Stanchev. Published in the International Journal on Artificial Intelligence Tools, 24(2). April, 2015. 1-30.

Measuring the Strength of the Semantic Relationship Between Words

Lubomir Stanchev

Computer Science Department, California Polytechnic State University
San Luis Obispo, CA, USA
stanchev@gmail.com

We propose a novel way for extracting the strength of the semantic relationship between words from semi-structured sources, such as WordNet. Unlike existing approaches that only explore the structured information (e.g., the hypernym relationship in WordNet), we present a framework that allows us to utilize all available information, including natural text descriptions. Our approach constructs a *similarity graph* that stores the strength of the semantic relationship between words. Specifically, an edge between two words describes the probability that someone who is interested in resources about the first word will be also interested in resources about the second word. Note that the graph is asymmetric because the probability that someone is interested in the second word given that they are interested int the first word is not the same as the probability that they are interested in the first word given that they are interested in the second word. The similarity between any two words in the graph can be computed as a function of the directed paths between the two nodes in the graph that represent the words.

We evaluate the quality of the data in the similarity graph 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. We also present an extended evaluation section that describes how the different heuristics that we use affect the correlation score.

Keywords: Similarity graph; similarity distance; WordNet.

1. Introduction

A plethora of research effort has focused on representing knowledge using an RDF[27] (stands for *resource description framework*) graph. Such a graph can be used to represent the relationship between concepts. For example, inside an RDF graph we can store that "For Whom the Bell Tolls" is written by "Ernest Hemingway". This information can help us in query-answering systems.[16] For example, if someone asks "Who wrote For Whom the Bell Tolls?", then we can use the graph to answer "Earnest Hemingway". However, an RDF graph alone is less useful for finding

resources and ranking them based on an input query. For example, consider someone driving and searching for a "Mexican Restaurant". If there are no Mexican restaurants near by, then the system will not return any results. However, a better option will be to consider all restaurants that are close by and return them ranked based on the semantic similarity to the phrase "Mexican Restaurant". For example, the system may contain the knowledge that "Puerto Rican Restaurant" is semantically closer to "Mexican Restaurant" than "Greek Restaurant" and therefore return Puerto Rican restaurants before Greek restaurants. Our ultimate goal is to build a similarity graph that stores the degree of semantic similarity between words and concepts. In this article, we show how to create a graph that contains about 150 000 English words that are taken from WordNet. Reference 33 shows how the algorithm can be extended to include titles of Wikipedia articles.

The goal of the similarity graph is not to replace existing RDF or OWL[37] (OWL stands for *web ontology language*) knowledgebases or text search engines. Instead, the graph is aimed to work in concert with existing technology. Consider for example a user that searches a set of documents for the word "car". Traditional search engines may apply the TF-IDF[11] (stands for term frequency, inverse document frequency) technique to rank the query result and every single document in the result will contain the word "car". However, one may argue that a document that contains the words "automobile", "Ford" and "Chrysler" multiple times and that does not contain the word "car" is more relevant to the user query than a document that contains the word "car" once. Having the knowledge that the word "car" is semantically similar to the words "automobile", "Ford", and "Chrysler" and the degree of this semantic similarity can transform a traditional keywords-based search engine into a semantic search engine that considers the meaning of the words in both the input query and the document corpus.

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 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 forward in automated natural language processing,[5,6,18] 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 article, 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[14,39] adapt the approach from Ref. 25 that semantic similarity can be measured solely based on the inheritance (a.k.a. kind-of) relationships and possibly data about the specificity of the words (i.e., their information content[24,15,10]). More recent papers[40] explore additional relationship between senses of 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,[19] we show that there is room for improvement. In particular, unstructured information, such as the definition of a sense of a word or an example use of a sense 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 river" 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 article, we propose a novel mechanism for measuring the semantic similarity between words based on WordNet.[20] WordNet contains information about 150 000 of the most common words in the English language. It contains the senses of each word, the definition and example use of the senses of each word, the relationship between senses, and so on. We show how all this information can be used to create a *similarity graph*, where the algorithm can be easily extended to include other sources (e.g., Wikipedia[33]). The graph is created using probability theory and corresponds to a simplified version of a Bayesian network.[23] The weight of an edge represents 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. For example, there is a high probability that some one who is interested in a blue jay will be also interested in birds because the blue jay is a bird. However, the probability that someone who is interested in birds will also be interested in the blue jay is relatively low because most birds are not blue jays. We use all available evidence from WordNet to create the similarity graph, including natural text descriptions and structured information. Every piece of evidence about the similarity between two words is modeled as a probability value and the available evidence is aggregated. As expected, modeling this probability is not exact science and we experimentally evaluate how different approaches affect the quality of the data.

In order to compute the similarity between two words, we consider all directed acyclic paths between the two nodes that they represent. Each path provides evidence about the relationship between the two words. The paths in one direction give us the probability that someone who is interested in the first word is also interested in the second. The paths in the other direction give us the probability that someone who is interested in the second word is also interested in the first word. We explore several options for aggregating the probabilities from the paths in each direction in order to compute the two numbers. We take the average of the two numbers in order to compute the similarity between the two words. We also explore how other functions, such as multiplying the two numbers, will affect the quality of the data. We

evaluate our algorithm under two independent benchmarks: Miller and Charles[19] and WordSimilarity-353.[4] The first benchmark contains 28 pairs of words and their similarity as determined by human subjects. We examine how the values for the different parameters of our algorithm affect the correlation with the data from the first benchmark. We pick the optimal parameters and then apply the algorithm on the second benchmark that contains 353 pairs of words. The algorithm that is optimized for the first benchmark produces higher correlation than all existing research on both benchmarks. We believe the reason is that 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 article are the introduction of the similarity graph, see Section 3, and the introduction of several parameterized 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. In Section 6, we compare the performance of our algorithm with that of existing systems on two different benchmarks[19,4] and we also examine how the different heuristics that are used in the algorithm affect the quality of the data. Concluding remarks and areas for future research are outlined in Section 7.

