Using Graph Traversal to Find Similarity Between Sentences

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Michael Anzuoni

Annandale-on-Hudson, New York
May, 2013

Abstract

Similarity is nominally the quality or condition of two things bearing likeness to one another. Semantic similarity, then, is a term used to determine how alike they are in what ideas they convey. Using semantic graphs, it is possible to create a method in which one can calculate semantic similarity algorithmically. ConceptNet is a large, human-generated graph that provides semantic and contexual information about lingual items called *concepts*. ConceptNet is a semantic graph on which the similarity between sentences is calculated. The calculation is performed by first ranking the node components, each representing a concept, to filter them down to what is essentially a sentence's semantic kernel. Then measure the semantic similarity of node pairs by calculating the length of the pathway between them as well as their proximity to other important nodes, summing up each pair's score in order to obtain a semantic rating for the sentences. Both word-to-word and and sentence-to-sentence semantic similarity measurements were accurate enough to warrant further investigation.

Contents

List of Figures

Dedication

This senior project is dedicated to the memory of Sean Munroe Leavitt.

Acknowledgments

Firstly, I would like to thank my advisor Sven Anderson. His contributions to the project were numerous and our meetings yielded a great deal of insight in how to tackle some of these thorny problems. Secondly, I would like to thank my parents, and my brother, for without them I would not be here - both in the sense that my parents union conceived me and also in the sense that they provided the support needed for a young man to pursue his dreams. I don't think I can ever repay the debt I owe them, but I will do my damnedest to try. I would like to thank Steven Spagnola for his email correspondance and willingness to divulge details of his research that led to fruitful results. Finally, I would like to thank my fellow CS student Anis Zaman Rony. Our conversations both in and out of the classroom helped to shape some of the more formless aspects of my project as well as generally serve as good points for comparison, since our projects are rather similar (ha ha).

I would also like to thank the following people who helped me in a less pragmatic sense but rather provided me with indirect inspiration as well as enduring memories that will no doubt serve me in more unfortunate times, much like a trash can fire does for a group of transients. May these names live on in search engines forever: Alex Baro, Augustus Cooper Mackenzie, Frank Mandell, Jackson Smith and/or Rory Hamovit, Tamas Panitz, Nikolas Jaeger, Joe Tisdall, Kit Schluter, Layla Wolfgang, Andrue Coombes, Daniel Schwartz, the guy I talked to in the subway about graph theory for twenty minutes, Alexander Ryan Gay, Ethan Osten, Toby Williams, Jake Coan, Grayling Bauer, Daniel Perlman, the man in McLean Hospital who said he was a millionaire and no one believed him until he bought the whole ward chinese food and then left in a limo, and Sidney Bechet.

1
Introduction

1.1 Semantic similarity

Similarity is nominally the quality or condition of two things bearing likeness to one another. In geometry, one can say that two triangles are similar if they both have two identical angles. We can say that two people look similar if they share corresponding physical features. Thus semantic similarity is a term used to denote the measurement of how aligned the meanings of sentences are with one another, that is to say how alike they are in what ideas they convey. We humans demonstrate a notion of what makes two sentences similar. It is quite easy for a human to distinguish that the sentences 'I am wearing a large hat' and 'The bear ate my sandwich' are quite dissimilar. We appear to possess linguistic faculties that guide us and we have a familiarity with the vocabularly and syntax in both sentences such that we know that a 'large hat' is not at all like a 'sandwich' or that 'wearing' is a different action than 'ate'.

However, this ability of ours is a very difficult problem for computers, which read the two sentences as strings of characters devoid of any meaning. Consequently, similarity metrics used in computation often rely on statistical analysis of the two sentences such as counting up the number of times the same word appears in both sentences. There are also string metric approachs such as the Levenshtein distance, which measures the minimum number of single-character edits need to change one word (or sentence) into another (Levenshtein, 1966). Of course, these approaches are severly limited in that they do not account at all for the actual value of the words as tokens bearing linguistic values ('tokens' being words as they are represented by computers) . For example, the words 'bank' and 'bank' look identical to a computer, even though, in terms of their semantics and context, one may be referring to a financial institution and the other to the edge of a river (e.g., 'I went to the bank to cash my check.' versus 'An alligator swam up on the bank to eat a bird.').

In finding a way to calculate the semantic similarity between sentences, we must have quantitative data to compare sentences to one another. This data could be used to comb through a document to find sentences or fragments related to a specific idea. Such semantic searching could be a significant addition to CTRL-F, form-based keyword searches, since it is possible that a document contains notions related to a specific keyword without actually including the keyword itself. Semantic similarity metrics could also be utilized to automatically generate document meta-information for use in the semantic web, allowing for more intelligent searching and organization. Computer-based algorithms to determine the similarity of sentences to one another would be widely useful in dealing with matters of natural language.

What is needed, then, is a method by which a computer can read in tokens and gain access to their semantic value. Fortunately, such representations of tokens as semantic units exist in the form of semantic graphs. Semantic graphs are databases that model the linguistic value of tokens. There are various ways to model these tokens. One approach, used by the popular database WordNet, is thoroughly hierarchical and lexical (Miller et al, 1990). In WordNet, the different senses of a word are grouped into a set of synonyms known as *synsets* and there exist separate graphs for verbs, nouns, adjectives, and adverbs.

WordNet contains different relations over each of these sets, such as a *hypernym* where a noun Y is a hypernym of the noun X if the entity X is a kind of Y (e.g., *canine* is a hypernym of *dog*.).

Lexical databases have their own limitations in terms of what they can represent. WordNet represents dogs as a type of mammal, but it does not have any information relating to the fact that dogs have teeth or that dogs have a propensity to play fetch or other contextual knowledge that our human experiences are able to provide. If we are going to analyze sentences, contextual information might augment part of speech information and thereby prove to be advantageous in our approach to finding semantic similarity.

1.2 The ConceptNet graph

Fortunately, there exists a database that provides human-generated 'common-sense' relations between tokens. ConceptNet is a directed semantic graph that provides relations between words as used in everyday speech and writing. Tokens, whether single words or multi-word phrases, are represented in ConceptNet as 'concepts', which act as nodes in the semantic graph. Connecting these nodes are 'assertions', which are edges containing semantic data linking the two nodes (Havasi and Speer, 2012).

