Creating a Similarity Graph from WordNet

Lubomir Stanchev
Indiana University – Purdue University Fort Wayne
Fort Wayne, Indiana, USA
stanchev@gmail.com

ABSTRACT

The paper addresses the problem of modeling the relationship between the words in the English language using a similarity graph. The mathematical model stores data about the strength of the relationship between words expressed as a decimal number. Both structured data from WordNet, such as that the word "canine" is a hypernym (i.e., kind of) of the word "dog", and textual descriptions, such as that the definition of the word "dog" is: "a member of the genus Canis that has been domesticated by man since prehistoric times", are used in creating the graph. The quality of the graph data is validated by comparing the similarity of pairs of words using our software that uses the graph with results of studies that are performed with human subjects. To the best of our knowledge, our software produces better correlation with the results of both the Miller and Charles study and the WordSimilarity-353 study than any other published research.

Categories and Subject Descriptors

I.2.7 [**Computing Methodologies**]: Artificial Intelligence—*Natural Language Processing*

General Terms

Algorithms

Keywords

similarity graph, similarity distance, WordNet

1. INTRODUCTION

The main goal of the paper is to describe how to create a similarity graph that can be used to calculate the degree of semantic similarity between the words of the English language. For example, the graph can be used to tell us that the similarity between the words "dog" and "cat" is around 0.13 because the two words are somewhat similar. In the same way, the graph can be used to tell us that the similarity between the words "dog" and "phone" is around 0.02 because there is little correlation between the two words. The graph can also be used to compute the strength of the asymmetric relationship between words. For example, the graph can tell us that someone who is interested in documents about the word "bird" is also interested in documents about the word "animal" with relatively high probability. However, just because someone is interested in documents about the word "animal" does not give us confidence that they are also interested in documents about the word "bird" because most animals are not birds.

If we type "automobile" in our favorite Internet search engine, for example Google or Bing, then all top results will contain the word "automobile". Most search engines will not return web pages that contain the word "car" but do not contain the word "automobile" as one of the top results. The reason is that most Internet search engines rely on keyword matching to compute the query result and do not posses the knowledge that the words "automobile" and "car" are semantically similar and the degree of this semantic similarity. The similarity graph captures this semantic similarity. For example, the graph can be used to find words that are semantically similarly to the word "automobile" and rewrite the query using these words. In this way, the similarity graph will allow us to not only perform semantic search (i.e., search based on the meaning of the words), but it will also help us rank the result. For example, results that contain the word "car" should appear before results that contain the word "auto" if, according to the graph, the word "car" is semantically closer to the word "automobile" than the word "auto". Another interesting software application is using the similarity graph to partition a set of documents based on the meaning of the words in them. The similarity graph can be used to measure the semantic similarity between any pair of documents. Then a clustering algorithm, such as K-Means clustering ([17]), can be applied. The similarity graph can also be used as part of a query-answering system, such as the IBM Watson Computer that competed on the Jeopardy game show and the Siri system for the iPhone. For example, suppose that the word "automobile" is part of the user query. Then the similarity graph can be used to rewrite the query using similar words, for example the word "car". Such a rewrite can help the system find more information that is related to the user query.

The problem of evaluating the strength of the semantic relationship between words is intrinsically hard because computers are not as proficient as humans in understanding

WIMS '14, June 2-4, 2014 Thessaloniki, Greece
Copyright 2014 ACM 978-1-4503-2538-7/14/06$15.00
http://dx.doi.org/10.1145/2611040.2611055.

natural language text. However, natural language descriptions can provide important evidence about the similarity between words. For example, the definition of the word "cat" in WordNet is: "feline mammal usually having thick soft fur and no ability to roar". This definition can be used as evidence about the strength of the semantic relationship between the words "cat" and "feline". Although significant effort has been put in automated natural language processing (e.g., [6, 7, 18]), current approaches fall short of understanding the precise meaning of human text. In fact, the question of whether computers will ever become as fluent as humans in understanding natural language text is an open problem. In this paper, unlike most natural language processing applications, we do not parse text and breakdown sentences into the primitive elements of the language (e.g., nouns, verbs, etc.). Instead, we only examine the words in the text and the order in which they appear.

Current approaches that extract information about word similarity from freely accessible sources focus on the structured information. In particular, most papers that deal with WordNet (e.g., [15, 35]) adapt the approach taken in [25] that semantic similarity can be measured solely based on the inheritance (a.k.a. kind-of) links and possibly data about the specificity of the words (i.e., their information content – see [24, 16, 11]). More recent papers, such as [36], explore additional relationship between words, such as the holonym (a.k.a. part-of) relationship. Although these approaches work well in practice and produce similarity data that closely correlates to data from human studies, such as [19], we show that there is room for improvement. In particular, unstructured information, such as the definition of a word or an example use of a word, is not considered. For example, the WordNet definition of one of the senses of "New York" is that it is a city that is located on the Hudson river. This close relationship between "New York" and "Hudson" is not considered by the algorithms of the papers that are cited in this paragraph because these algorithms do not process textual information.

In this paper, we propose a novel mechanism for measuring the semantic similarity between words based on WordNet. We show how information from WordNet can be used to create a *similarity graph*, where the algorithm can be easily extended to include other sources (e.g., Wikipedia). The graph is created using probability theory and corresponds to a simplified version of a Bayesian network ([23]). The weight of an edge represent the probability that someone is interested in the content of the destination node given that they are interested in the content of the source node. Note that the weight function is asymmetric. We experimentally validate the quality of our algorithm on two independent benchmarks: Miller and Charles ([19]) and WordSimilarity-353 ([5]). Our approach outperforms existing algorithms that we are familiar with on both benchmarks because we process more information as input, including natural language descriptions, and we are able to apply this information to build a better model of the semantic relationships between words.

In what follows, in Section 2 we review related research. The major contributions of the paper are the introduction of the similarity graph, see Section 3, and the introduction of two novel algorithms for measuring the semantic similarity between words, which are presented in Section 4. Section 5 describes how the similarity graph can be used to measure the semantic similarity between a pair of documents. Section 6 shows how our system compares with existing systems that measure word similarity, while concluding remarks and areas for future research are outlined in Section 7.

