Question we investigated
Are there systematic differences between similarity ranking measures?

Method
We implemented four similarity measures: cosine similarity, cosine similarity with $PPMI_{alpha}$, Jaccard index, and Sørensen–Dice coefficient (Dice coefficient).

The $PPMI_{alpha}$ function was implemented based on Jurafsky & Martin (3rd Edition Draft) discussion of positive pointwise mutual information (PPMI). They observe that PPMI tends to be biased. Infrequent words may skew the rankings. To reduce this effect, the word counts are mitigated with an alpha value. We used the suggested value of 0.75 (Jurafsky and Martin, 2016)(Levy et. al, 2015).

The Jaccard index and the Dice coefficient are strongly similar indexing measures that use weighted vector proportions to create a similarity ratio. Our Jaccard index is calculate in a set-wise manner without regard for word counts. Where two words share common vectors the cardinality of the intersection is divided by the cardinality of the union of the vectors.

With the Dice coefficient, we similarly took the measure over the sets of vectors without regard for the magnitude of word counts. It is calculated as twice the cardinality of the overlapping vectors and divide by the sum of the cardinality of the two vectors. Both of these measures as implemented tend to be independent of the frequency of the terms' co-occurrence.

We arbitrarily selected a set of words we thought might have high to intermediate similarity {computer, pc, mac, machine, mouse, phone, apple} and a set of words that we thought would have some lack of similarity {cat, shovel, car, #egypt, trump, easy}.

For each set of words and for each indexing method, we sorted the word pairs by the calculated index value to get a ranked order of words.

Results
When we examined our similar words, the rankings by similarity had different orderings (except between Jaccard and Dice). The top six results for our similar and different words sets are in the appended material.

While observing the output result, we defined a new measure metric called "absolute proportional word count difference", which is calculated as the difference between two counts divided by the sum of their counts. Although not 100%, it could be noticed that there is a general trend that the word pairs with large similarity may have less absolute proportional difference.

In our set of similar seeming words, the order of the words differs between the cosine indexing methods and the Jaccard/Dice indexing methods. The differences appear to be caused by different proportions of word counts.

When we examined our dissimilar words, we saw a larger change in pair similarity measures. The $PPMI_{alpha}$ index more stongly suggests the lack of similarity between the words we selected to be dissimilar. The Jaccard index, as implemented, ignores word counts so its numbers may be sufficient for some kind of rank but probably not sufficient for qualitative statements.

Discussion and future directions
From our investigations, it appears that for ranking of distributional similarity a measure that uses word counts will yield results that seem intuitively plausible. The $PPMI_{alpha}$ measure provides more zeros for strongly dissimilar words; it does not promote words without a weight of word counts to

support it. One draw-back is that the alpha requires tuning for application-specific results. This has not been invested in this project.

The Jaccard and Dice measures we used do not take into account the word counts of the vector elements that appear, just the count of intersecting, co-occurrent words. We believe that the adjustment suggested by Jurafsky and Martin (Jurafsky 2016) would take into account word frequencies may make the importance of common occurrence word pairs seem more distributionally similar.

One difference we noticed between the all the measures we tested was the differing slopes in the curves of the ordered results. The cosine similarity measures appear to be almost logarithmic while the Jaccard and Dice measures are closer to linear.

It may actually be more useful with Twitter analysis to not process out any small word counts nor to harshly discount them, since Twitter diction and grammar often contain multiple spellings (misspellings), and invented words specific to an event.