Consider the word 'jazz'. In ConceptNet, this is modeled as the concept 'jazz', with concepts being the atomic unit with which ConceptNet represents tokens (tokens being either single words, such as 'jazz' or multi-word phrases). ConceptNet contains a lot of information about jazz, with this knowledge represented as assertions between the 'jazz' concept and other nodes. An example assertion could be modeled as *IsA*(jazz, genre of music), expressing the relation that jazz is indeed a genre of music. Other assertions could include *AtLocation*(jazz, new orleans) and *plays percussion in*(jazz, jazz drummer). The expressive power of ConceptNet comes from these assertions and the fact that they are human-generated thereby avoiding a rigid taxonomy. Rather, the flexibility allows for new

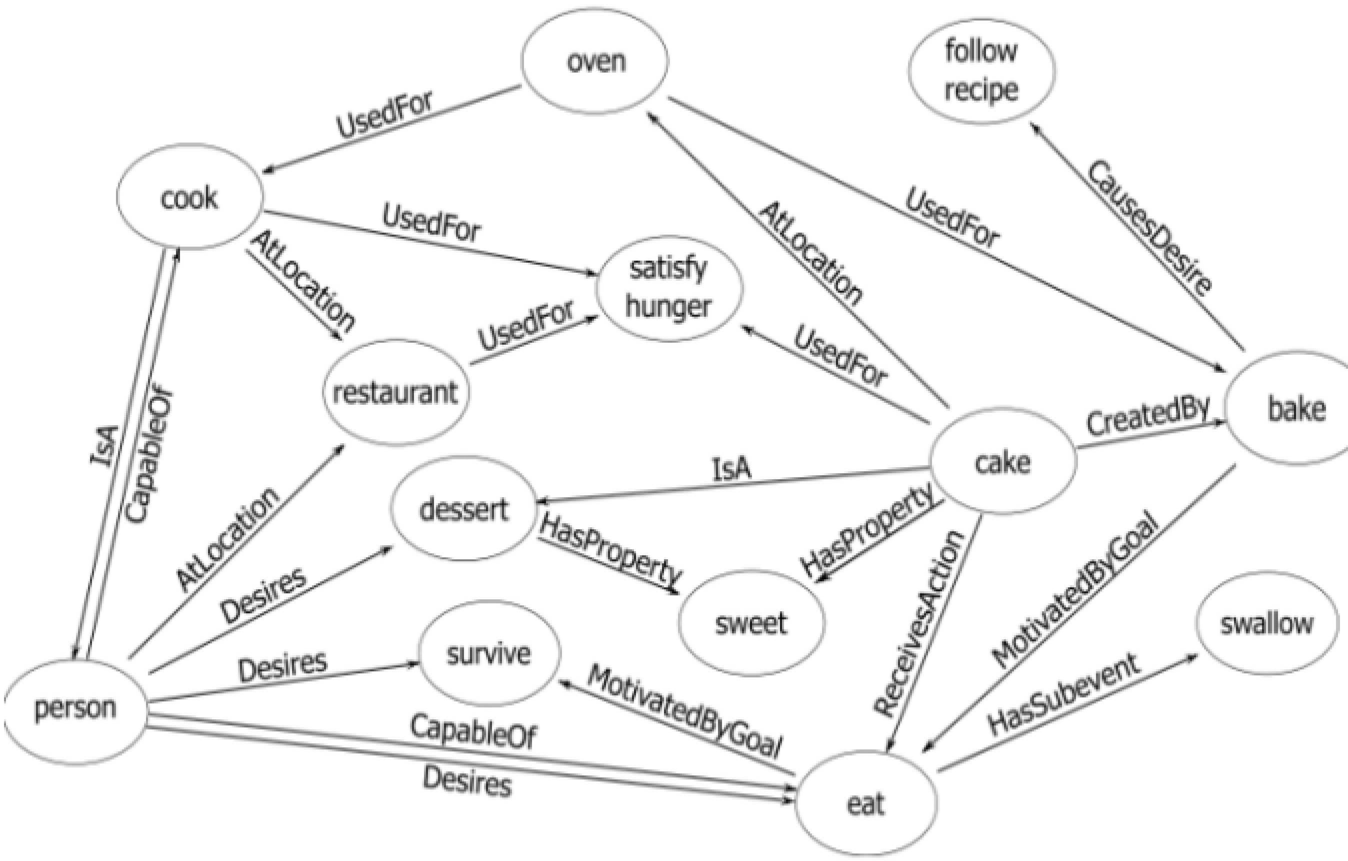
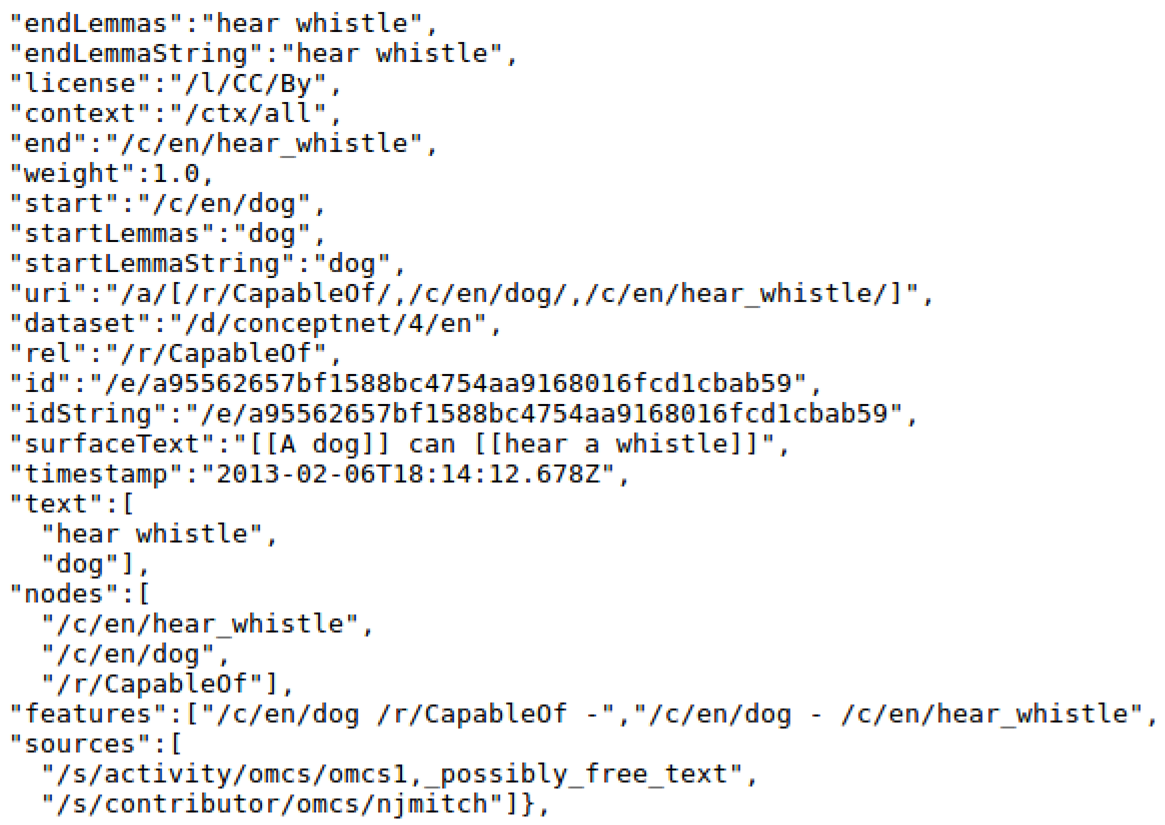