2. Related Research

A preliminary version of this article was published in the conference proceedings of the Fourth International Conference on Web Intelligence, Mining and Semantics.[34] Here, the paper is significantly revised. Corrections are made and more detailed explanations are provided in every section. However, the major contribution of this article is extending the experimental section and showing how different parameters and heuristics that are used in the algorithm affect the quality of the data.

Existing research that applies Bayesian networks to represent knowledge deals with the uncertain or probabilistic information in the knowledgebase.[26,22] In this article, we will take a different approach and we will not use a Bayesian network 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. Unlike Bayesian networks, we do not store the probability that a word is relevant given that an adjacent in the graph word is unrelated. The reason is that the later probability will not help us compute the similarity between words in the graph.

The idea of creating a graph that stores the degree of semantic similarity between words is not new. For example, Simone Ponzetto and Michael Strube show how to create a graph that only represents inheritance of words,[12,28] while Glen Jeh and Jennifer Widom show how to approximate the similarity between words based on information about the structure of the graph in which they appear.[9] 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[35] 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[3] 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 papers by Danushka Bollegala, Yutaka Matsuo and Mitsuru IshizukaRef[1] and by Swarnim Kulkami and Doina Caragea.[13] The first paper searches for lexicographical patterns between the words using a search engine. For example, in order to compute the similarity between the words "dog" and "cat" the system will search the Internet for the phrase "dog is a cat", among others. The second paper uses the Internet to create a *concept cloud* around each word and then compares the two concept clouds. For example, the word "feline" is part of the concept cloud for the word "cat". 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 one of the senses 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[11] (stands for term frequency – inverse document frequency) to compute the strength of the relationship. In the TF-IDF model, if the word "ice" appears two times in the definition of one of the senses 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 senses. For example, if most senses contain the word "ice" as part of their definition, then the fact that one of the senses of the word "hockey" contains this word is not consequential. Conversely, if the word "ice" appears only in the definition of few senses, then the fact that the definition of one of the senses of the word "hockey" contains the word "ice" in its definition is statically meaningful.

Note that plenty of research effort has recently focused on using a description language, such as OWL,[37] to describe document resources. A semantic query language, such as SPARQL[30] (a recursive acronym that stands for SPARQL Protocol and RDF Query Language), 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.)

Lastly, note that the similarity graph can be applied to the problem of query expansion in natural language search systems.[29] For example, a user may search for "Mediterranean Restaurants". A smart search engine needs to expand the search query and also search for Egyptian, Moroccan, Syrian, and Turkish restaurants, among others. This expansion is based on the knowledge in the similarity graph.

3. Creating the Similarity Graph

3.1. *About WordNet*

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 the 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. Consider the word "chair". One of its senses has the definition: "a seat for one person, with a support for the back" and the example use: "he put his coat over the back of the chair and sat down". Two other senses of the word have the definitions: "the position of a professor" and "the officer who presides at the meetings of an organization". We will mine these textual descriptions to extract evidence about the strength of the relationship between the initial word and the words that appear in the definition and example use of its senses. Note that WordNet also provides information about the frequency of use of each sense. This represents the popularity of the sense in the English language relative to the popularity of the other senses of the word form. For example, the first sense of the word "chair" (a seat for one person, with a support for the back) is given a frequency of 35, the second sense (the position of a professor) is given frequency of just two, while the third sense (the officer who presides at the meetings of an organization) is given a frequency of one.

WordNet 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 hypernym and hyponym

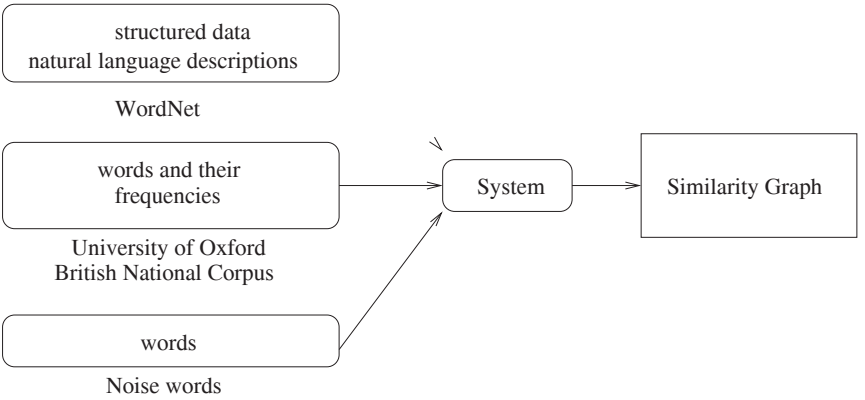


Fig. 1. Input and output of the graph creation system.

relationship between nouns. The *hypernym* relationship corresponds to the "kind-of" relationship. For example, "canine" in a hypernym of "dog". The *hyponym* relationship is the opposite. For example, "dog" is a hyponym of canine. WordNet also provides information about the meronym and holonym relationship between noun senses. The *meronym* relationship corresponds to the "part-of" relationship. Note that WordNet provides three types of meronyms: *part*, *member*, and *substance*. The three types of meronyms can be explained with the following examples: a "tire" is part of a "car", "car" is a member of "traffic jam", and a "wheel" is made from "rubber", respectively. The *holonym* relationship is the reverse of the meronym relationship. For example, "building" is a holonym of "window". For verbs, WordNet defines the hypernym and troponym relationship. X is a hypernym of Y if performing X is one way of performing Y. For example, "to perceive" is a hypernym of "to listen". 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". Lastly, WordNet defines the *related to* and *similar to* relationship between adjective senses, which are self explanatory. We will use all this structured information from WordNet as evidence about the degree of semantic similarity between senses.

3.2. *Description of the input and output of the graph creation system*

The input and output of the system are depicted in Fig. 1. Our software system uses information from the University of Oxford British National Corpus.[2] The corpus contains information about the frequency of use of a little over 200 000 of the most common words in the English language. This information, together with the sense frequencies in WordNet, helps us determine the probability that the user is interested in a sense given that they are interested in an adjacent in the similarity graph sense. For example, all other things being equal, it is more likely that a user will be interested in a sense that is more popular. Our computer system also takes

as input a list of about 100 noise words from the English language. When we process natural text, such as the description or example use of a sense, we ignore the noise words because they provide little evidence about the strength of semantic similarity between words.

