Exploring Distributional Similarity in Twitter

s1885540 s1838650

1 Introduction and Preliminary Task

In natural language processing, word meaning identification is a crucial task for many application such as machine translation or text to speech. In this report, we will explore the concept of Distributional Similarity: similar words tend to exist in similar contexts. For the preliminary task, we transform words to vectors with Positive Pointwise Mutual Information(PPMI) and compare using Cosine Similarity.
The cosine similarity scores for the default word pairs are shown in table 1:

Cosine Sim	Word Pair
0.36	('cat', 'dog')
0.17	('comput', 'mous')
0.12	('cat', 'mous')
0.09	('mous', 'dog')
0.07	('cat', 'comput')
0.06	('comput', 'dog')
0.02	('@justinbieber', 'dog')
0.01	('cat', '@justinbieber')
0.01	('@justinbieber', 'comput')
0.01	('@justinbieber', 'mous')

Table 1: Ranked List with Cosine Similarity Scores

1.1 Method Choosing

In addition to PPMI and Cosine similarities. We try t-test scores for vector computation, and Jaccard for similarity comparison. Here, we use generalized Jaccard similarity (a.k.a Ruzicka similarity [1]); if $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n)$ are two vectors with all real $x_i, y_i \geq 0$, then their generalized Jaccard similarity coefficient is defined as

$$J(\mathbf{x}, \mathbf{y}) = \frac{\sum_i min(x_i, y_i)}{\sum_i max(x_i, y_i)} \tag{1}$$

The association t-test for 2 word **a** and **b** as defined in [3] with p(a),p(b) equal to frequency of word **a**(count(a)/N) and word **b**(count(b)/N) respectively. p(a,b) is the frequency of both words **a, b** occur in a same Tweet. It will be interesting to compare t-test to PPMI as t-test range is [-1,1] while PPMI is in range [0,∞] :

$$t = \frac{p(a,b) - p(a)p(b)}{\sqrt{p(a)p(b)}} \tag{2}$$

1.2 Research Hypothesis

Our paper aim to provide a detailed comparison between methods for creating word vector: PPMI and t-test with respect to word frequency and try to investigate their correlation. Also, we will look at some particular word pairs to try understand how they rank word and their effectiveness. All methods will be analyzed on the Twitter dataset, which is a little portion of all tweets sent during 2011.

2 Word Choosing and Result

For this experiment, we choose this particular word set['hurt','die','@bieber','@jdbiebercrews', 'love','loov', '#bieberfact'], and the detail and result of word pairs are shown in table 2. ['@bieber','@jdbiebercrews','#bieberfact'] will be used for qualitative analysis during PPMI and t-test comparison. We will be using the set [love, loov, die, hurt] for comparing Cosine and Jaccard Similarity since it covers enough range of frequency from low count (loov: 3892) to very high count(love: 5178829). In addition, we expect that our methods can correctly identify that 'loov' is just a teen's spelling way for 'love' or 'die' and 'hurt' should be more similar than 'love' and 'hurt'. Besides the manually selected words, we randomly pick 100 word types and generate all possible word pairs for quantitative analysis during PPMI and t-test comparison.

Word 1	Count	Word 2	Count	PPMI-Cosine	PPMI-Jaccard	t-test-Cosine	t-test-Jaccard
@jdbiebercrews	548	@bieber	132	0.33	0.12	0.21	5.22
@bieber	132	#bieberfact	34901	0.04	0.00	0.30	-1.15
@jdbiebercrews	548	#bieberfact	34901	0.06	0.24	0.06	-1.20
love	5178829	loov	3892	0.05	0.0	0.20	-2.16
love	5178829	hurt	460885	0.18	0.06	0.20	-4.58
die	497828	hurt	460885	0.12	0.10	0.14	4.23
love	5178829	die	497828	0.08	0.04	0.10	-3.18
love	5178829	#bieberfact	34901	0.08	0.01	0.09	-1.98
die	497828	#bieberfact	34901	0.05	0.04	0.02	-2.34
loov	3892	@bieber	132	0.04	0.01	0.06	8.41
hurt	460885	#bieberfact	34901	0.03	0.03	0.01	-4.01
@jdbiebercrews	548	loov	3892	0.03	0.04	0.01	4.90
die	497828	loov	3892	0.01	0.00	-0.01	-3.07
hurt	460885	loov	3892	0.01	0.00	0.02	-6.18
@jdbiebercrews	548	die	497828	0.01	0.07	0.00	-2.51
love	5178829	@bieber	132	0.01	0.00	0.06	-2.01
@jdbiebercrews	548	hurt	460885	0.01	0.03	-0.01	-4.32