Figure 1.2.1. A graphical description of ConceptNet, courtesy of Havasi and Speer. The circles are concepts in the graph conntected by the arrows which represent the assertions between concepts.

assertions to be created, giving the semantic graph the ability to represent many relations simply not covered in lexical databases such as WordNet. Wordnet can tell you that ‘canine’ is a hypernym of ‘dog’, but ConceptNet can tell you that dogs are raced at dog tracks or that dogs enjoy playing fetch. A disadvantage in this approach is that the graph is very bushy and due to its less rigid nature is a bit more difficult to search through.

Furthermore, ConceptNet provides qualitative and quantitative information about assertions that can prove to be most useful in semantic analysis. Each edge is represented as a dictionary with fields and values (see Figure 1.2.2). These fields include the relation of



```
"endLemmas":"hear whistle",
"endLemmaString":"hear whistle",
"license":"/l/CC/By",
"context":"/ctx/all",
"end":"/c/en/hear_whistle",
"weight":1.0,
"start":"/c/en/dog",
"startLemmas":"dog",
"startLemmaString":"dog",
"uri":"/a/[/r/CapableOf/,/c/en/dog/,/c/en/hear_whistle/]",
"dataset":"/d/conceptnet/4/en",
"rel":"/r/CapableOf",
"id":"/e/a95562657bf1588bc4754aa9168016fcd1cbab59",
"idString":"/e/a95562657bf1588bc4754aa9168016fcd1cbab59",
"surfaceText":"[[A dog]] can [[hear a whistle]]",
"timestamp":"2013-02-06T18:14:12.678Z",
"text":[
  "hear whistle",
  "dog"],
"nodes":[
  "/c/en/hear_whistle",
  "/c/en/dog",
  "/r/CapableOf"],
"features":["/c/en/dog /r/CapableOf -","/c/en/dog - /c/en/hear_whistle",
"sources":[
  "/s/activity/omcs/omcs1,_possibly_free_text",
  "/s/contributor/omcs/njmitch"]},
```

Figure 1.2.2. Sample output from a ConceptNet query for 'dog'. The 'end' and 'start' fields denote the nodes which the edge (as represented by the field 'rel') connects. 'Weight', then, is the numerical score given to an edge to rank the reliability of its assertion. In this case, the assertion that a dog is capable of hearing a whistle is scored as 1.0

the assertion (IsA, HasA, and so on), the starting and ending nodes, and a weight which represents the strength of the edge (i.e., how reliable this assertion is). The weighting of the edges will prove most important in the semantic similarity calculations, but other fields will serve to help us in orienting ourselves in the massive graph that is ConceptNet. For example, the field 'rel' tells us what sort of relation the two nodes (denoted as the fields 'start' and 'end') share.

1.2.1 ConceptNet's graph structure as the basis for a measurment of semantic similarity

ConceptNet's graph structure allows for the application of commonplace graph algorithms, such as breadth-first search. From this, we can derive a means to which we can quantitatively measure semantic similar. If a concept is connected to another concept by some assertion, then those concepts are probably semantically or contextually related. This can be extrapolated to the notion that if a concept is connected to another concept by some pathway of nodes, then we can deduce there exists a relation between those concepts.

Using the pathway between two concepts, it is possible to derive a measure of the semantic similarity of the concepts. We see similar approaches used in our own natural language: a bear is similar to a dog in that they both are mammals. The words 'bear' and 'dog' are connected by means of their shared mammality. If we consider 'bear' and 'dog' as concepts in the ConceptNet graph, then they are linked by the relations *IsA*(bear, mammal) and *IsA*(mammal, dog).

Thus by searching the graph for the pathway between two concepts, we have a means by which we can measure semantic similarity. However, there are several factors that must still be considered. How can this graph search be used to find the semantic similarity between sentences and not just words? What sort of information can be derived from the ConceptNet data about the pathway between nodes? How can this information be used to determine semantic similarity? These are concerns that are addressed in the next section where we will be delving into the composition of an algorithm that measures semantic similarity.

2
Graph Traversal and Semantic Similarity

2.1 Li et al.'s sentence similarity measurements

In order to measure the accuracy of our sentence-to-sentence measurements, it is imperative to have data to compare against. Li et al 2006. provide sentence-to-sentence comparisons using the Rubenstein and Goodenough sentence pairs dataset. This dataset consists of sentence pairs that were scored by human subjects on a scale from 0.0 to 4.0, with 0.0 being not at all similar and 4.0 being exactly the same.