2. RELATED RESEARCH

A preliminary version of this paper was published in a workshop ([30]). Since then, all the algorithms have been significantly revised. As a result, the experimental results show significant improvement. The correlation with the results of human subjects in both the Miller and Charles study and the WordSimilarity-353 study is now higher. For example, the algorithms from [30] produces worst quality data than the results that are published in [2], while the revised algorithms that are presented here produce correlation that is higher than the results from [2]. In addition, this paper includes a previously unpublished algorithm for comparing the semantic similarity between a pair of text documents.

Existing research that applies Bayesian networks to represent knowledge deals with the uncertain or probabilistic information in the knowledgebase (e.g., [26, 22]). In this paper, we will take a different approach and we will not use Bayesian networks to model uncertain information. In contrast, we will create a probabilistic graph that stores information about the similarity of different words. Unlike Bayesian networks, we store only the probability that a word is relevant given that an adjacent (in the graph) word is also relevant (e.g., unlike Bayesian networks, we do not store the probability that a word is unrelated given that an adjacent in the graph word is unrelated).

The idea of creating a graph that stores the degree of semantic similarity between words is not new. For example, [13, 27] show how to create a graph that only represents inheritance of words, while [10] approximates the similarity of words based on information about the structure of the graph in which they appear. These papers, however, differ from our approach because we suggest representing available evidence from all type of sources, including natural language descriptions. Our approach is also different from the use of a semantic network ([32]) because the latter does not consider the strength of the relationship between the nodes in the graph.

There are alternative methods to measure the semantic similarity between words. The most notable approach is the Google approach ([4]) in which the similarity between two words is measured as a function of the number of Google results that are returned by each word individually and the two words combined. Other approaches that rely on data from the Internet include [2] and [14]. Although these approaches produce good measurement of semantic similarity, they have their limitations. First, they do not make use of structured information, such as the hyponym relationship in WordNet. Second, they do not provide evidence about how the two words that are compared are related. In contrast, our approach can show the paths in the similarity graph between the two words, which serves as evidence that supports the similarity score.

Research from information retrieval is also relevant to creating and using the similarity graph. For example, if the word "ice" appears multiple times in the definition of the word "hockey", then this provides evidence about the relationship between the two words. Our approach will use a model that is similar to TF-IDF (stands for term frequency, inverse document frequency – see [12]) to compute the strength of the relationship. In the TF-IDF model, if the word "ice" appears two times in the definition of the word "hockey", then the term frequency can be computed as 2. This number is multiplied by a number that is inversely proportional to how often the word "ice" appears in the definition of other words. For example, if most words contain the word "ice" as part of their definition, then the fact that the word "hockey" contains this word is not consequential. Conversely, if the word "ice" appears only in the definition of few words, then the fact that the word "hockey" contains the word "ice" in its definition is statically meaningful.

Note that a lot of research effort has recently focused on using a description language, such as OWL (stands for Web Ontology Language – [1]), to describe document resources. A semantic query language, such as SPARQL (a recursive acronym that stands for SPARQL Protocol and RDF Query Language – [28]), can be used to search for relevant documents. This approach differs from our approach because it does not provide ranking of the query result. At the same time, a SPARQL query returns exactly the resources that fulfil the query description. Alternatively, our system can return resources that are related to the input query in ranked order. Using a similarity graph has some added advantages: there is no need to describe the resources using a mathematical language, there is no need to phrase the query using a mathematical language, and the system is much more scalable (OWL knowledgebases are usually applied only to a limited knowledge domain because query answering over them is intrinsically computationally expensive.)

3. MODELING WORDNET AS A SIMILARITY GRAPH

WordNet ([20]) gives us information about the words in the English language. In our study, we use WordNet 3.0, which contains approximately 150,000 different words. WordNet also contains phrases, such as "sports utility vehicle". WordNet uses the term *word form* to refer to both the words and the phrases in the corpus. Note that the meaning of a word form is not precise. For example, the word "spring" can mean the season after winter, a metal elastic device, or natural flow of ground water, among others. This is the reason why WordNet uses the concept of a *sense*. For example, earlier in this paragraph we cited three different senses of the word "spring". Every word form has one or more senses and every sense is represented by one or more word forms. A human can usually determine which of the many senses a word form represents by the context in which the word form is used.

WordNet contains a plethora of information about word forms and senses. For example, it contains the definition and example use of each sense. It also contains information about the relationship between senses. The senses in WordNet are divided into four categories: nouns, verbs, adjectives, and adverbs. For example, WordNet stores information about the *hyponym* and *meronym* relationship for nouns. The *hyponym* relationship corresponds to the "kind-of" relationship (for example, "dog" is a hyponym of "canine"). The *meronym* relationship corresponds to the "part-of" relationship (for example, "window" is a meronym of "building"). Similar relationships are also defined for verbs, adjectives, and adverbs. Our software system also uses information from the University of Oxford's British National Corpus ([3]). The corpus contains information about the frequency of use of a little over 200,000 of the most common words in the English language. In addition to that, our computer system uses a list of about 100 noise words in the English language.

The output of our computer system is a *similarity graph* that stores the relationship between the word forms and senses in WordNet and the strength of each relationship expressed as a decimal number. A node in the graph is created for every word form and every sense. While the label of a word form node is the word form, the label of a sense node is the definition of the sense. In order to calculate the strength of the semantic relationship between two word forms, the system may go through several word form nodes and several sense nodes. In general, our similarity algorithms traverses the paths in the graph between two word form nodes. Note that the graph is directed and there can be edges with different weights in each direction between two nodes. The weight of an edge is the approximation of the probability that a user is interested in the concept that is described by the destination node of the edge given that they are interested in the concept that is described by the source node. We calculate this probability based on the data from WordNet, the University of Oxford's British National Corpus, and our list of noise words.