The output of the 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 an 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 British National Corpus, and our list of noise words.

Before describing how the graph is created, we formally define its meaning.

Definition 3.1. The nodes in the graph represent either word forms or senses. The label of a word form node is the word form. The label of a sense node is the definition of the sense. An edge in the graph between node n_1 and node n_2 with weight p denotes that the probability that a user is interested in n_2 given that they are interested in n_1 is equal to p. We will sometimes use $P(n_2|n_1)$ to denote the weight of the edge between n_1 and n_2.

When clear from the context, we will refer to the relationship between two nodes as the *conditional probability*. Note that this function is not a similarity metric because it is asymmetric. However, as we will see in Section 4, it can be used to derive such a metric.

3.3. *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. The label of each node is the word form from which it originates in all lowercase letters. This helps us to avoid storing the same word in the graph multiple times, but with different capitalization of letters. The implementation is written in Java and we store the nodes using the Java hash structures from the standard library. For comparison, we also tried using Berkeley DB,[21] which is a popular key-value store, Neo4J,[38] which is a graph database, and MySQL, which is a full-fledged database management system. Unfortunately, the performance of the alternatives was not satisfactory.

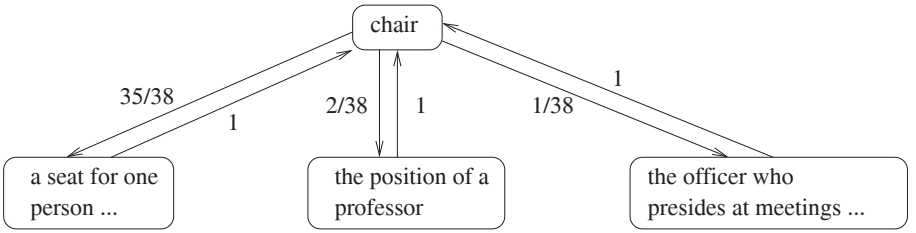


Fig. 2. Example edges between a word form and its senses.

3.4. *Processing the senses*

We next examine all the word form nodes in the graph (i.e., all the nodes that we have created so far) and use JAWS,[31] 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.

In the rest of this section, we will describe how the algorithm creates the edges and assigns weights to them. Sometimes, it is possible to create multiple edges between the same two nodes. In this case, we create a single edge that has weight that is the sum of all the weights. We restrict the value of the aggregate weight for an edge to one because this value represents a probability function.

We start by adding 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 Fig. 2. WordNet gives us the information about the frequency of use of each sense. The frequency of the first sense 35, the frequency of the second sense is 2, and the frequency of the third sense is 1. We will therefore create the outgoing edges from the node "chair" that are shown in Fig. 2. 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 piece of furniture on which we sit. The backward edges to the node "chair" represent the knowledge that all three senses represent the same word "chair". The weight is equal to 1 because if someone is thinking about one of the three senses in the figure, then they must be thinking about the word "chair".

3.5. *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 Fig. 3). The graph represents the connection between a sense and the non-noise words in its definition. We will assume that the first words in the definition of

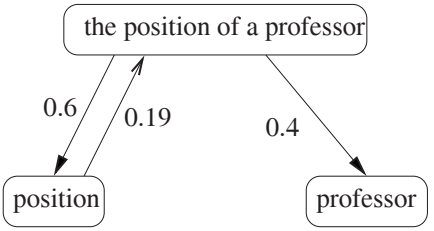


Fig. 3. Example edges between a sense and the word forms in its definition.

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. In the experimental section, we show that applying this heuristic results in slight improvement (up to 5%) in the quality of the data in the similarity graph (see Fig. 14).

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 20 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 word and the sense, 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.

Through this section, we will introduce multiple parameters, such as the value 0.6 that was described in the previous paragraph. In the experimental section, we will show how changing the values of these parameters affects the quality of the data. In particular, we show that we get 47% improvement in the quality of data by setting the parameters as described in this paper compared to setting all of them to one (see Fig. 16).

The *computeMinMax* function smoothens the value of the *ratio* parameter. For example, a word that appears as one of the 20 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 *computeMinMax* function is defined as follows.

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

Note that when $ratio = 0.5$, 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.

The affect of using the *computeMinMax* function is shown in the experimental section of the paper (see Fig. 15). In particular, using the function can lead to up to 9% improvement in the correlation of our data with the data from an experiment with human subjects.

Figure 3 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.

3.6. *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 4 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, the first words in the example use are not more important. Therefore, we ignore the order of the words in the example use of a sense. We will explain the backward edge from the word "coat" to the sense in the next subsection.

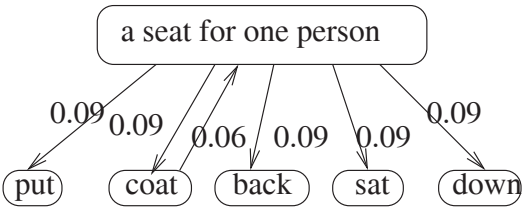


Fig. 4. Example edges between a word sense and the words in its example use.

3.7. *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 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 form will also be interested in one of the senses that have the word form in their definition. Here, we assume that the backward relationship is not as strong as the forward relationship. As an 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 Fig. 3 with weight that is equal to $computeMinMax(0, 0.3, 1/3) = 0.19$ (see Fig. 3).

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 all senses. The constant 0.1 relates to the probability that someone who is interested in a word form will also be interested in one of the senses that have the word form in their example use. This value is smaller than the value for the definition of a sense because the words in the definition of a sense are closely related to the meaning of the sense. As an 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 Fig. 4 with weight that is equal to $computeMinMax(0, 0.1, 1/3) = 0.06$ (see Fig. 4).

3.8. *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[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.[2]

Definition 3.2. Let m be a sense. 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 of 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 above formula approximates the size of a sense by looking at all the word forms that represent the sense and figuring out how much each word form contributes to the sense. 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 3.2, $p_m(president) = 3/31 = 0.096$ 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, such as "chairman", will also contribute to the size of the sense.