Li et al.'s results are useful because their approach to sentence similarity is not unlike with the one utilized here. Li and his team built a similarity engine that incorporates both lexical database information as well as corpus statistics. The lexical database (Li et al are using WordNet) provides for the semantic similarity of the words by means of mesuring the path length between them, not unlike our own approach. For the statistical approach, the Brown Corpus, a body of work consisting of 1,014,000 American english words, was used to derive an 'information content' - nominally a number representing the relative frequency of a selected word which is then used to weight the word within its sentence. At the sentence level, Li et al. create a joint word set T consisting of all the distinct words from the two sentences which is worked over combinatorically by their above described similarity engine..

Li et al's research showed strong results, with the Pearson correlation coeffecient between their algorithm scoring and the human scoring done by Rubenstein-Goodenough subjects at 0.81 (Li et al, 2006).

2.2 Creating semantic kernels to represent a sentence

With Li's approach and results in mind, we can begin constructing our graph-search algorithm. One candidate for a sentence-level semantic comparison is a combinatoric calculation in which each word from sentence S_1 is compared to each word in sentence S_2 and scored based on some metric, such as the number of edges it takes to go from one node to the other in the ConceptNet graph. Of course, this is computationally intensive and also ignores the rich contextual and semantic data the ConceptNet graph offers about concepts and their relation to one another. This approach naively treats all words as being of equal importance to the sentence's semantic value.

A second approach is to use the connectivity of a node in the semantic graph to devise a voting system in which we can extract the most important words of a sentence and just compare those. In this system, highly-connected nodes are considered as being more important to a sentence's semantic content, in effect creating a representational kernel of a sentence. Instead of comparing every word in the sentence, we can just compare these sentence kernels to one another. Suppose sentence S_1 produces two kernels, K_1 and K_2. Sentence S_2 produces one kernel, K_3. Using this approach, we can reduce the number of calculations needed to derive sentence similarity by just comparing K_1 and K_2 to K_3 under some scoring formula.

In order to generate the kernels for each respective sentence, a slight variant of the famous PageRank algorithm was employed (S. Brin and L. Page, 1998). PageRank is effectively a measure of how imporant a node is in relation to other nodes. Edges that connect to the node in question are seen as votes of confidence to the node's importance. Consequently, the more edges that connect to a node, the more influence that node has in voting up other nodes. Thus, this algorithm lends itself nicely to the problem of identifying important nodes within a sentence.

PageRank operates by iterating over the graph with this particular scoring mechanism and halts once a certain number of iterations have been run or if the difference between the last score a node has received and its current score is less than a certain amount (i.e., the algorithm has converged). A typical PageRank algorithm often performs vector calculations on a large matrix which stores the scores for each node. In this particular case, we are instead using a Python dictionary data store to keep track of each node, as seen in the code sample below. Another difference is that PageRank is usually run over the entirety of the graph. For reasons of space and efficiency, this version acts only upon a local neighborhood within the graph as defined by an edge limiter we set in the ConceptNetNode object.

```
def node_pagerank(cnet_node, damping_factor=0.85, max_iterations=100, min_delta=0.00001)
    nodes = list(cnet_node.edges)
    nodes.append(cnet_node.node_name)
    rank = 0.0
    neighbor_count = len(nodes)
    min_value = (1.0 - damping_factor) / neighbor_count
    rank_dict = dict.fromkeys(nodes, 1.0 / neighbor_count)
    for i in xrange(max_iterations):
        diff = 0
        for node in nodes:
            activated_node = ConceptNetNode(node)
            rank = min_value
            for referring_node in activated_node.edges:
                if referring_node not in rank_dict:
                    outdegree = len(activated_node.edges)
                    rank_dict[referring_node] = 1.0 / neighbor_count
                try:
                    rank += damping_factor * rank_dict[referring_node] / outdegree
                except:
                    rank = min_value
            diff += abs(rank_dict[node] - rank)
            rank_dict[node] = rank
        if diff < min_delta:
            break
    return rank_dict[cnet_node.node_name]
```

This function takes a ConceptNetNode object and returns its score as calculated by an adapted version of the PageRank algorithm. It begins by creating a list of all the node's incoming edges. It then calculates the minimum value a node can have if it has no incoming edges. Following this, a dictionary is initialized with its keys being all the nodes with edges pointing towards the input node and the value being set to a baseline amount. The *for*-loop is where the actual PageRank calculations take place. For each node, we activate it (that is, a ConceptNetNode object for it is created) and then for each referring node it has (a node that has edges pointing to it), we add it to the ranking dictionary if it is not already there and then we accumulate the score based on the PageRank scoring algorithm, which is

$$\frac{DampingFactor * NodeScore}{outdegree} \tag{2.2.1}$$

If it has no incoming edges, we just give it the minimum score. Then, once the graph has converged or run for the maximum number of set iterations, we return the node's score.

But now that we have scores for each node in the sentence (not necessarily for every word in the sentence, for there may not exist a corresponding node in the ConceptNet graph), how do we determine whether or not to add it as a kernel? The solution is to use a thresholding system that can filter out nodes based on their scores. In the case of my graph search, I find the average score of the sentence's nodes and then divide each subsequent node by this average score. If a node, when its own score is divided by the average, exceeds this threshold amount (such as 1.0 or 1.5), then it is deemed to be important and added as a kernel.

2.3 The semantic scoring of words

In order to score senence kernels, I have adapted Lagoze and Spagnola's (Lagoze and Spagnola, 2011) method for semantic scoring using a graph search. Their method utilizes a breadth-first search to find the possible pathways between one node and another. In doing so, they kept track of three important variables - $SCORES$, $EDGES$, and $TRANSITIONS$. $SCORES$ are derived from the weight given to each edge in the ConceptNet graph (Recall that the weight of an edge is an human-assigned score of how reliable the assertion is). The cube root of each of these weights along the traversal is then summed up. $EDGES$ is simply the square of the number of edges traversed going from one node to another. $TRANSITIONS$ is sum of the edge transition probabilities. Transition scores were calculated based on the overall distribution of ConceptNet edge types (IsA, HasA relations and so on). Numerically, they represent the probability of one type of edge to transition into another (The chance, for example, of an IsA edge leading to a HasA edge). These transitions are used by Lagoze and Spagnola to further contextualize edges as they move along pathways in the graph search. The intuition behind this equation is that it builds upon the inverse distance method by adding an interaction between the weights of an edge and the transition scores which help provide a context for its placement in the graph.