We next present the algorithm for creating the similarity graph in great details. Each subsection describes a method in our system. Note that there maybe multiple evidence about the strength of the relationship between two nodes. Instead of drawing multiple edges between the two nodes, we draw a single edge that has weight that is equal to the sum of the weights of these edges.

3.1 Processing the Word Forms

WordNet contains files that contain all the nouns, verbs, adverbs, and adjectives in the corpus. Our program scans through these files and creates a node for every word form. We extract every word form from the file and store it as a node in the graph. The label of a node is the word form it originated from in all lowercase letters. This helps us to avoid storing the same word in the graph multiple times, but with different capitalization of letters. We use Berkeley DB ([21]), which is a popular key-value store that is supported by Oracle Corporation, to store the graph. By testing different systems, we have reached the conclusion that this is a more efficient approach for our needs than using a full-fledged database management system or a graph database, such as Neo4J ([34]).

3.2 Processing the Senses

We next examine all the word form nodes in the graph (i.e., all the nodes that we have so far) and use JAWS ([29]), the WordNet Java API, to find the senses for each word form. Then we create a node for every sense. The label of a sense node is the definition of the sense in all lowercase letters. We also add edges between the word form nodes and the sense nodes. For example, we will create edges between the node for the word "chair" and the nodes for the three different senses of the word – see Figure 1. WordNet gives us the information about the frequency of use of each sense. The frequency of the first use is 35, the frequency of the second use is 2, and the frequency of the third use is 1. We will therefore create the outgoing edges from the node "chair" that are shown in Figure 1. The reasons is that, based on the available information, the probability that a user that requests information about the word "chair" is interested in the first sense of the word is equal to $35/(35+2+1) = 0.92$. We assume that the information in WordNet tells us that 92% of the time when someone refers to a chair, they have in mind the first meaning. The backward edges to the node "chair" represent the knowledge that all three senses represent the same word "chair". The weight is 1 because if someone is thinking about one of the three senses in the figure, then they must be thinking about the word "chair".

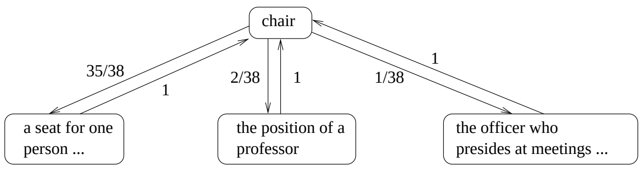

chair
35/38
1
2/38
1
1/38
1
a seat for one person ...
the position of a professor
the officer who presides at meetings ...

Figure 1: Example edges between a word form and its senses.

3.3 Adding Definition Edges

Next, let us consider the second sense of the word "chair": "the position of a professor". The noise words: "the", "of", and "a" will be ignored. We will therefore create edges between the node for the sense and the words "position" and "professor" (see Figure 2). The graph represents the connection between a sense and the non-noise words in its definition. Empirical studies have shown that the first words in the definition of a sense are far more important than the later words. We will therefore multiple the weight of the edge between the sense and the first word by *coef* $= 1.0$ and keep decreasing this coefficient by 0.2 for each sequential word until the value of the coefficient reaches 0.2. We will compute the weight of each edge as *coef* $*$ *computeMinMax*$(0, 0.6, ratio)$, where the variable *ratio* is calculated as the number of times the word appears in the definition of the sense divided by the total number of non-noise words. The variable denotes the importance of the word in the definition of the sense. For example, if there are only two words in the definition of the sense, then they are both very important. However, if there are twenty words in the definition of the sense, then each individual word is less important.

The *computeMinMax* function returns a number that is in most cases between the first two arguments, where the magnitude of the number is determined by the third argument. Since the appearance of a word in the definition of a sense is not a reliable source of evidence about the relationship between the two, the value of the second argument is set to 0.6 for definition edges. The constant 0.6 is related to the probability that someone who is interested in a sense will be also interested in one of the words in the definition of the sense. The *computeMinMax* function smooths the value of the *ratio* parameter. For example, a word that appears as one of the twenty non-noise words in the definition of a sense is not ten times less important than a word that appears as one of the two non-noise word in the definition of a sense. The function makes the difference between the two cases less extreme. Using this function, the weight of the edge in the second case will be only roughly four times smaller than the weight of the edge in the first case. This is a common approach when processing text. The importance of a word in a text decreases as the size of the text increases, but the importance of the word decreases at a slower rate than the rate of the growth of the text. Formally, the function *computeMinMax* is defined as follows.

$$computeMinMax(minValue, maxValue, ratio) = \\ minValue + (maxValue - minValue) * \frac{-1}{log_2(ratio)}$$

Note that when *raio* $= 0.5$, then the function returns *maxValue*. An unusual case is when the value of the variable *ratio* is bigger than 0.5. For example, if *ratio* $= 1$, then we have division by zero and the value for the function is undefined. We handle this case separately and assign value to the function equal to $1.2 * maxValue$. This is an extraordinary case when there is a single non-noise word in the text description and we need to assign higher weight to the similarity edge.

Figure 2 shows the portion of the graph that we described. For *ratio* of $\frac{1}{2}$, $\frac{-1}{log_2(ratio)}$ will be equal to 1. As the ratio decreases, so will the similarity score. We have used the logarithmic function in order to smoothen the decrease of the similarity score as the value of the *ratio* decreases. To summarize, we assume that the probability that a user is interested in a word will be higher if : (1) the word appears multiple times in the definition of the sense, (2) the word is one of only few words in the definition of the sense, and (3) the word is one of the first words of the definition of the sense. For now, ignore the backward edge between the word "position" and the sense.

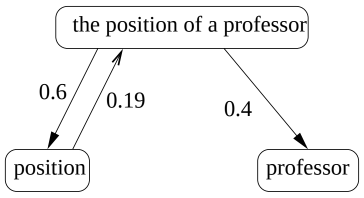


Figure 2: Example edges between a sense and the word forms in its definition.