Table 2: Similarity Results

3 Analysis

3.1 Impact of frequency to vector computation: PPMI vs. t-test

We will analyze the impact of word frequency from qualitative and quantitative aspects. For qualitative analysis, we will discuss the behaviours of word pairs about Justin Bieber. For quantitative analysis, we will use correlation coefficient to measure the dependency of word similarity on word frequency.

3.1.1 Qualitative Analysis

To control variables, we fix cosine similarity and begin the following analysis.

Word	Top-5 Co-occurrence Words
#bieberfact	justin girl fan love @myfactbieber
@jdbiebercrews	give follow pin ill justin
@bieber	justin bieber follow dm #bieberfact

Table 3: Word and Its Top Co-occurrence Words

From table 3, we can find the top occurrence words of "@jdbiebercrews", "@bieber" and "#bieberfact" are all related to idolize Justin Bieber, such as "love", "fan", "follow", "#neversaynev", thus, they are supposed to be similar. However, from table 2. for word pairs ("@jdbiebercrews", "@bieber") and ("#bieberfact", "@bieber"), their cosine similarities are quite different (0.33 and 0.04 respectively) under PPMI computation. Since PPMI is based on word counts and is over sensitive to the infrequent words, and here the amount of "@jdbiebercrews" is almost two orders of magnitude less than that of "#bieberfact", which leads to inaccurate similarity computation. On the other hand, the cosine similarity scores are 0.21 and 0.30 respectively under t-test computation. This more reasonable result is benefited from the fact that t-test looks at the difference between the sample means and expected means over the variance of the data.

Tweets Related to @jdbiebercrews	Tweets Related to #bieberfact
RT @JDBieberCrews: #BieberFact: JB won Social Artist, Streaming Artist, Digital Media Artist, Pop Album, (cont) http://tl.gd/akrtd5	#bieberfact - if justin ever has kids, he said he wanted to have a boy first so that if he had a girl,the boy would protect her.
U lie justin not using bb anymore.RT @JDBieberCrews : retweet if you want justinbieber's pin bbm ! so i can give you ... http://tmi.me/aukPb	#bieberfact Justin said that if he was fighting with his girlfriend, and she started crying, he would shut up and hug her tight.

Table 4: 2 Random Tweets Related to @jdbiebercrews and #bieberfact

Now we take a deeper evaluation for the accuracy of similarity score under t-test computation: *Is "@bieber" more similar to "@jdbiebercrews" or "#bieberfact"?* We randomly select 2 tweets from 2011-01-01 to 2011-05-31 for both "@jdbiebercrews" and "#bieberfact". From the tweets shown in table 4, the tweets mentioned "@jdbiebercrews" are about to retweet from "@jdbiebercrews" and try to get Justin Bieber's private information (such as PIN number) by following each other. On the other hand, the tweets containing "#bieberfact" tend to share an anecdote about Justin Bieber. Therefore, "#bieberfact" are supposed to be more similar to "@bieber" than "@jdbiebercrews". From the results under t-test computation, similarity score of ("#bieberfact", "@bieber") is 0.30 and that of ("@jdbiebercrews", "@bieber") is only 0.21, which means that t-test computation shows a correct and accurate similarity score.

