222222222222222), ('artificialmente', 0.2222222222222222), ('artificialities', 0.2222222222222222), ('artificial turf', 0.2222222222222222), ('artificial', 0.21739130434782608)]

Top 10 similar words using edit distance:

[('artificial intelligence', 4), ('artificial intelligences', 5), ('artificial life', 9), ('artificial insemination', 9), ('artificialities', 9), ('artificial horizons', 9), ('artificial persons', 9), ('artificial abortions', 9), ('artificial language', 10), ('adaptive intelligence', 10)]


Input String: agglumetation

Top 10 similar words using Jaccard similarity (tri-grams):

[('agglutination', 0.375), ('arietation', 0.35714285714285715), ('agglutinations', 0.35294117647058826), ('arietationi', 0.3333333333333333), ('arietatione',
```

0.3333333333333333), ('arietations', 0.3333333333333333), ('arietationes', 0.3125), ('arietationis', 0.3125), ('arietationum', 0.3125), ('arietationem', 0.3125)]

Top 10 similar words using edit distance:

[('agglomeration', 2), ('agglomeration', 2), ('agglutination', 3), ('agglomerations', 3), ('argumentation', 3), ('argumentation', 3), ('agglomeratioun', 3), ('acclimatation', 4), ('augmentation', 4), ('aggregation', 4)]

BIGRAMS TEST

Input String: abreviation

Top 10 similar words using Jaccard similarity (Bi-grams):

[('abbreviation', 0.9090909090909091), ('abbreviations', 0.8333333333333334), ('abbreviationi', 0.8333333333333334), ('abbreviatione', 0.8333333333333334), ('abbreviatio', 0.8181818181818182), ('abbreviationis', 0.7692307692307693), ('abbreviationem', 0.7692307692307693), ('abbreviationes', 0.7692307692307693), ('abbreviationum', 0.7692307692307693), ('abbreviati', 0.7272727272727273)]

Top 10 similar words using edit distance:

[('abbreviation', 1), ('abbreviatio', 2), ('abbreviations', 2), ('alleviation', 2), ('abbreviationi', 2), ('abbreviatione', 2), ('abreviaron', 2), ('adbreviatio', 2), ('adbreviationi', 2), ('adbreviatione', 2)]

Input String: abstrictiveness

Top 10 similar words using Jaccard similarity (Bi-grams):

[('abstractiveness', 0.75), ('astrictives', 0.6), ('activeness', 0.5333333333333333), ('abstivesen', 0.5333333333333333), ('astrictive', 0.5333333333333333), ('abstivesse', 0.5333333333333333), ('abstivesses', 0.5333333333333333), ('addictiveness', 0.5294117647058824), ('absorptiveness', 0.5), ('abstersiveness', 0.5)]

Top 10 similar words using edit distance:

[('abstractiveness', 1), ('absorptiveness', 3), ('attractiveness', 3), ('abortiveness', 4), ('abstersiveness', 4), ('abstractedness', 4), ('abstractness', 4), ('assertiveness', 4), ('abstrictions', 4), ('attributiveness', 4)]

Input String: accanthopterigious

Top 10 similar words using Jaccard similarity (Bi-grams):

```
[('acanthopterygious', 0.7368421052631579), ('acanthopterous', 0.6666666666666666),
('acanthopteri', 0.6470588235294118), ('acanthopteri', 0.6470588235294118), ('acanthopteran',
0.5555555555555556), ('acanthopterygian', 0.55), ('acanthopterygii', 0.55), ('acanthopterans',
0.5263157894736842), ('acanthopterygians', 0.5238095238095238), ('acanthopodious', 0.5)]

Top 10 similar words using edit distance:

[('acanthopterygious', 2), ('acanthopterous', 4), ('acanthopterygians', 4), ('acanthopterygian', 5),
('acanthopterygii', 5), ('acanthopodious', 6), ('acanthophorous', 6), ('acanthopteri', 6),
('acanthopteri', 6), ('acanthopterans', 6)]


Input String: artifitial inteligwnse

Top 10 similar words using Jaccard similarity (Bi-grams):

[('artificial intelligence', 0.5555555555555556), ('artificial intelligences', 0.5357142857142857),
('artificial insemination', 0.42857142857142855), ('artificiality', 0.391304347826087), ('artificial
life', 0.375), ('artificialities', 0.375), ('artificiali', 0.36363636363636365), ('artigliante',
0.36363636363636365), ('artificialia', 0.36363636363636365), ('artificialmente', 0.36)]

Top 10 similar words using edit distance:

[('artificial intelligence', 4), ('artificial intelligences', 5), ('artificial life', 9), ('artificial
insemination', 9), ('artificialities', 9), ('artificial horizons', 9), ('artificial persons', 9), ('artificial
abortions', 9), ('artificial language', 10), ('adaptive intelligence', 10)]


Input String: agglumetation

Top 10 similar words using Jaccard similarity (Bi-grams):

[('agglutination', 0.5333333333333333), ('agglomeration', 0.5), ('agglutinations', 0.5),
('agglomeration', 0.5), ('agglomerations', 0.47058823529411764), ('agglutinatarum',
0.47058823529411764), ('agglutinata', 0.4666666666666667), ('agitationum',
0.4666666666666667), ('agglomerationen', 0.4444444444444444), ('autoagglutination',
0.4444444444444444)]

Top 10 similar words using edit distance:

[('agglomeration', 2), ('agglomeration', 2), ('agglutination', 3), ('agglomerations', 3),
('argumentation', 3), ('argumentation', 3), ('agglomeratioun', 3), ('acclimatation', 4),
('augmentation', 4), ('aggregation', 4)]
```