Preliminary Task Output

```
PMI check passed
Getting all the word id's as a set
Creating word vectors...
Sort by cosine similarity
0.36    ('cat', 'dog')                  169733   287114
0.17    ('comput', 'mous')              160828   22265
0.12    ('cat', 'mous')                 169733   22265
0.09    ('mous', 'dog')                 22265    287114
0.07    ('cat', 'comput')               169733   160828
0.06    ('comput', 'dog')               160828   287114
0.02    ('@justinbieber', 'dog')        703307   287114
0.01    ('cat', '@justinbieber')        169733   703307
0.01    ('@justinbieber', 'comput')     703307   160828
0.01    ('@justinbieber', 'mous')       703307   22265
```

Results

Top six similarity pair-wise rankings of the set {computer, pc, mac, machine, mouse, phone, apple}

Cosine similarity with PPMI		**Cosine similarity with $PPMI_{alpha}$**		**Jaccard index**		**Dice coefficient**	
0.40	('comput', 'pc')	0.26	('comput', 'pc')	0.50	('comput', 'mac')	0.67	('comput', 'mac')
0.29	('pc', 'mac')	0.12	('pc', 'mac')	0.50	('mac', 'machin')	0.67	('mac', 'machin')
0.27	('comput', 'mac')	0.08	('pc', 'mous')	0.49	('pc', 'mac')	0.66	('pc', 'mac')
0.20	('comput', 'machin')	0.08	('comput', 'mac')	0.48	('comput', 'machin')	0.65	('comput', 'machin')
0.18	('pc', 'machin')	0.07	('mous', 'mac')	0.47	('comput', 'pc')	0.64	('comput', 'pc')
0.18	('pc', 'mous')	0.06	('comput', 'mous')	0.47	('pc', 'machin')	0.64	('pc', 'machin')

Top six similarity pair-wise rankings of the set {cat, shovel, car, #egypt, trump, easy}

Cosine similarity with PPMI		**Cosine similarity with $PPMI_{alpha}$**		**Jaccard index**		**Dice Coefficient**	
0.08	('cat', 'car')	0.01	('car', 'shovel')	0.51	('cat', 'easi')	0.68	('cat', 'easi')
0.08	('#egypt', 'trump')	0.01	('#egypt', 'trump')	0.44	('easi', 'car')	0.61	('easi', 'car')
0.08	('easi', 'car')	0.01	('trump', 'shovel')	0.41	('cat', 'car')	0.58	('cat', 'car')
0.06	('cat', 'easi')	0.00	('easi', 'car')	0.31	('#egypt', 'trump')	0.47	('#egypt', 'trump')
0.05	('cat', 'shovel')	0.00	('#egypt', 'shovel')	0.23	('cat', 'trump')	0.37	('cat', 'trump')
0.05	('car', 'shovel')	0.00	('easi', 'trump')	0.20	('easi', 'trump')	0.33	('easi', 'trump')

Sample similarity measures

Includes absolute proportional word count difference (APWCD)

)pmi_alpha)

Similarity	APWCD
0.26	0.44
0.12	0.20
0.08	0.47
0.08	0.26
0.07	0.62
0.06	0.76
0.02	0.86
0.02	0.95
0.02	0.06
0.02	0.39
0.01	0.80
0.01	0.68
0.01	0.85
0.01	0.14
0.01	0.52

Sorted by Jaccard similarity

Word Pairs	Similarity	APWCD
('mac', 'machin')	0.5	0.14
('comput', 'mac')	0.5	0.26
('pc', 'mac')	0.49	0.20
('comput', 'machin')	0.48	0.39
('pc', 'machin')	0.47	0.06
('comput', 'pc')	0.47	0.44
('mous', 'machin')	0.35	0.52
('comput', 'phone')	0.35	0.68
('pc', 'mous')	0.33	0.47
('mous', 'mac')	0.31	0.62
('phone', 'mac')	0.27	0.80
('comput', 'mous')	0.26	0.76
('phone', 'machin')	0.25	0.85
('pc', 'phone')	0.23	0.86
('mous', 'phone')	0.11	0.95

Bibliography

Daniel Jurafsky & James H. Martin. (2016). Speech and Language Processing. Draft of November 7, 2016. Section 15.2

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving distributional similarity with lessons learned from word embeddings. TACL, 3, 211–225.

http://dataconomy.com/implementing-the-five-most-popular-similarity-measures-in-python/