The resulting equation is thus

$$PathScore = \frac{SCORES * TRANSITIONS}{|EDGES|^2} \tag{2.3.1}$$

The significance of Lagoze and Spagnola's research was that the addition of the edge weights and transition scores aided the search through the graph, increasing the accuracy of the search by 21.88 percent compared to simple edge counting methods. These results provide solid evidence that ConceptNet could be utilized for semantic similarity calculations. From this, one could use Lagoze and Spagnola's graph-searching methods as the basis for a word similarity calculation, which could then extrapolated to perform full sentence-to-sentence comparisons.

2.4 Proximity scoring

Adapting Lagoze and Spagnola's technique was complicated by the lack of access to the transition probability scores used. It should also be noted that Lagoze and Spagnola were using an older version of ConceptNet that had the assertion types built-in, whereas now ConceptNet allows for new user-defined assertions. Thus, it was imperative to find a replacement metric that could be used to calculate similarity between words. In place of Lagoze and Spagnola's probability transition scores, the proximity scoring given by Majewski and Szymanski (2008) was used . Their calculation of semantic proximity relies on the notion of spreading activation through semantic memory and seeks to capture concept proximity as a function of distance between edges aided by some decaying factor.

We model their algorithm with the use of an analogy to that of a liquid spreading through a system of pipes. First, the initial node is injected with an amount of some virtual substance. A fraction of this substance remains in the node and does not go any further (this becomes the proximity score of the node). The rest of the substance is split into smaller flows proportial to the number of edges going out from the node and the process is repeated for successive nodes. If some edge connects to a node that has been visited by the substance before, we assume that the node is saturated and thus does not take in any more of the substance. The algorithm terminates when all reachable nodes under consideration have been saturated. The score is the sum of each node's proximity rating along a pathway divided by the root node's proximity score, so $A \to C$ would result in a score of 1.250.

The rationale for using this score as part of the graph search is motivated by results of Lagoze and Spagnola. Similarity measures based on edge counting and weight scoring did not perform as well as just edge-counting (Lagoze and Spagnola, 2011). Yet scores using edge counting as well as transition scores and weights fared far better. Hence there was a need for an augmentation of the edge counting, since the transition scores were unavailable. Majewski and Szymanski's scoring system is also advantangeous since it runs as a single diffusion operation. This adds little overhead to a graph traversal algorithm (in this case, a breadth-first search) which runs in $\mathrm{O}\big(f(b)\big)$, where b is the branching factor in the graph. It should be noted that since we are using PageRank to generate the semantic kernels used for our search, the branching factor is often quite high since highly-connected nodes are prioritized. As a result, much like Lagoze and Spagnola, a pathway length limit for the search was introduced in order to cut down on overhead (See Section 3.1.1).

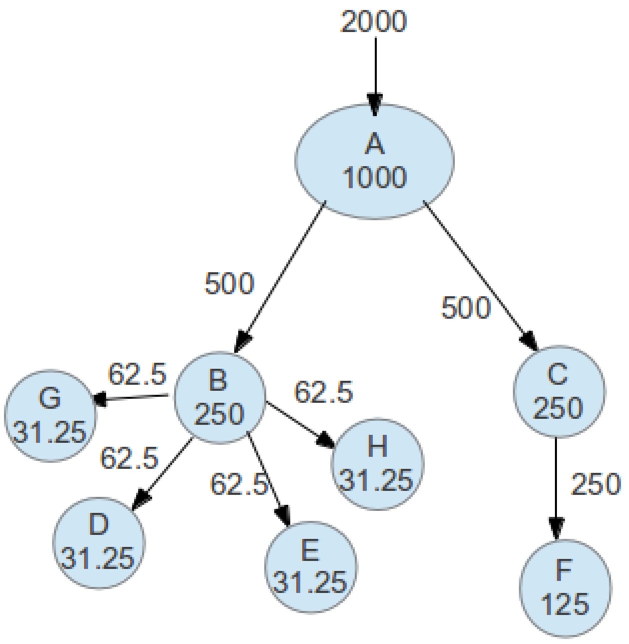


Figure 2.4.1. A visual example of Majewski and Szymanski's proximity scoring algorithm. The root node A is injected with 2000 units of a virtual substance. In this case, we set the retain amount to be 0.5, so the node has then 1000 units left to distribute amongst its two edges, the result being that the two adjacent nodes are each injected with 500 units. Since the retain amount is 0.5, these nodes' proximity scores are all 250, with 250 left over to distribute to their neighbors.

I decided to discount $EDGES$ (the squared number of edges between two nodes) with the $PROXIMITY$ score on the basis that this semantic calculation is not merely a shortest-path problem. If it were, then we would want to prioritize the number of edges. However, we are looking for a more holistic metric that can utilize the semantic information given to us by the ConceptNet graph. The $PROXIMITY$ score reduces the strength of the $EDGES$ variable in the similarity calculation by weighting nodes that are more directly connected. Consider Figure 2.3.1. In this graph, nodes F and H are both two edges away from node A. However, H is connected to B which is a highly-connected node. Consequently, its semantic proximity to A is diminished, since $A \to B \to H$ is a less direct connection than $A \to C \to F$. Because C shares only two connections, one of which is with A, this connection is prioritized and F as a result has a higher score.

Using Majewski and Szymanski's proximity measure in conjunction with Lagoze and Spagnola's search metrics yielded this equation used for word-to-word semantic scoring in my graph search:

$$Similarity = \frac{SCORES}{|EDGES|^2 - PROXIMITY} \tag{2.4.1}$$

To score sentences, we use the above described PageRank algorithm to find the semantic kernels for each sentence. Then we score each kernel-to-kernel relation (each kernel being a concept existing in ConceptNet) using equation 2.3.1, accumulating the score in some variable S. This variable is then divided by the number of kernels to account for differences in sentence length and we are left with a numeric measurement of the sentences' semantic similarity.

3 Results

This chapter covers both the tests performed on a word-to-word basis as well as the sentence-to-sentence level comparisons. It was found that the graph-search results for word-to-word and sentence-to-sentence comparisons performed reasonably well, with some issues in terms of performance overhead.