Word	Total Number of t-test Entries	Total Number of PPMI Entries
#bieberfact	210516	1502
@jdbiebercrews	210516	44
@bieber	210516	7

Table 5: Word Vector Size

However, there are defects existing in t-test: the vector after t-test computation is quite dense, Unlike PPMI, despite there is no co-occurrence between word w_1 and w_2, a t-test score can still be derived, which means the total number of vector entries under t-test is exactly the number of occurrence words and leads to enormous time and space overhead. As shown in table 5, the vector size under t-test is orders of magnitude larger than that under PPMI. In the twitter case, under t-test computation, a single word vector costs about 0.8MB (210516*4bytes) memory footprints, which is not scalable for big data analysis.

3.1.2 Quantitative Analysis

To explore the dependency of word similarity on word frequency. we randomly pick 100 word types and generate all possible word pairs. For each word pair, we use the minimum word counts of either word as frequency. The scatter figures about word frequency and similarity scores are shown in figure 1-2 (omit figure about ppmi-jaccard and t-test-jaccard).

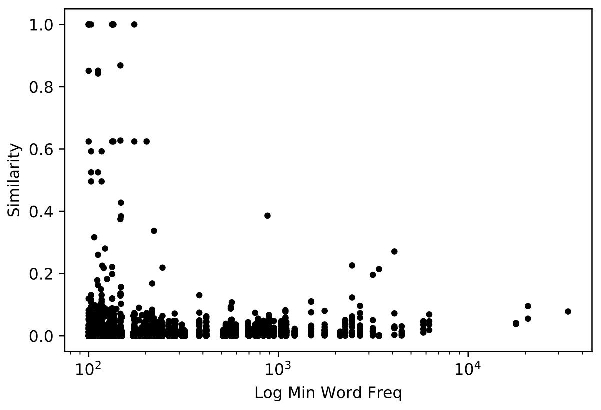

Figure 1: PPMI - cosine

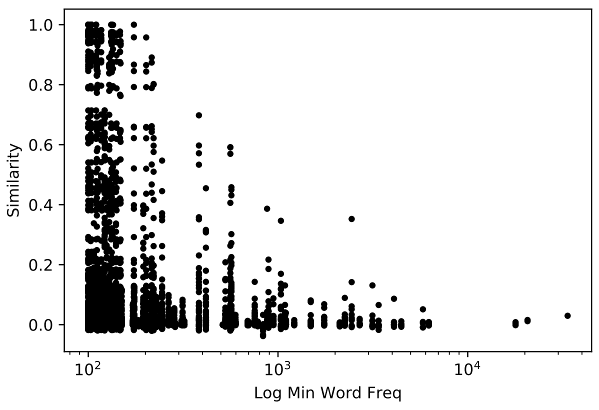

Figure 2: t-test - cosine

After data visualization, we find that the correlation is non-linear, thus Spearman Correlation[2] is used for measurement. From table 6, compared with PPMI computation, word similarity is less dependent (almost independent) on word frequency under t-test computation, which matches the analysis in section 3.1.1.

Method	Correlation
PPMI-Cosine	0.46
PPMI-Jaccard	0.49
t-test - Cosine	-0.18
t-test - Jaccard	-0.04

Table 6: Spearman Correlation Value

Method	Similarity
PPMI-Cosine	0.05
PPMI-Jaccard	0.0
t-test - Cosine	0.2
t-test - Jaccard	-2.16

