Claire Van Poperin
LING83600
Gorman
10/4/20

Language Technology

MP1: Word Similarity

Part 1: WordNet

def get_sense_lst()

In order to calculate the word similarity of the provided word pairs, it is necessary to first select a sense from the synset. I used WordNet. To simplify the task, I chose to use the first sense of each word. Iterating over the rows in the provided .tsv file, I used wordnet.synsets(word)[0] to access that sense which I then appended to a list (sense_lst)along with the human score of similarity. This list is returned by the function and then passed to each of the various word similarity functions. In addition, I used a counter to calculate the number of pairs found. In other words, if both words in the word pair were found (or had a sense on WordNet), the count of found pairs increased by one.

def get_path_sim()

This function use the WordNet method path_similiarity to determine the path length for the pair of word senses contained in sense_lst. WordNet is a tree-structured ontology of senses, a sort of structured thesaurus. This method scores word similarity based on the number of arcs between two senses. A word (or word sense) has a distance of one to itself and two to the nearest word (or word sense). This length is computed as 1/(distance + 1). A score closer to 1 indicates greater similarity, with 1 being the score of a word sense to itself. This means that the path_similarity method recognizes if the path it to the same sense and returns 1 rather than .5 as you might expect (1/(1+1)). The next closest score would be .33, or 1/(2+1). The plus one method is used to avoid dividing by zero.

Here are a few example results:

(Synset('tiger.n.01'), Synset('cat.n.01'), '7.35'): path sim = 0.09090909090909091
Synset('tiger.n.01'), Synset('tiger.n.01'), '10.00'): path sim = 1.0

Intuitively, I would expect a higher similarity between tiger and cat. As mentioned in the assignment, it would be preferable to return the highest similarity between all senses of two words.

After computing the individual similarities between pairs and storing these scores as a list (path_sim_lst), I then computed the Spearman Rho correlation coefficient between the path_similarity score and the provided human score. This indicates correlation in terms of monotonicity between the method generated score and the human score. In addition, I tracked the number of pairs found to calculate coverage or percent found. It has a range of -1 to 1, with a score of -1 or +1 indicating "an exact monotonic relationship."[1] The Spearman Rho score for this method and coverage are included in the results table.

def get_res_sim()

One of the main problems with the path_similarity method is that it assumes each arc to be of equal importance. In order to address this, alternative methods rely on information content. In the Resnik method, this is addressed by assigning higher probability to higher nodes or entities and lower probability to children of that entity.This is used to calculate the probability that a word is an instance of a given sense c.[2] The information content is then:

$IC(c) = -\log P(c)$

Resnik similarity uses the lowest common node (parent) between two senses to measure the amount of information in common between two words. As Juarfsky notes, "The more two words have in common, the more similar they are."[3]

As Jurafsky continues, the Resnik method can be represented as:[4]

$$sim_{resnik}(c1,c2) = -\log P(LCS(c1,c2))$$

In order to calculate the Resnik similarity for the given word pairs, I again used the first sense from WordNet. The Resnik method requires that the sense have the same part of speech. Iterating over sense_lst with an if statement (if pair[0].pos() == pair[1].pos():) allowed me to limit the method to pairs with the same part of speech. In addition, I used a tracker to calculate the coverage of this method. The Resnik method takes an IC as an argument. There are various available on WordNet. In order to ensure consistency of results for all IC based methods, I chose brown_ic. The Spearman Rho score for this method and coverage are included in the results table.

---

[1] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html
[2] http://wellformedness.com/courses/LING83600/PDFs/lecture02-wordnet.pdf, page 18.
[3] Jurafsky, Lecture 3: Word Similarity and Thesaurus Methods: https://www.youtube.com/watch?v=KHxbsDg4w6M
[4] Jurafsky, Lecture 3: Word Similarity and Thesaurus Methods: https://www.youtube.com/watch?v=KHxbsDg4w6M

Remaining methods: def get_lch_sim(), def get_jcn_sim(), def get_lin_sim(), def get_wup_sim()

The code for these four functions is essentially the same as the Resnik function with the similarity method changing. As mentioned above, I used the same IC for all these methods. These all also require matching part of speech.

Part 2-3: PPMI and Word2Vec similarity

In order to calculate the PPMI of the word pairs using ppmi.py, I first tokenized the data and then joined the tokens of each sentence using a space as required by ppmi.py. In addition, I created a two column .tsv of the word pairs provided. After running ppmi from the command line, I then used those results to calculate the Spearman Rho for the PPMI scores and the human scores. PPMI itself is a calculation that indicates if two words occur together more frequently than their joint probabilities. It is a distributional method as opposed to the thesaurus methods we saw in Part 1. In positive PPMI, the negative scores are replaced with zero.

One difficulty in calculating the PPMI stems from the fact that the results returned from ppmi are not in the same order as the original word pairs. In order to calculate Spearman Rho, the scores must be in the same order. First I read the original pairs with their scores into a list treating the word pair as a tuple and then the score: ((row[0], row[1]), row[2]). Next, I read the pairs from the PPMI results into a list as a tuple, pmi_pair = (ppmi_row[0], ppmi_row[1]). With both word pairs now in the same format, I was able to construct and if statement for matching pairs and then append the original (human) scores to a list and the PPMI scores to a list. I then passed the lists to the SciPy method for Spearman Rho.

For Word2Vec, the process was essentially the same as PPMI.