3.1 Word-to-word similarity scores

In order to test our graph search, the Rubenstein and Goodenough word pair dataset was used. This is a data set consisting of 65 word pairs that were scored based on their similarity to one another on a floating point scale from 0.0 to 4.0 (Rubenstein and Goodenough, 1965). The word pair 'gem, jewel', for example, was given the score 3.94. To score each word pair, the graph search algorithm described in Section 2.2 was employed.

It should be noted that although the graph search runs in $\mathrm{O}\big(|V| + |E|\big)$, the size of V and E is incredibly large, with there being about 12.5 million edges connecting 3.9 million vertices. Iterating over the entirety of the graph would be intractable and impractical. Thus, a pathway limit was enforced, limiting the graph search will not to go more than

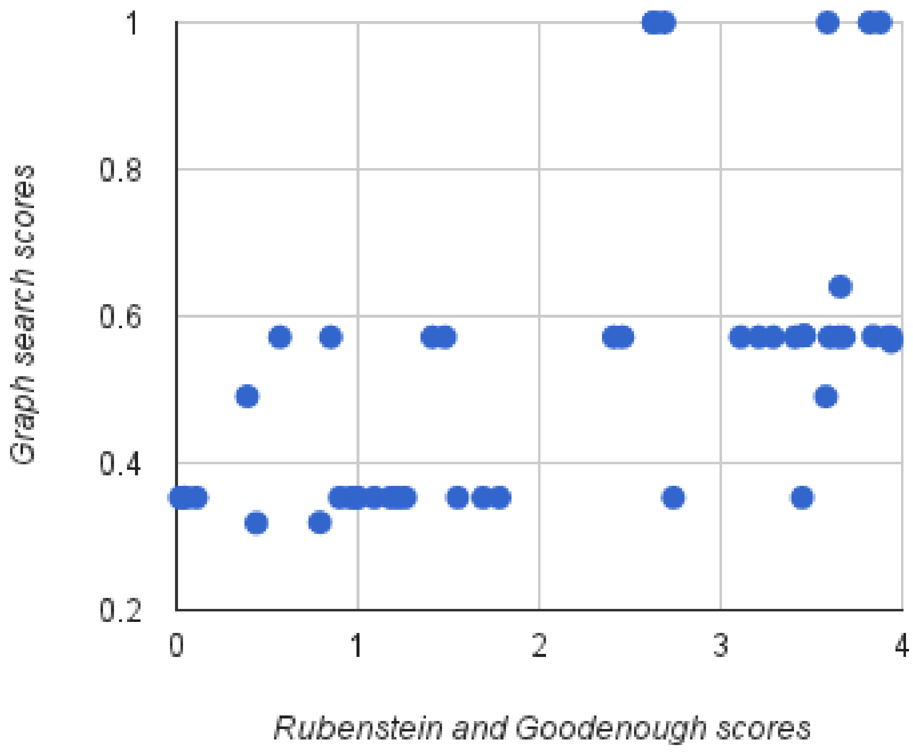


Figure 3.1.1. The Pearson's correlation coefficient when using proximity scoring is 0.57

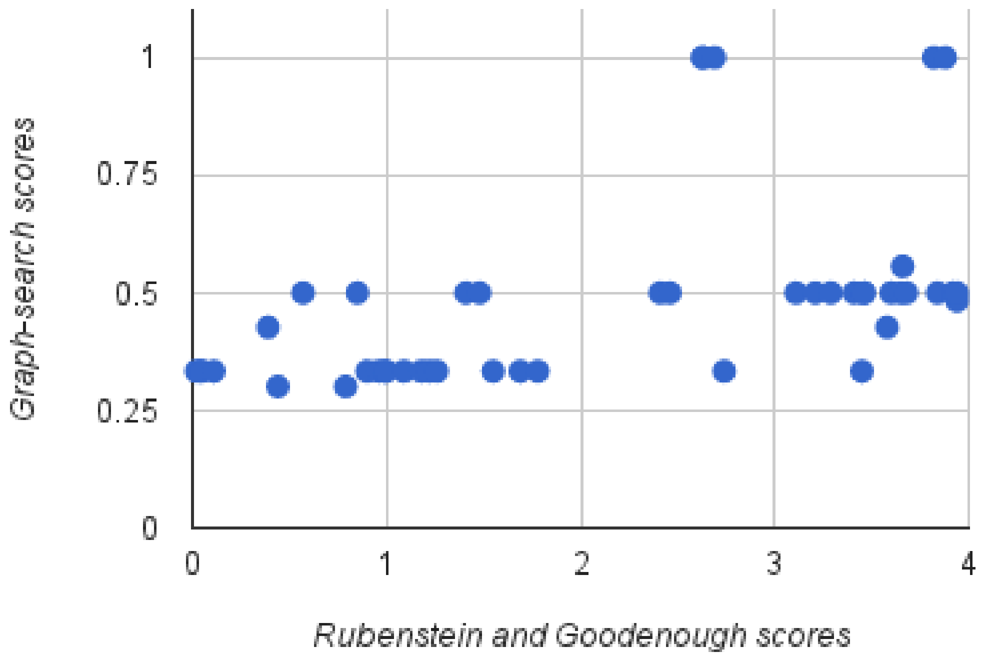


Figure 3.1.2. Without proximity scoring, the Pearson's correlation coefficient is 0.49

n'hops' from the root node. Although this might seem arbitrary, there is a rationale behind it. Namely, if a node is many hops away from the root node, then its similarity to the root node decreases. Furthermore, empirical testing of pathway limits showed that in ConceptNet 5 (the version this algorithm is using), a pathway length limit greater than 3 gave us an intractable search. If you refer back to Figure 1.2.1, you'll note the pathway $cake \rightarrow oven \rightarrow cook \rightarrow person \rightarrow restaurant$. With a pathway limit of 3, the ultimate edge of the pathway (*person AtLocation restaurant*) would be discounted in the search.

The graph-search was performed in two sets: one using the proximity scoring in the equation and one without it. The point of including both is to highlight the usefulness of including an additional weighting factor in the graph traversal scoring. Without the proximity scoring (Figure 3.1.3) we see a large amount of 'chunking' in the graph wherein a lot of the scores fall within similar tiers. The graph for scores using the proximity measure is far more varied and has a strong correlation with the Rubenstein-Goodenough dataset (Figure 3.1.2). This behavior is discussed further in section 4.1.