Table 7: Similarity Value of love and loove

3.2 How Cosine Similarity and Jaccard rank word

Based on our result, we can see that Cosine Similarity always performs better than Jaccard method. The poor performance of Jaccard is because of the difference in word frequency in our word set. As seen from previous section, low count word vector often has many 0 entries under PMMI and negative entries under t-test. For two word vectors **X** and **Y**, the numerator of Cosine is $\sum_i (X_i * Y_i)$ while numerator of Jaccard is $\sum_i min(X_i, Y_i)$, which makes the similarity easily go to 0 under PPMI and negative under t-test for word pair with much different in frequency . In fact, looking at the word pair {love-loov} in Table 7, which has a huge gap in frequency(5178829 vs 3892). These words have high similarity value because **loov** is an alternative spelling of **love**. A quick search on Twitter gives us tweets like 'he knows that i loov him' or 'If you can make me laugh I automatically loov u < 3'. Cosine method did a good job showing these word are somewhat similar (0.05 with PPMI and 0.2 with-test). Unfortunately, Jaccard method rank this word pair as dissimilar (0.0 with PPMI and -2.16 with t-test).
We are more interested in how these methods rank words with similar frequency. We choose 3 words ['love', 'hurt', 'die'], which all have frequency > 460000. Surprisingly, Cosine method ranks {love, hurt} more similar than {die, hurt} while Jaccard suggests that {die, hurt} is more similar with both PPMI and t-test.'hurt' can be similar with 'love' because they are both used to describe feeling, but 'hurt' also similar with 'die' to show negative feeling. We found that there are 2479 co-occurrence words in our 'love' and 'hurt' word vector and 2151 words in 'die' and 'hurt' vector. More interestingly, there are 1132 words that exist in all both set and out of the top 20 co-occurrence in each set, 15 words are similar. Those words are [im, dont, time, peopl, feel, girl, life, your, friend, ill, man, hate, wanna, guy, hope]. Table 8 and 9 are the remaining 5 different words.

Word	Count
give	1157410
ur	1077218
stop	992115
ive	942366
god	932546

Table 8: love and hurt

Word	Count
love	5178829
fuck	2074423
start	1273407
shit	1885815
gonna	1177934

Table 9: die and hurt

Looking at table 8 and 9, it is unexpected to see that 'love' is the top co-occurrence with 'die' and 'hurt'. But we also notice that the frequency of words in {die, hurt } pair is much larger. Thus, we believes that the high frequency makes the total vector length much larger under Cosine method, which then ranks {die, hurt} less similar to {love, hurt}. In Jaccard case, considering PPMI method, word vector 'love' has 40250 non zero entries while word vector 'hurt' only has 4758 non zero entries and they have 2470 similar words in each vector, which means they have to fill extra 40250 +4758 -2479 = 42529 max numbers in the denominator while keeping the numerator the same. On the other hand, word vector 'die' only has 9978 non zeros entries which means we only have to fill 9978 + 4758- 2151 = 12585 max numbers in the denominator. Therefore, the word vector 'hurt' is more similar with 'die' than 'love' under Jaccard method.

4 Conclusion

About the impact of word frequency to vector computation, as discussed in section 3.1, PPMI and t-test have both advantages and disadvantages. PPMI is over sensitive to word frequency, which leads to inaccurate prediction of word similarity. On the other hand, t-test can avoid the dependency of word similarity on word frequency and return a more accurate prediction. However, it cannot be represented as a sparse vector and is not scalable for big data analysis. But PPMI can be represented in sparse vector.
After these experiment, we believe that either Cosine or Jaccard has its own strength and weakness. With unbalanced word set like ours, Cosine generally works better calculating the angle between two words while Jaccard often push the value to 0 or lower. But sometimes in situation where we have sufficient data and frequency is not as important as uniqueness, Jaccard method tends to perform well as suggested in Section 3.2.

References

[1] Michel-Marie Deza and Elena Deza. *Dictionary of distances.* Elsevier, 2006.

[2] Ann Lehman, Norm O'Rourke, Larry Hatcher, and Edward Stepanski. *JMP for basic univariate and multivariate statistics: Methods for researchers and social scientists.* SAS Institute, 2013.

[3] Tamara Polajnar and Stephen Clark. *Improving Distributional Semantic Vectors through Context Selection and Normalization.* 2013.