3.9. *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 15 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 who is interested in a sense is also interested in one of the sub-senses be equal to 0.9. This probability is high because, for example, someone who is interested in the first sense of the word "chair" is probably also interested in one of the chair types. 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 would be constructed as shown in Fig. 5. The weight of each edge is equal to 0.9 multiplied by the size of the sense

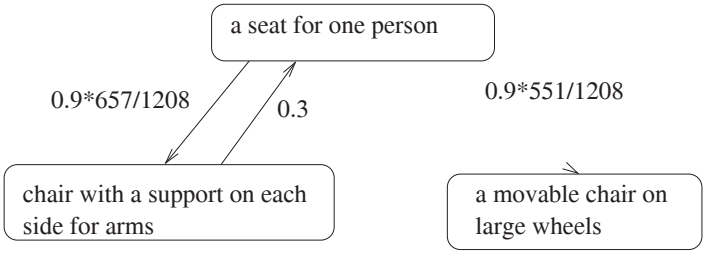


Fig. 5. Example edges between a word sense and its hyponyms.

and 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. Note that here we do not apply the *computeMinMax* function. The reason is that the function is relevant only when it comes to words that are contained in a text.

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 first sense of the word chair. However, this probability is not a function of the different hypernyms of the sense. Figure 5 shows an example of how the edge weights are computed.

We next consider the *meronym* (a.k.a. part-of) relationship between nouns. Note that we do not make a distinction between the three types of meronyms (part, member, and substance) and process them identically. WordNet contains 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 shown in Fig. 6. In general, the weight of a forward edge is

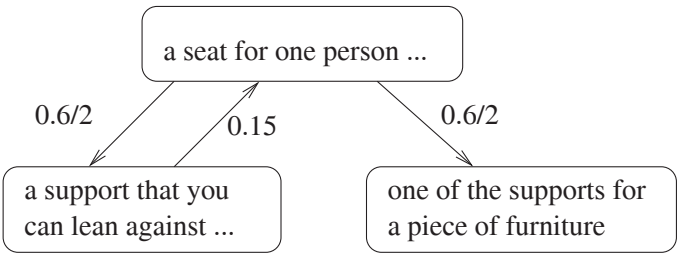


Fig. 6. Representing meronyms and holonyms.

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 reasoning behind the formula is that the more meronyms a sense has, the less likely is that we are interested in a specific one of the meronyms.

We also represent the *holonym* (a.k.a. contains) 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". 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. For example, the fact that someone is interested in the first sense of the word "window" does not translate in strong confidence that they are also interested in the whole building. For our running 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 Fig. 6).

3.10. *Processing structured knowledge about verbs*

We will first represent the *troponym* (a.k.a. doing in some manner) relationship for verbs. 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 are 18, 1, and 95, respectively, then we will create the edges that are shown in Fig. 7. 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 Fig. 7.

The hyponym and hypernym relationships are defined not only for nouns, but also for verbs. The two relationships are the reverse of each other. In other words, if X is a hyponym of Y, then Y is a hypernym of X. The hypernym relationship for verbs corresponds to the "one way to" relationship. For example, the verb "perceive" is the hypernym of the verb "listen" because one way of perceiving

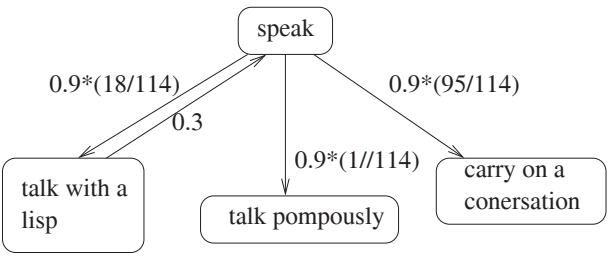


Fig. 7. Representing troponyms.

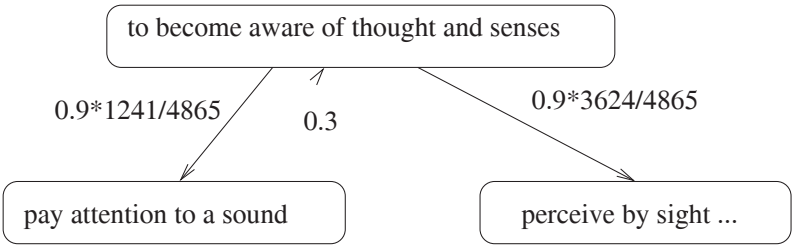


Fig. 8. Representing hyponyms and hypernyms between verb senses.

something is by listening. As expected, the verb "listen" is a hyponym of the verb "perceive". The first sense of the word "perceive" is "to become aware of through the senses". Suppose that the first senses of the verbs "listen" and "see" are the only hypernyms of the verb.

We will assume that the probability that someone who is interested in a verb sense is also interested in one of the hyponym senses be equal to 0.9. This probability is high because, for example, someone who is interested in perceiving is probably also interested in one of the ways to perceive. 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 "listen" is 1241 and the frequency of "see" is 3624. Since both senses are associated with a single word form, we do not need to consider the frequency of use of each sense. If "perceive" and "see" were the only hyponyms of the sense "to become aware of thought and senses", then the corresponding part of the similarity graph will be constructed as shown in Fig. 8. The weight of each edge is equal to 0.9 multiplied by the size of the sense and 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. The weight of each hypernym sense will be equal to 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 "see", then they may be also interested in the first sense of the word perceive. However, this probability is not a function of the different hypernyms of the sense.

3.11. *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 Fig. 9. This represents that there is a 60% probability that someone who is interested in an adjective is also interested in a "related to" adjective. This

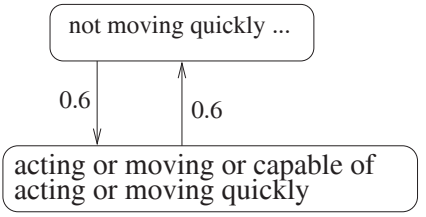


Fig. 9. Representing the *related to* relationship between adjectives.