3.4 Adding Example Use Edges

WordNet also includes example use for each sense. For example, it contains the sentence "he put his coat over the back of the chair and sat down" as an example use of the first sense of word "chair". Since an example use does not have as strong a correlation as the definition of a sense, we will calculate the weight of an edge as *computeMinMax*$(0, 0.2, ratio)$. Here the variable *ratio* is the number of times the word appears in the example use divided by the total number of non-noise words in the example use. The constant 0.2 is related to the probability that someone who is interested in a sense will be also interested in one of the words in the example use of the sense. Figure 3 shows the graph that is created for the example use of the first sense of the word "chair". Note that the noise words have been omitted. The similarity is the same for all edges because all words appear once in the example use. Unlike the case with the definition of a sense, there is not empirical evidence that the first words of the example use are more important. We will explain the backward edge from the word "coat" to the sense in the next subsection.

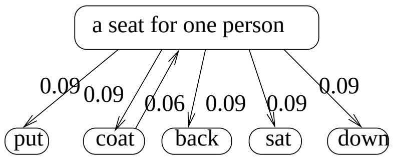


Figure 3: Example edges between a word sense and the words in its example use.

3.5 Adding Backward Edges

We also draw an edge between a word form node and a sense node for every edge between a sense node and a word form that appears in its definition. The weight of an edge will be computed as $computeMinMax(0, 0.3, ratio)$ for reverse definition edges. The variable *ratio* is the number of the times the word form appears in the definition of the sense divided by the total number of occurrences of the word form in the label of a sense. The constant 0.3 relates to the probability that someone who is interested in a word will also be interested in one of the senses that have the word in their definition. For example, if the word "position" occurred as part of the definition of only three senses and exactly once in each definition, then there will be an edge between the nodes "position" and "the position of a professor" in Figure 2 with weight that is equal to $computeMinMax(0, 0.3, 1/3) = 0.19$ (see Figure 2).

Similarly, we will draw an edge between a word form and a sense node for every edge between a sense node and word form that appears in its example use. The weight of an edge in this case will be equal to $computeMinMax(0, 0.1, ratio)$, where *ratio* is the number of the times the word form appears in the example use of the sense divided by the total number of occurrences of the word form in the example use of senses. The constant 0.1 relates to the probability that someone who is interested in a word will also be interested in one of the senses that have the word in their example use. For example, if the word "coat" occurred as part of the example use of only three senses and exactly once in each sense, then there will be an edge between the nodes "coat" and "a seat for one person" in Figure 3 with weight that is equal to $computeMinMax(0, 0.1, 1/3) = 0.06$ (see Figure 3).

3.6 Populating the Frequency of the Senses

So far, we have shown how to extract information from textual sources, such as the text for the definition and example use of a word sense. We will next show how structured knowledge, such as the hyponym (a.k.a. kind-of) relationship between senses, can be represented in the similarity graph. Most existing approaches (e.g., [24]) explore these relationships by evaluating the *information content* of different word forms. Here, we adjust this approach and focus on the frequency of use of each word in the English language as described in the University of Oxford's British National Corpus ([3]).

DEFINITION 1 (SIZE OF A SENSE). *Consider the sense m. Let $\{w_i\}_{i=1}^n$ be the word forms for that sense. We will use $BNC(w)$ to denote the frequency of the word form w in the British National Corpus. Let $p_m(w)$ be the frequency of use the sense m of the word form w, as specified in WordNet, divided by the sum of the frequencies of use of all senses of w (also as defined in WordNet). Then we define the* size *of m to be equal to $\sum_{i=1}^{n}(BNC(w_i) * p_m(w_i))$.*

The size of a sense approximates its popularity. For example, according to WordNet the word "president" has six different senses with frequencies: 14, 5, 5, 3, 3, and 1. Let us refer to the fourth sense: "The officer who presides at the meetings ..." as m. According to Definition 1, $p_m(president) = 3/31 = 0.096$ because the frequency of m is 3 and the sum of all the frequencies is 31. Since the British National Corpus gives the word "president" a frequency of 9781, the contribution of the word "president" to the size of the sense m will be equal to $BNC(president) * p_m(president) = 9781 * 0.096 = 938.976$. Other word forms that represent the sense m will also contribute to the size of the sense.

3.7 Processing Structured Knowledge About Nouns

WordNet defines the *hyponym* (a.k.a. kind-of) relationship between senses that represent nouns. For example, the most popular sense of the word "dog" is a hyponym of the most popular sense of the word "canine". Consider the first sense of the word "chair": "a seat for one person ...". WordNet defines fifteen hyponyms for this sense, including senses for the words "armchair", "wheelchair", and so on. In the similarity graph, we will draw an edge between this first sense of the word "chair" and each of the hyponyms. Let the probability that someone that is interested in a sense is also interested in one of the sub-senses be equal to 0.9. In order to determine the weight of the edges, we need to compute the size of each sense. In the British National Corpus, the frequency of "armchair" is 657 and the frequency of "wheelchair" is 551. Since both senses are associated with a single word form, we do not need to consider the frequency of use of each sense. If "armchair" and "wheelchair" were the only hyponyms of the sense "a seat for one person ...", then the corresponding part of the similarity graph will be constructed as shown in Figure 4. The weight of each edge is equal to 0.9 multiplied by the size of the sense divided by the sum of the sizes of all the hyponym senses of the initial sense. The idea is that the weights of the edges to "bigger" senses will be bigger because it is more likely that the user is thinking about one of these senses.

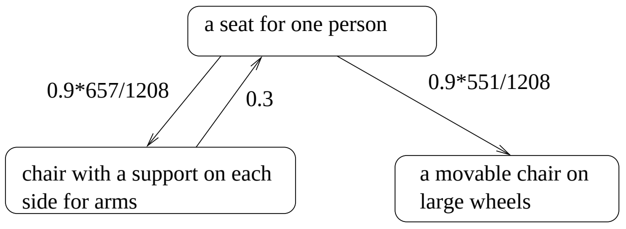

a seat for one person
0.9*657/1208
0.3
0.9*551/1208
chair with a support on each side for arms
a movable chair on large wheels

Figure 4: Example edges between a word sense and its hyponyms.