3.2 Sentence-to-sentence similarity scores

Rubenstein and Goodenough also provide similarity data for sentence comparisons. When applied to the test sentences provided by Rubenstein and Goodenough, we saw less of a correlation than with the word-to-word results. To save time the path length limit was initially set to 2, since sentence comparisons are more intensive than the word pair calculations. However, increasing the path length limit from 2 to 3 provided an increase in the Pearson's coefficient score, indicating that perhaps by allowing longer pathways we can arrive at more accurate scores. Of course, by increasing the path length limit, we also increase the running time of the algorithm with some sentence-to-sentence comparisons taking hours to resolve due to the sheer amount of edges and nodes available in the search.

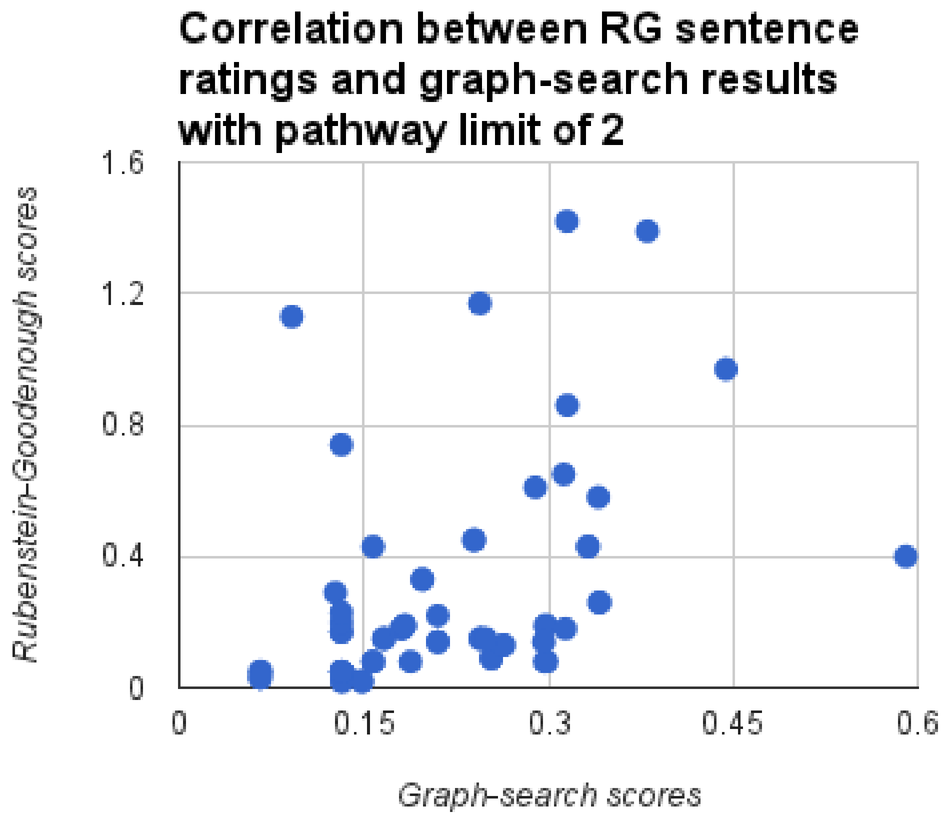


Figure 3.2.1. The Pearson's correlation coefficient was 0.406 when a pathway limit of 2 was used.

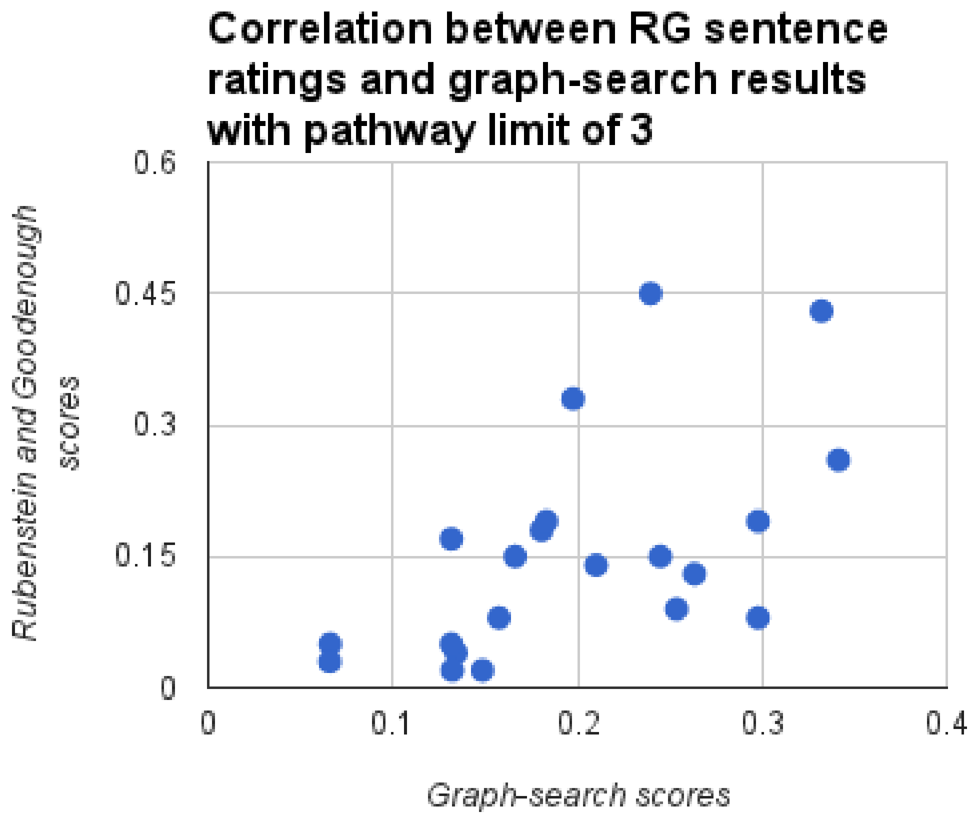


Figure 3.2.2. The Pearson's correlation coefficient was 0.582 when a pathway limit of 3 was used.