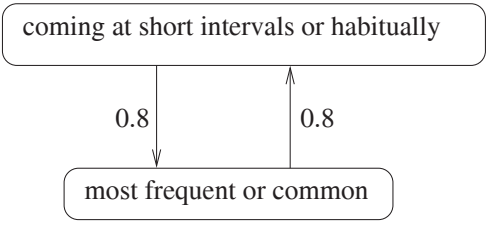


Fig. 10. Representing the *similar to* relationship between adjectives.

probability is high because the "related to" relationship represents relatively strong semantic similarity.

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 an 80% probability that someone who is interested in an adjective 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 Fig. 10. Note that both the "similar to" and "related to" relationships are both symmetric and therefore we draw edges both ways with the same weights.

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 nodes using the following formula.

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

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

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.

We also explore alternative ways of computing the similarity between two nodes. The reason is that it is not obvious that adding up the available evidence is the best approach. For example, consider the following alternative to Eq. (1).

$$A \rightarrow_s C = 1 - \prod_{Pt \text{ is a cycleless path from A to C}} (1 - P_{Pt}(C|A)) \,. \tag{3}$$

For example, suppose that there are two paths between the nodes "car" and "auto". The first path has weight of 0.6 and the second path has a weight of 0.5. In other words, we have evidence that someone who is interested in "car" is also interested in "auto" with probability 60% and 50%. If we combine the available evidence, then we get directional similarity of $1 - (1 - 0.6) \cdot (1 - 0.5) = 0.8$. This is the probability that we succeed in at least one of two independent tries, where the probability of success in the first try is 50% and the probability of success in the second try is 60%. In other words, every path brings new evidence that can increases the value of the directional similarity, but the value can never become more than one. We will show in the experimental section (see Fig. 17) that this is an inferior approach and can produce results that are up to 9% worse than applying the algorithm that aggregates the evidence from the paths between two nodes.

Next, we present two functions for measuring similarity. The linear function for computing the similarity between two word forms is shown in Eq. (4).

$$|w_1, w_2|_{lin} = min\left(\alpha, \frac{w_1 \rightarrow_s w_2 + w_2 \rightarrow_s w_1}{2}\right) * \frac{1}{\alpha} \,. \tag{4}$$

The minimum function was used in order to cap the value of the similarity function at one. α is a coefficient that amplifies the available evidence ($\alpha \leq 1$). The experimental section of the article shows how the value of α affects the correlation between the results of the system and that of human judgement. Note that when α is equal to one, then the function simply takes the average of the two numbers and caps the result at 1. The experimental section shows that using this function and the optimal value for α can increase the correlation with the results of an experiment that records the strength of semantic relationship between words as determined by human judgement by 53%.

Note that we take the average of the two directional similarity distances in order to determine the similarity score. Multiplying the two numbers is an 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 learning about an animal is particularly interested in the ostrich. Figure 18 in the experimental section shows that indeed multiplying the directional similarities is an inferior approach and can lead to results that have significantly lower correlation with the results of experiments with human subjects.

The second similarity function is inverse logarithmic, that is, it amplifies the smaller values. It is shown in Eq. (5). 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.

$$|w_1, w_2|_{log} = norm \quad \frac{-1}{log_2(min(\alpha, \frac{w_1 \rightarrow_s w_2 + w_2 \rightarrow_s w_1}{2}))} \quad . \tag{5}$$

A comparison between the linear and the logarithmic approach is presented in the experimental section (see Fig. 12). For an optimized value of α is chosen, the linear and logarithmic metric perform similarly and produce high correlation with results of experiments with human subjects.

Given two nodes, the similarity between them is computed by performing a depth-first traversal of the graph from one of the nodes. The algorithm runs in linear time relative to the number of visited nodes. When the weight of a path falls under 0.001, we prune out the path. In the experimental section (see Fig. 19), we examine the effect of lowering this threshold to 0.0001. Surprisingly, this worsens the quality of the data. The reason is that our algorithm can introduce semantic relationship with very low similarity score between words that are unrelated and adding up this evidence can lead to substantial similarity score between words that are unrelated.

In our experimental results, we only consider paths of lengths 100 edges or less. A path with length of more than 100 edges will provide little evidence about the relationship between two word forms.

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

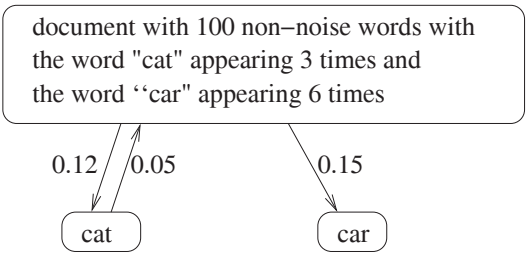


Fig. 11. Example of edges between a document node and word form nodes.

will then be measured by computing the semantic distance between the two nodes using the linear or logarithmic algorithm from the previous section.

In order to demonstrate our approach, consider a fictitious document that contains a total of 100 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 Fig. 11. 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.

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. Once the similarity graph is extended with the documents, the distance between two documents can be calculated using the $|\cdot|_{log}$ or $|\cdot|_{ling}$ metrics that are described in the previous section. The semantic distance between documents is a useful metric and it can be used, for example, to cluster the documents by applying the K-Means[17] clustering algorithm.

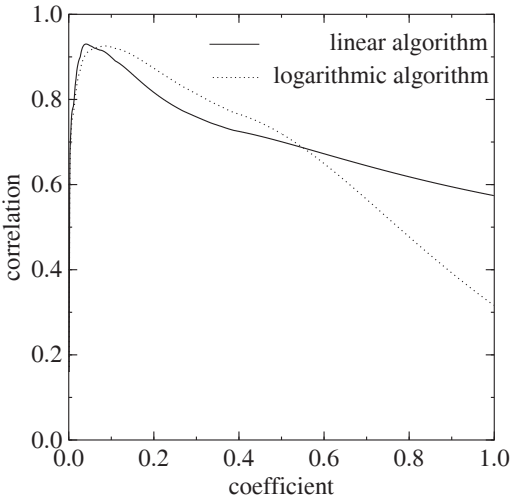


Fig. 12. A comparison of the linear and the logarithmic similarity metric.