We will also draw edges for the *hypernym* relationship (the inverse of the hyponym relationship). For example, the first sense of the word "canine" is a hypernym of the first sense of the word "dog". The weight of each edge will be the same and equal to the value 0.3. This represents the probability that someone who is interested in a sense will be also interested in the hypernym of this sense. For example, if a user is interested in the sense "wheelchair", then they may be also interested in the the first sense of the word chair. However, this probability is not a function of the different hypernyms of the sense. Figure 4 shows an example of how the edge weights are computed.

We next consider the *meronym* (a.k.a. part-of) relationship between nouns. For example, WordNet contains the information that the sense of the word "back": "a support that you can lean against ..." and the sense of the word "leg": "one of the supports for a piece of furniture" are both meronyms of the first sense of the word "chair". In other words, back and legs are building parts of a chair. This information can be represented in a similarity graph, as show in Figure 5. In general, the weight of a forward edge is set to $0.6/n$, where n is the number of meronyms of the sense. The constant 0.6 represents the probability that a user that is interested in a sense of a word form is also interested in one of its meronyms. In our system, this coefficient is set to 0.6 because the meronym relationship is not as strong as the hyponym relationship. The idea of the formula is that the the more meronyms a sense has, the less likely is that we are interested in one of the meronyms.

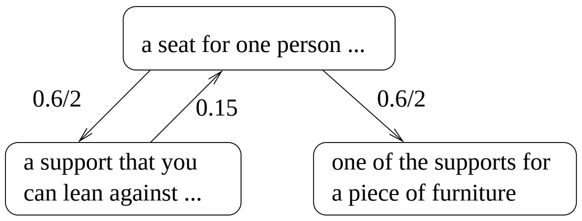

a seat for one person ...
0.6/2
0.15
0.6/2
a support that you can lean against ...
one of the supports for a piece of furniture

Figure 5: Representing meronyms and holonyms.

We also represent the *holonym* relationship in the similarity graph. For example, the main sense of the word "building" is a holonym of the main sense of the word "window". In general, X is a holonym of of Y exactly when Y is a meronym of X. Similar to hypernyms, we set the weights of edges for the holonym relationship to a constant. The constant is 0.15 because the holonym relationship is not as strong as the hypernym relation. It provides information that the an object is a building part of a different object, which does not translate in strong relevance about the relationship between the two objects. In our example, we draw an edge between the sense for the word "back" and the sense for the word "chair" that is equal to 0.15 (see Figure 5).

3.8 Processing Structured Knowledge About Verbs

We will first represent the *troponym* relationship for verbs. The verb Y is a troponym of the verb X if the activity Y is doing X in some manner. For example, to lisp is a troponym of to talk. Suppose that the verb "talk" has only three troponyms: "lisp", "orate", and "converse". If the sizes of the main senses of the three verbs is 18, 1, and 95, respectively, then we will create the edges that are shown in Figure 6. Note that the forward edges are multiplied by the constant 0.9. This represents that there is a 90% chance that if someone is interested in a verb, then they are also interested in one of its troponyms. We will add a reverse edge with constant weight of 0.3. This means that if someone is interested in one of the troponyms, then there is 30% chance that they are also interested in the original verb – see Figure 6.

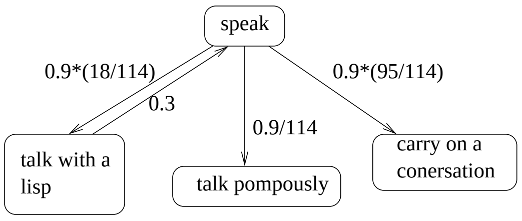


Figure 6: Representing troponyms.

The hypernym relationship is defined not only for nouns, but also for verbs. For example, the verb Y is a hypernym of the verb X if the activity X ia a kind of Y. For example, the verb "perceive" is the hypernym of the verb "listen". We will handle the hypernym relationship for verbs the same way that we handled it for nouns.

3.9 Processing Structured Knowledge About Adjectives

WordNet defines two relationships for adjectives: *related to* and *similar to*. For example, the first sense of the adjective "slow" has definition: "not moving quickly...", while the first sense of the adjective "fast" has the definition: "acting or moving or capable of acting or moving quickly". WordNet specifies that the two senses are *related to* each other. We will draw an edge between the two senses with weight 0.6 – see Figure 7. This represents that there is a 60% probability that someone who is interested in an adjective is also interested in a *related to* adjective.

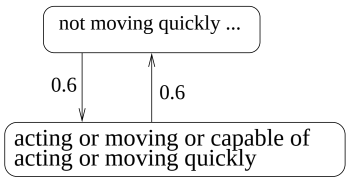


Figure 7: Representing the *related to* relationship between adjectives.

WordNet also defines the *similar to* relationship between adjectives. We draw edges with weight 0.8 between similar senses because the similar to relationship is stronger than the related to relationship. In other words, we believe that there is 80% probability that someone who is interested in an adjectives is also interested in a *similar to* adjective. For example, WordNet contains the information that the sense for the word "frequent": "coming at short intervals or habitually" and the sense for the word "prevailing": "most frequent or common" are similar to each other. We will therefore draw edges with weight of 0.8 between the two senses – see Figure 8.

4. MEASURING SEMANTIC SIMILARITY BETWEEN WORD FORMS

The similarity graph is used to represent the conditional probability that a user is interested in a word form given that they are interested in an adjacent word form in the graph. We compute the directional similarity between two

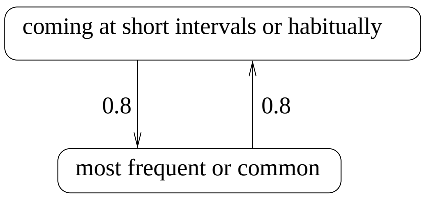


Figure 8: Representing the *similar to* relationship between adjectives.

nodes using the following formula.

$$A \rightarrow_s C = \sum_{Pt \text{ is a cycleless path from node A to node C}} P_{Pt}(C|A) \quad (1)$$

$$P_{Pt}(C|A) = \prod_{(n_1,n_2) \text{ is an edge in the path } Pt} P(n_2|n_1) \quad (2)$$

Informally, we compute the directional similarity between two nodes in the graph as the sum of all the paths between the two nodes, where we eliminate cycles from the paths. Each path provides evidence about the similarity between the word forms that are represented by the two nodes. We compute the similarity between two nodes along a path as the product of the weights of the edges along the path, which follows the Markov chain model. Since the weight of an edge along the path is almost always smaller than one (i.e., equal to one only in rear circumstances), the value of the conditional probability will decrease as the length of the path increases. This is a desirable behavior because a longer path provides less evidence about the similarity of the two end nodes.