4
Discussion

4.1 Word-to-word similarity

4.1.1 The usefulness of additional weights in similarity calculations

Edges between concepts in ConceptNet already contain a 'weight' attribute which is a positive or negative floating-point number indicating the 'reliability' of the relation. In my graph search, the weights are used within the scoring formula to add contextual data in addition to the pathway counting. However, a formula that just uses the summed weights of the pathway divided by the squared length of the pathway gives less accurate results than a scoring formula that uses an additional weight. Lagoze and Spagnola used a transition probability metric in their own graph search, which resulted in a significant increase in their Pearson correlation coefficient (Lagoze and Spagnola, 2011). In my own graph traversal, a proximity score was used. The addition of the proximity scoring to the similarity equation (Eq. 2.3.1) resulted in a 6.125 % increase in accuracy, as determined by Pearson's correlation coefficient.

As such, it stands to reason that when working with semantic graphs such as ConceptNet, one may utilize a wide range of metrics to help them augment a scoring formula rather than rely on the traditional approach in which weight is divided by pathway length squared. Both Lagoze and Spagnola's approach as well as my own graph traversal outperformed said approach.

4.1.2 Stratification in word-to-word results

In Figure 3.1.3, we notice that much of the results fall within three sections. The majority of results are either in the 0.5 to 0.556 range or a solid 1.0, with a few outliers scoring 0.33. The reason behind this is because many of the weights given in ConceptNet fall within similar ranges, with 0.5 and 1.0 being incredibly common. Consequently, scores derived from weights alone tend to clump together.

Additional weighting, such as proximity scoring, has the advantage that it breaks up the data by making it more contextual and nuanced to the specific node. An edge between 'cushion' and 'pillow' may have the weight 0.5, the same as a weight between say 'automobile' and 'cushion', causing both to fall in the same range although they are markedly different (and scored as such on the Rubenstein and Goodenough dataset where I take these examples from) when using inverse pathway scoring. Proximity scoring, however, tailors the results to the context in which these concepts appear in the graph, taking into account their placement in relation to other important nodes and utilizing a scoring mechanism more sophisticated than pathway counting. The insight yielded from this measurement directly impacted the accuracy of the word-to-word similarity score in a positive manner.

4.2 Sentence-to-sentence similarity

As it stands, the results from the graph-search were quite lower than Li et al's but still rather accurate with far less stratification than seen in word-to-word comparisons. Li et al. achieved a Pearson's coefficient of 0.81, whereas the best performance of my graph-search gave us a Pearson's coefficient of 0.582. In comparison to my own word-to-word results, there was a wide variety in the results generated which matches with the variety seen in the sentences, implying that the graph-search is able differentiate between semantics with a fine amount of granularity.

Increasing the pathway length limit from 2 to 3 yielded more accurate results, as measured by the Pearson's correlation coefficient. Thus, it would seem that having a higher pathway length limit is the key to achieving more accurate results. However, increasing the path length limit resulted in an overall slowdown of score generation. Some results were not included because they had not finished calculating at the time this paper was due. Access to more powerful hardware, such as a server cluster, would no doubt expedite the score generation making a higher pathway limit a more attractive option since its appears to produce increased accuracy.

4.3 Considerations and Conclusions

Overall, in using my graph-search, word-to-word and sentence-to-sentence semantic similarity measurements were quite accurate. Although my sentence-to-sentence measurement did not outperform Li et al's current it boasted a respectable amount of accuracy and did so in nearly linear time.

In analyzing these results, it is imperative to also examine the data tested. There does not exist a large corpus of similarity measurement data from humans that we can use to compare and as such must rely on datasets such as Rubenstein and Goodenough. These datasets are, of course, limited. 65 sentences is simply not that much. What would be needed for further investigation into semantic similarity is the generation of more comprehensive datasets by human subjects, including human input regarding the accuracy of the graph-search results.

Another factor to be considered is the overhead of graph-searching. Although breadth-first search runs in $\mathrm{O}\big(|V|+|E|\big)$, it is confounded by the sheer size of the branching factor inherent to the structure of ConceptNet, since a large amount of nodes have more than ten edges. Searches can take up to several hours and beyond. Therefore, it is my hope to refine the approach of the graph search in particular by looking at more sophisticated graph-search algorithms, such as iterative deepening or bi-directional searches.

Semantic similarity is still very much an open problem in computer science. As computers amass more and more textual data, such as a news / journal articles or blog posts, it is imperative to create algorithms that can help computers sift through this information in an intelligent and accurate manner. By building this graph-search algorithm, I hope to have contributed some useful information to the field of natural language processing.

Bibliography

[1] Vladimir Levenshtein, *Binary codes capable of correcting deletions, insertions, and reversals*, Soviet Physics Doklady (1966), 707–710.

[2] G. A. Miller and R. Beckwith and C. D. Fellbaum and D. Gross and K. Miller., *WordNet: An online lexical database*, International Journal of Lexicography (1990), 235–244.

[3] S. Brin and L. Page, *The anatomy of a large-scale hypertextual Web search engine*, Computer Networks and ISDN Systems (1998), 107-117.

[4] Carl Lagoze and Steve Spagnola, *Edge dependent pathway scoring for calculating semantic similarity in ConceptNet* (2011), 1–5.

[5] Pawel Majewski and Julian Szymanski, *Text Categorization with Semantic Commonsense Knowledge: First Results*, Neural Information Processing (2008), 1–10.

[6] Zuhair A. Bandar and Keeley Crockett and Yuhua Li and David McLean and James D. O'Shea, *Sentence Similarity Based on Semantic Nets and Corpus Statistics*, IEEE Transactions on Knowledge and Data Engineering **18** (2006), 1138–1150.

[7] Herbert Rubenstein and John B. Goodenough, *Contextual correlates of synonymy*, Communications of the ACM (1965), 627-633.

[8] Catherine Havasi and Robert Speer, *Representing General Relational Knowledge in ConceptNet 5*, Conference on Language Resources and Evaluation (2012), 3679–3686.

[9] Hugo Liu and Push Singh, *Commonsense Reasoning in and over Natural Languge* (2004), 1–14.