6. Experimental Results

The system consists of 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.[31] All experiments were performed on a Silicon Graphics UV10 Linux machine. The Web interface of the system was created using JavaServer Pages (JSP).[32] It takes about seven minutes to build the similarity graph and save it to the hard disk. The size of the graph file is 92MB and it easily fits in main memory. The average time for computing the similarity distance between two words is one second.

6.1. *The Millers and Charles benchmark*

The goal of this section is to evaluate the quality of the data in the similarity graph. We use 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.

We ran both the linear and the logarithmic algorithm with all values of α between 0 and 1 in increment of 0.01. The results are shown in Fig. 12. As the figure suggests, the linear and the logarithmic algorithm perform similarly and produce highest correlation of 0.93. This is achieved when $\alpha = 0.041$ for the linear case and when $\alpha = 0.085$ for the logarithmic case. For these values of α, Table 1 shows the words of the benchmark and the similarity as determined by the human subjects of the study and by our system.

The results show very high correlation (0.93) with the results of the study with human subject. In the next subsection we also test the similarity graph with a different benchmark of 353 words and again achieve good results. Although, as

Table 1. Results on the Millers and Charles benchmark for the 28 words.

Word 1	Word 2	M&C	Linear	Logarithmic
car	automobile	3.92	1.00	1.00
gem	jewel	3.84	1.00	1.00
journey	voyage	3.84	1.00	1.00
boy	lad	3.76	1.00	1.00
coast	shore	3.7	1.00	1.00
asylum	madhouse	3.61	1.00	1.00
magician	wizard	3.5	0.72	0.70
midday	noon	3.42	1.00	1.00
furnace	stove	3.11	1.0	0.92
food	fruit	3.08	1.0	0.94
bird	cock	3.05	1.0	0.83
bird	crane	2.97	0.64	0.68
tool	implement	2.95	1.00	1.00
brother	monk	2.82	1.0	0.88
crane	implement	1.68	0.04	0.38
lad	brother	1.66	0.41	0.60
journey	car	1.16	0.67	0.68
monk	oracle	1.1	0.17	0.34
food	rooster	0.89	0.32	0.57
coast	hill	0.87	0.27	0.58
forest	graveyard	0.84	0.04	0.39
monk	slave	0.55	0.03	0.37
coast	forest	0.42	0.11	0.46
lad	wizard	0.42	0.04	0.38
chord	smile	0.13	0.04	0.38
glass	magician	0.11	0.03	0.37
noon	string	0.08	0.02	0.36
rooster	voyage	0.08	0.06	0.41

explained in the introduction, the main purpose of the similarity graph is not to compute the similarity between words, the results on these benchmarks give us confidence about the quality of the data in the similarity graph and its usefulness for other applications, such as semantic search.

Table 2 shows how our results compare with other proposals for extracting semantic similarity between word forms from WordNet. As the table suggests, both

Table 2. Correlation results with the Millers and Charles benchmark.

Algorithm	Correlation
Hirst and St-Onge[7]	0.74
Leacock and Chodorow[14]	0.82
Resnik[24]	0.77
Jiang and Conrath[10]	0.85
Lin[15]	0.83
$\|\cdot\|_{lin}$	0.93
$\|\cdot\|_{log}$	0.93

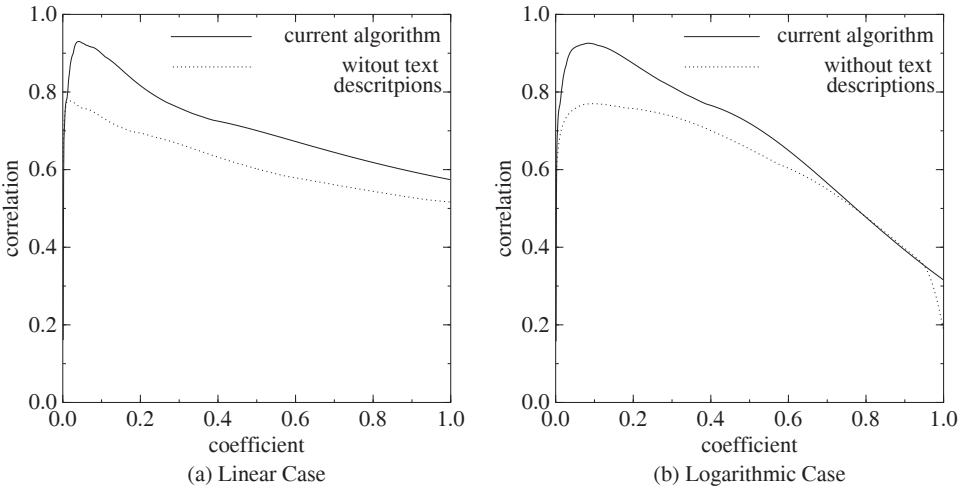


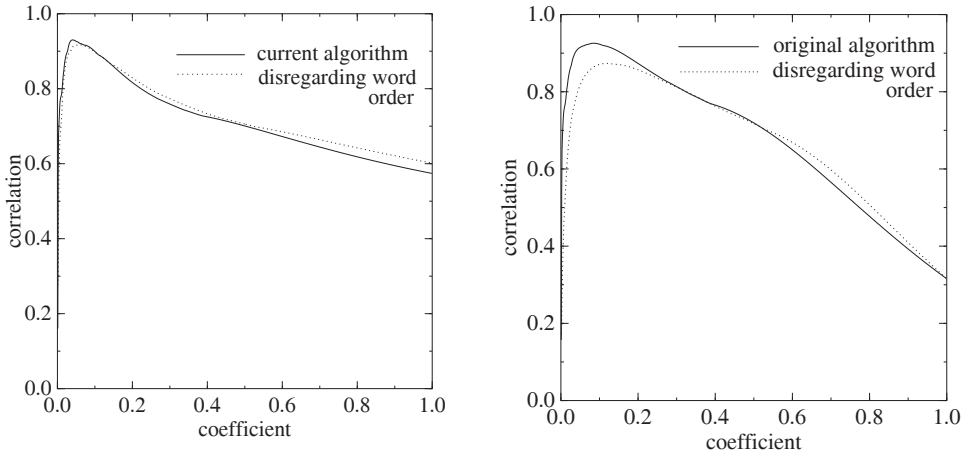
Fig. 13. The effect of processing textual information.

our algorithms produce better results (i.e., closer correlation with the results from the experiment with human subjects in Ref. 19) than existing algorithms. A correlation score of 0.93 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.