Next, we present two functions for measuring similarity. The linear function for computing the similarity between two word forms is shown in Equation 3.

$$|w_1, w_2|_{lin} = min(\alpha, \frac{w_1 \rightarrow_s w_2 + w_2 \rightarrow_s w_1}{2}) * \frac{1}{\alpha} \quad (3)$$

The minimum function was used in order to to cap the value of the similarity function at 1. α is a coefficient that amplifies the available evidence. The experimental section of the paper shows how the value of α affects the correlation between the results of the system and that of human judgement.

The second similarity function is inverse logarithmic, that is, it amplifies the smaller values. It is shown in Equation 4. The *norm* function simply multiplies the result by a constant (i.e., $-log_2(\alpha)$) in order to move the result value in the range [0,1]. Note that the *norm* function does not affect the correlation results.

$$|w_1, w_2|_{log} = norm(\frac{-1}{log_2(min(\alpha, \frac{w_1 \rightarrow_s w_2 + w_2 \rightarrow_s w_1}{2}))}) \quad (4)$$

Given two nodes, the similarity between them is computed by performing a breadth-first traversal of the graph from each node in parallel. Common nodes between the two traversals identify paths between the two nodes. When the weight of a path becomes under 0.001, we prune out the path. We do this in order to make the algorithm more efficient. Paths with weight under 0.001 will have little effect on the semantic similarity between two word forms. In our experimental results, we only consider path of lengths 100 edges or less. A path with length more than 100 edges will provide little evidence about the relationship between two word forms.

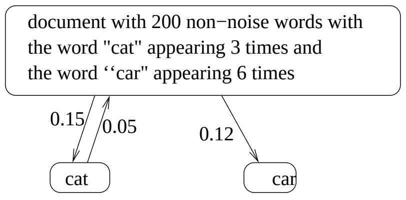
Note that we take the average of the two directed similarity distances to determine the similarity score. Empirical observations have shown that multiplying the two numbers is a inferior approach because often one of the two numbers is very small. For example, consider trying to compute the similarity distance between the words "ostrich" and "animal". One should hope this score to be high because the two words are clearly related. However, the directional similarity between the words "animal" and "ostrich" is low because there is very little evidence that someone who is interested in an animal is interested exactly in an ostrich.

5. MEASURING SEMANTIC SIMILARITY BETWEEN DOCUMENTS

In the previous section, we described how to measure the semantic similarity between two word forms that appear in WordNet. If one of the inputs is not a word form from WordNet, then the algorithm returns 0 as the semantic similarity. In this section we describe how to measure the semantic similarity between any two text documents. The idea is to create a node for each document and then connect the nodes to the existing graph. The semantic similarity between the two documents will then be measured by computing the semantic distance between the two nodes using one of the algorithms from the previous section.

In order to demonstrate our approach, consider a fictitious document that contains a total of one hundred non-noise words. Among these non-noise words, suppose that the word "cat" appears three times and the word "car" appears six times. We will represent this information by drawing the graph that is shown in Figure 9. The weight of the edge between the document and the word "car" is equal to $computeMinMax(0, 0.6, 6/100) = 0.15$. Similarly, the weight of the edge between the document and the word "cat" is equal to $computeMinMax(0, 0.6, 3/100) = 0.12$. This is the same formula that we used to compute the weight of an edge between a sense and the words in its definition.

Next, consider the backward edge between the word "cat" and the document. Suppose that the word appears a total of 200 times in all documents. Then the weight of the edge between the word "cat" and the document will be equal to $computeMinMax(0, 0.3, 3/200) = 0.05$. This is the same formula that we used for computing the weights of the backward edges between a word form and the sense definitions in which it appears.



document with 200 non−noise words with the word "cat" appearing 3 times and the word "car" appearing 6 times
0.15
0.05
0.12
cat
car

Figure 9: Example of edges between a document node and word form nodes.

First, note that the noun "cat" has eight different senses.

Our algorithm does not try to identify which of these senses the document refers to. For example, it may be possible that different occurrences of the word in the document refer to different senses. Instead, our algorithm identifies the edges to the word forms. The strength of the relationship to particular senses will be computed based on additional evidence. For example, if the document also contains the word "feline", then there will be stronger connection between the document and the main sense of the word "cat".

Second, note that the distance between two documents is not calculated in isolation. In particular, the other documents in the corpus are also taken into account when calculating the backward edges. In other words, we calculate how similar two documents are relative to the other documents in the corpus. If we want to cluster the documents, then we can perform K-Means clustering after we calculate the distances between all pairs of documents.

6. EXPERIMENTAL RESULTS

The system consists of the two programs: one that creates the similarity graph and one that queries the similarity graph. We used the Java API for WordNet Searching (JAWS) to connect to WordNet. The interface was developed by Brett Spell ([29]). All experiments were performed on a Silicon Graphics UV10 Linux machine. The Web interface of the system was created using JavaServer Pages (JSP)([31]). It takes about ten minutes to build the similarity graph and save it to the hard disk and about one minute to load it from the hard disk. The average time for computing the similarity distance between two words is thirty seconds.

We used the system to compute the similarity of 28 pairs of words from the Miller and Charles study ([19]). The study presented the words to humans and computed the mean score of the human ranking. The results are shown in Table 1. The table shows the result for α equal to 0.1, 0.2, and 0.3 for the linear and logarithmic similarity distance. As Table 2 suggests, the correlation drops as the value of α diverges from these values.