Part 4: Results

	IC	Spearman Rho	Coverage
PPMI	n/a	0.0646	0.3744

Path	n/a	0.4291	0.9901
LCH	Brown	0.4291	0.9901
JCN	Brown	0.442	0.9901
LIN	Brown	0.5161	0.9901
WUP	Brown	0.5309	0.9901
Resnik	Brown	0.5551	0.9901
Resnik	Semcor	0.5613	0.9901
Word2Vec	n/a	0.6105	0.9606

Table of Spearman Rho correlation and Coverage

Looking at the results, we can see that the range of scores is .0646 (PPMI) to .6105 (Word2Vec). The Spearman Rho correlation serves as an evaluation of the similarity method. In order to understand the results, it is necessary to understand how the metric is computed.

Looking first at path similarity methods, path similarity and LCH, we can see that the similarity score is based on the length of the path between senses:[5]

simpath(c1,c2) = 1/pathlen(c1,c2)[6]

$$\text{sim}_{lch} = -\log \frac{\text{spath}(c_1, c_2)}{2 * D}$$

LCH relies on path distance as well as the "depth of the taxonomy D." Interestingly, these two methods returned identical Spearman Rho coefficients: 0.4291. This suggests that incorporating depth in LCH did not result in increased correlation to human scores.

WUP is also based on depth of the senses as well as the depth of the LCS (lowest common subsumer). WUP yielded a higher Spearman Rho coefficient, suggesting that lcs provides relevant information. It yielded the second highest accuracy of the WordNet methods.

$$\text{sim}_{wup} = \frac{2 * \text{depth}(\text{lcs}(c_1, c_2))}{\text{depth}(c_1) + \text{depth}(c_2)}$$

[5] All equations and quotations in the following section are from unless otherwise noted: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3864022/#:~:text=Lin%20defined%20the%20similarity%20between,the%20depth%20of%20the%20concepts.

[6] Jurafsky, Lecture 3: Word Similarity and Thesaurus Methods: https://www.youtube.com/watch?v=KHxbsDg4w6M

Resnik offers a starting point for JCN and LIN. The main intuition behind Resnik's formulation is that the LCS represents the common information shared by two senses.[7] This is referred to as IC, or information content. As Budanitsky and Hirst explain, "As a consequence, the higher the position of the most specific subsumer for given two concepts in the taxonomy (i.e., the more abstract it is), the lower their similarity."[8]

$$sim_{res} = -\log P(LCS(c1, c2))$$

It is important to note that the IC chosen can influence the Spearman Rho coefficient. For example, we see a slight increase in the Spearman Rho for Resnik similarity using the Semcor IC versus the Brown IC.

One of the drawbacks of the Resnik method is "the indistinguishability, in terms of semantic distance, of any two pairs of concepts having the same most-specific subsumer."[9] At the same time, it is important to note that Resnik yielded the most accurate estimation of similarity of the WordNet methods. As a response, JCN addresses this by including the IC of both c_1 and c_2:

$$\text{sim}_{jcn} = \frac{1}{\text{IC}(c_1) + \text{IC}(c_2) - 2 * \text{IC}(\text{lcs}(c_1, c_2))}$$

Lin, on the other hand, sees similarity as a ratio of the common information of two senses to the combined information they contain.

$$\text{sim}_{\text{lin}} = \frac{2 * \text{IC}(\text{lcs}(c_1, c_2))}{\text{IC}(c_1) + \text{IC}(c_2)}$$

Interestingly, the lowest Spearman Rho, PPMI, and the highest, Word2Vec, are also distributional methods. Both methods have arguments that can be changed including window size. Looking at the provided code, the PPMI window is set to 10 as default whereas the Word2Vec window was set to 5. Perhaps this accounts for the difference in scores.

Finally, looking at coverage, all but the distributional methods require the same part of speech and so had the same coverage, 0.9901. The coverage for the distributional methods would be relative to the window size. The larger window size of PPMI (default 10) seems to have decreased the coverage (0.3744) while the smaller window size of Word2Vec (5) increased the coverage (0.9606)

[7] http://www.cs.toronto.edu/pub/gh/Budanitsky+Hirst-2004.pdf, p. 12.
[8] http://www.cs.toronto.edu/pub/gh/Budanitsky+Hirst-2004.pdf, p. 12.
[9] http://www.cs.toronto.edu/pub/gh/Budanitsky+Hirst-2004.pdf, p. 13.

Print Statements

Path Similarity:
Spearman Rho: 0.4291
Total pairs: 203
Found pairs: 201
Percent found: 0.9901

Leacock-Chodorow Similarity:
Spearman Rho: 0.4291
Total pairs: 203
Found pairs: 201
Percent found: 0.9901

Resnik Similarity:
Spearman Rho: 0.5613
Total pairs: 203
Found pairs: 201
Percent found: 0.9901

Jiang-Conrath Similarity:
Spearman Rho: 0.442
Total pairs: 203
Found pairs: 201
Percent found: 0.9901

Lin Similarity:
Spearman Rho: 0.5161
Total pairs: 203
Found pairs: 201
Percent found: 0.9901

Wu Palmer Similarity:
Spearman Rho: 0.5309
Total pairs: 203
Found pairs: 201
Percent found: 0.9901
Print Statements

PPMI Similarity:
Spearman Rho: 0.0932
Total pairs: 203
Found pairs: 80
Percent found: 0.3941

Word2Vec Similarity:
Spearman Rho: 0.5909
Total pairs: 203
Found pairs: 196
Percent found: 0.9655