One of the reasons for the high quality of data in the similarity graph is that we consider not only structured knowledge, such as the hypernym relationship for nouns, but also the natural language descriptions from the definition and example use of senses. For example, Fig. 13 shows that the quality of the data will significantly deteriorate if natural text is not processed by our algorithm. Similarly, as Fig. 14 suggests, the approach of giving special preference to the first words in the definition of a sense also improves the quality of the data in the similarity graph.

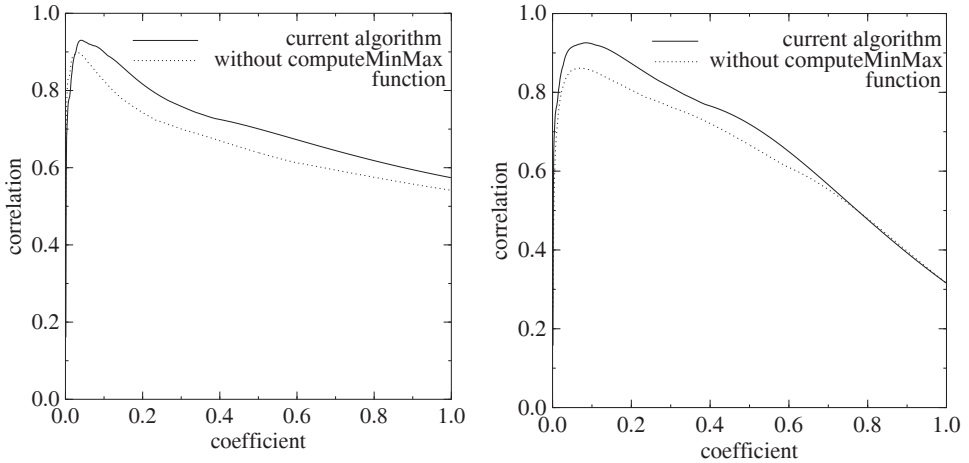
Throughout the paper, we use several heuristics and parameters to create the similarity graph. Here, we briefly examine the effect of some of these heuristics and parameters. For example, we use the *computeMinMax* function when we consider a word that appears in a text. Obviously, the importance of a word in a small text is greater than the importance of a word in a large text. However, this correlation is not linear. Just because the size of a document doubles the importance of a word in it does not decrease by half. We therefore adopt the *computeMinMax* function that decreases the rate of growth of the importance of a word in a text as the size of the text grows. Figure 15 shows the improvement in the quality of the data in the similarity graph that is achieved by using the *computeMinMax* function.

In our algorithm, we use ten concrete numbers to approximate the probability coefficients in the similarity graph. Unfortunately, experimentally determining the



(a) Linear Case (b) Logarithmic case

Fig. 14. The effect of considering the word ordering in the definition of a sense.



(a) Linear Case (b) Logarithmic Case

Fig. 15. The effect of using the *computeMinMax* function.

optimal value for each parameter is not feasible because it takes substantial time to construct the similarity graph for each value of the parameters. Here, we simply show that setting this parameters to the correct value is important. For example, if all the parameters are set to 1, then we will get the results that are shown in Fig. 16.

When computing the similarity between two word forms, our algorithm starts by finding the words in the graph. This is possible because the labels of the nodes in the graph are unique. We then consider all cycles paths between the two words. Every

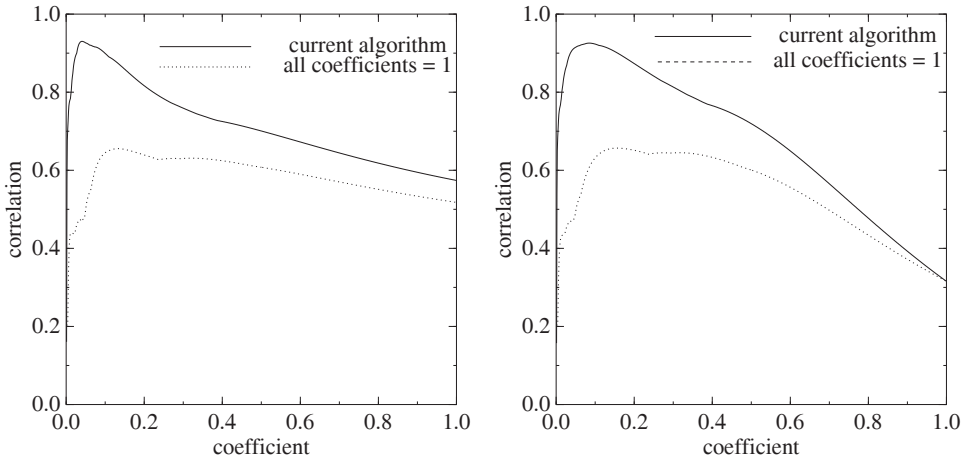


Fig. 16. The effect of setting all parameters to 1.

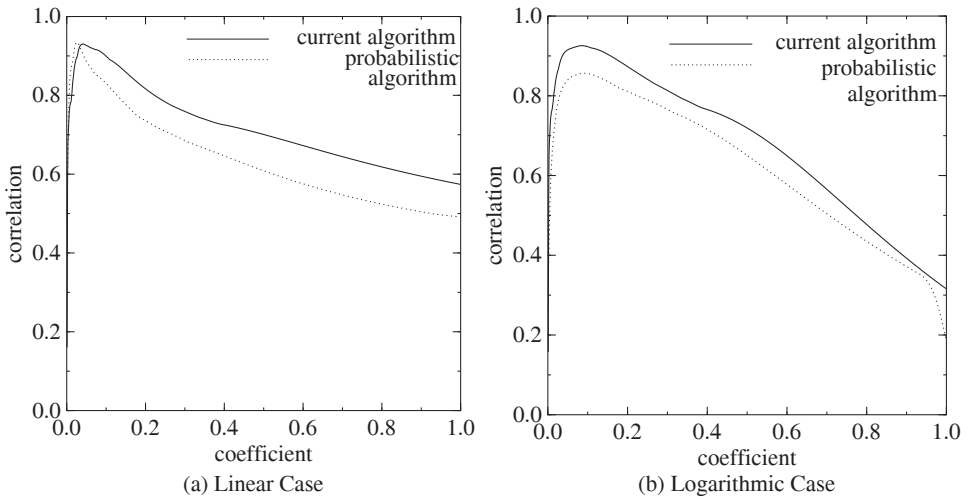


Fig. 17. The effect of applying the alternative probability-based similarity metric.

single path provides evidence about the relationship between the two words. Our algorithm adds this available evidence. A different approach would be to consider each piece of evidence as an independent event and compute the probability that one of the events happens. However, as shown in Fig. 17, this approach produces worse data quality in the similarity graph.