Table 2 shows the result of the correlation with different values for α. Table 3 show how our results compare with other proposals for extracting semantic similarity between word forms from WordNet. The results are for $\alpha = 0.1$. As the table suggests, both our algorithms produce better results (i.e., closer correlation with the results from the human judgement experiment in [19]) than existing algorithms.

α	$\lvert\cdot\rvert_{lin}$	$\lvert\cdot\rvert_{log}$
0.1	0.92	0.88
0.2	0.88	0.89
0.3	0.84	0.85
0.4	0.82	0.82
0.5	0.79	0.77
0.6	0.76	0.71
0.7	0.70	0.64
0.8	0.72	0.54
0.9	0.69	0.43
1.0	0.67	0.32

Table 2: Correlation results for different values of α on the Millers and Charles benchmark.

algorithm	*correlation*
Hirst and St-Onge ([8])	0.74
Leacock and Chodorow ([15])	0.82
Resnik ([24])	0.77
Jiang and Conrath ([11])	0.85
Lin ([16])	0.83
$\lvert\cdot\rvert_{lin}$	0.92
$\lvert\cdot\rvert_{log}$	0.88

Table 3: Correlation results with the Millers and Charles benchmark.

We explore how the coefficient α affects the quality of the result. We get the highest correlation with the results from the Miller and Charles study ([19]) when α is equal to 0.1 and when we use the linear similarity metric (0.92 correlation score). A correlation score of 0.92 shows very close correlation between the results produced by our system and the human judgement from the Miller and Charles study. To the best of our knowledge, this is the highest correlation with the study ever achieved in published research.

In order to avoid overfitting, we decided to check if similar results hold for a different benchmark. In particular, we used the WordSimilarity-353 dataset ([5]). It contains 353 word pairs. Thirteen humans were used to rate the similarity between each pair of words and give a score between 1 and 10 (10 meaning that the words have the same meaning and 1 meaning that the words are unrelated). The average similarity rating for each word pair was recorded. Table 4 shows the correlation of our linear and logarithmic algorithms with different values of α with the results from the WordSimilarity-353 benchmark.

α	$\lvert\cdot\rvert_{lin}$	$\lvert\cdot\rvert_{log}$
0.1	0.51	0.49
0.2	0.53	0.53
0.3	0.53	0.53
0.4	0.52	0.52
0.5	0.50	0.49
0.6	0.49	0.45
0.7	0.47	0.40
0.8	0.46	0.35
0.9	0.45	0.29
1.0	0.45	0.17

Table 4: Correlation results for different values of α on the WordSimilarity-353 benchmark.

Table 5 shows how our system compares with eight existing systems that have documented their performance on the WordSimilarity-353 benchmark. The results of our system are for $\alpha = 0.2$. As the table shows, our system produces better results then all other systems. Note that some algorithm (e.g., [2]) use additional information from the web, while our algorithm only uses information from WordNet. As we extend our system to use information from Wikipedia, we hope to further improve the quality of the results of our system.

7. CONCLUSION AND FUTURE RESEARCH

We presented an algorithm for building a similarity graph from WordNet. We verified the data quality of the algo-

algorithm	*correlation*
Jarmasz ([9])	0.27
Hirst and St-Onge ([8])	0.34
Jiang and Conrath ([11])	0.34
Strube and Ponzetto ([33])	0.19-0.48
Leacock and Chodrow ([15])	0.36
Lin ([16])	0.36
Resnik ([24]	0.37
Bollegala et al. ([2])	0.50
$\vert \cdot \vert_{lin}$	0.53
$\vert \cdot \vert_{log}$	0.53

Table 5: Correlation Results with [5]

rithm by showing that it can be used to compute the semantic similarity between word forms and we experimentally verified that the algorithm produces better quality results than existing algorithms on the Charles and Miller and WordSimilarity-353 word pairs benchmarks. We believe that we outperform existing algorithms because our algorithm processes not only structured data, but also natural language. The next major topic for future research is to extend the similarity graph and incorporate data from Wikipedia.

8. REFERENCES

[1] OWL Web Ontology Language Guide. *http://www.w3.org/TR/owl-guide/.*

[2] D. Bollegala, Y. Matsuo, and M. Ishizuka. A Relational Model of Semantic Similarity Between Words Using Automatically Extracted Lexical Pattern Clusters from Web. *Conference on Empirical Methods in Natural Language Processing*, 2009.

[3] L. Burnard. Reference Guide for the British National Corpus (XML Edition). *http://www.natcorp.ox.ac.uk*, 2007.

[4] R. L. Cilibrasi and P. M. Vitanyi. The Google Similarity Distance. *IEEE ITSOC Inforamtion Theory Workshop*, 2005.

[5] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing Search in Context: The Concept Revisited. *ACM Transactions on Information Systems*, 20(1):116–131, January 2002.

[6] C. Fox. Lexical Analysis and Stoplists. *Information Retrieval: Data Structures and Algorithms*, pages 102–130, 1992.

[7] W. Frakes. Stemming Algorithms. *Information Retrieval: Data Structures and Algorithms*, pages 131–160, 1992.

[8] G. Hirst and D. St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. *Fellbaum*, pages 305–332, 1998.

[9] M. Jarmasz. Roget's Thesaurus as a Lexical Resource for Natural Language Processing. *Master's thesis, University of Ottawa*, 1993.

[10] G. Jeh and J. Widom. SimRank: A Measure of Structural-context Similarity. *Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 538–543, 2002.

[11] J. Jiang and D. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. *Proceedings on International Conference on Research in Computational Linguistics*, pages 19–33, 1997.

[12] K. Jones. "a statistical interpretation of term specificity and its application in retrieval". *Journal of Documentation*, 28(1):11–21, 1972.

[13] R. Knappe, H. Bulskov, and T. Andreasen. Similarity Graphs. *Fourteenth International Symposium on Foundations of Intelligent Systems*, 2003.

[14] S. Kulkami and D. Caragea. Computation of the Semantic Relatedness Between Words Using Concept Clouds. *International Conference of Knowledge Discovery and Information Retrieval*, 2009.

[15] C. Leacock and M. Chodorow. Combining Local Context and WordNet Similarity for Word Sense Identification. *WordNet: An electronic lexical database*, pages 265–283, 1998.

[16] D. Lin. An Information-theoretic Definition of Similarity. *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 296–304, 1998.

[17] J. B. MacQueen. Some Methods for classification and Analysis of Multivariate Observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, page 281Ű297, 1967.

[18] M.F.Porter. An Algorithm for Suffix Stripping. *Readings in Information Retrieval*, pages 313–316, 1997.

[19] G. Miller and W. Charles. Contextual Correlates of Semantic Similarity. *Language and Congnitive Processing*, 6(1):1–28, 1991.

[20] G. A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[21] Oracle. Berkeley DB. *http://www.oracle.com.*

[22] R. Pan, Z. Ding, Y. Yu, and Y. Peng. A Bayesian Network Approach to Ontology Mapping. *Proceedings of the Fourth International Semantic Web Conference*, 2005.

[23] J. Pearl. Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning. *Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine, CA.*, page 329Ű334, 1985.

[24] P.Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. *International Joint Conference on Artificial Intelligence*, pages 448–453, 1995.

[25] R. Rada, H. Mili, E. Bickness, and M. Blettner. Development and Application of a Metric on Semantic Nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30, 1989.

[26] Q. Rajput and S. Haider. Use of Bayesian Networks in Information Extraction from Unstructured Data Sources. *Proceedings of International Conference on Ontological and Semantic Engineering*, pages 325–331, 2009.

[27] Simone Paolo Ponzetto and Michael Strube. Deriving a Large Scale Taxonomy from Wikipedia. *22nd International conference on Artificial intelligence*,

2007.

[28] E. Sirin and B. Parsia. SPARQL-DL: SPARQL Query for OWL-DL. *3rd OWL: Experiences and Directions Workshop (OWLED)*, 2007.

[29] B. Spell. Java API for WordNet Searching (JAWS). *http://lyle.smu.edu/ tspell/jaws/index.html*, 2009.

[30] L. Stanchev. Building Semantic Corpus from WordNet. *The First International Workshop on the role of Semantic Web in Literature-Based Discovery*, 2012.

[31] L. Stanchev. Similarity Software. *http://softbase.ipfw.edu:8080/Similarity*, 2012.

[32] M. Steyvers and J. Tenenbaum. The Large-Scale Structure of Semantic Networks: Statistical Analyses and a Model of Semantic Growth. *Cognitive Science*, 29(1):41–78, 2005.

[33] M. Strube and S.P.Ponzetto. Wikirelate! Computing Semantic Relatedness using Wikipedia. *Association for the Advancement of Artificial Intelligence Conference*, 2006.

[34] J. Webber and I. Robinson. *Graph Databases*. O'Reilly, 2013.

[35] Z. Wu and M. Palmer. Verb semantics and lexcial selection. *Annual Meeting of the Association for Computational Linguistics*, pages 133–138, 1994.

[36] D. Yang and D. M. Powers. Measureing Semantic Similarity in the Taxonomy of WordNet. *Australian Computer Science Conference*, pages 315–322, 2005.

			(*linear*)			(*logarithmic*)		
word 1	*word 2*	*M&C*	0.10	0.20	0.30	0.1	0.2	0.3
car	automobile	3.92	1.00	1.00	1.00	1.00	1.00	1.00
gem	jewel	3.84	1.00	1.00	1.00	1.00	1.00	1.00
journey	voyage	3.84	1.00	1.00	0.83	1.00	1.00	0.87
boy	lad	3.76	1.00	1.00	1.00	1.00	1.00	1.00
coast	shore	3.7	1.00	1.00	1.00	1.00	1.00	1.00
asylum	madhouse	3.61	1.00	1.00	1.00	1.00	1.00	1.00
magician	wizard	3.5	1.00	1.00	0.93	1.00	1.00	0.94
midday	noon	3.42	1.00	1.00	1.00	1.00	1.00	1.00
furnace	stove	3.11	0.71	0.36	0.24	0.87	0.61	0.46
food	fruit	3.08	0.62	0.31	0.21	0.83	0.58	0.43
bird	cock	3.05	1.00	0.50	0.33	1.00	0.70	0.52
bird	crane	2.97	0.44	0.22	0.15	0.74	0.52	0.39
tool	implement	2.95	1.00	1.00	1.00	1.00	1.00	1.00
brother	monk	2.82	0.80	0.40	0.27	0.91	0.64	0.48
crane	implement	1.68	0.06	0.03	0.02	0.44	0.31	0.23
lad	brother	1.66	0.59	0.30	0.20	0.81	0.57	0.43
journey	car	1.16	0.54	0.27	0.18	0.79	0.55	0.41
monk	oracle	1.1	0.01	0.00	0.00	0.31	0.22	0.16
food	rooster	0.89	0.42	0.21	0.14	0.72	0.51	0.38
coast	hill	0.87	0.19	0.10	0.06	0.58	0.41	0.30
forest	graveyard	0.84	0.23	0.11	0.08	0.61	0.42	0.32
monk	slave	0.55	0.04	0.02	0.01	0.41	0.29	0.22
coast	forest	0.42	0.10	0.05	0.03	0.50	0.35	0.26
lad	wizard	0.42	0.05	0.02	0.02	0.43	0.30	0.23
chord	smile	0.13	0.25	0.13	0.08	0.62	0.44	0.33
glass	magician	0.11	0.05	0.03	0.02	0.44	0.31	0.23
noon	string	0.08	0.16	0.08	0.05	0.56	0.39	0.29
rooster	voyage	0.08	0.06	0.03	0.02	0.46	0.32	0.24
correlation		1.0	0.92	0.88	0.84	0.88	0.89	0.85

Table 1: Similarity results for different values of α on the Milers and Charles benchmark.