After we compute the directional similarity between two nodes in the graph, our algorithm takes the average of the two numbers. An alternative will be to multiply the two numbers. However, this approach will not produce good results because one

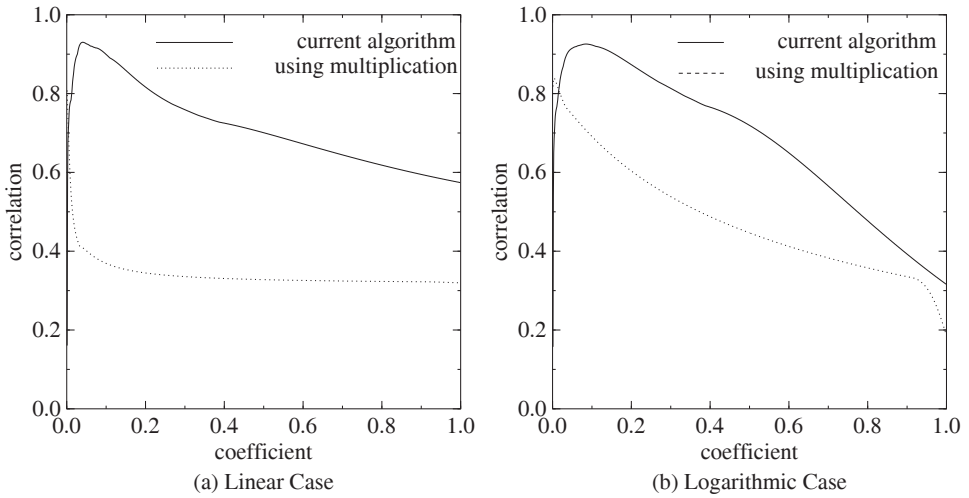


Fig. 18. The effect of multiplying the directional similarities.

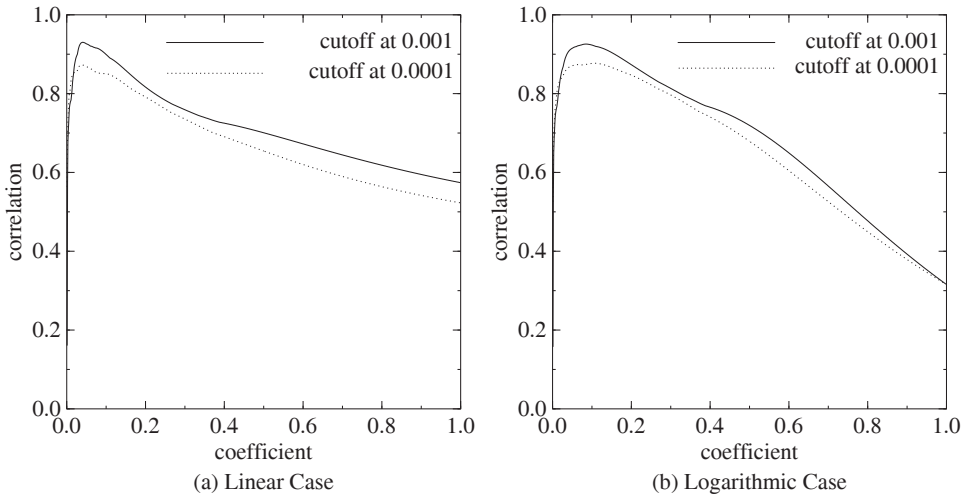


Fig. 19. The effect of using cutoff 0.0001 instead of 0.001.

of the numbers being small does not imply that the semantic similarity between the word forms should be low. Figure 18 confirms this reasoning.

Lastly, when computing the directional similarity between two nodes, we prune our paths when the similarity distance decreases below 0.001. The reason is that we do not want weak evidence to contribute to the similarity score. Figure 19 shows that if we lower this threshold to 0.0001, then the quality of the data will worsen.

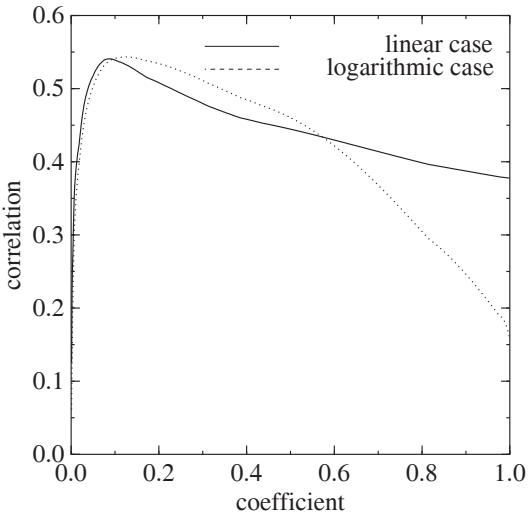


Fig. 20. Comparison of the linear and the logarithmic case for the WordSimilarity-353 benchmark.

6.2. *WordSimilarity-353 benchmark*

In the previous subsection, we experimentally determined the best possible value for the α coefficient for the linear and the logarithmic case. We also experimentally justified some of the heuristics that we used in our algorithm. In this section, we will apply the algorithm to the WordSimilarity-353 dataset.[4] It contains 353 word pairs. Thirteen humans were used to rate the similarity between each pairs 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.

As was the case for the previous benchmark, we ran the linear and the logarithmic algorithm with different values of α between 0 and 1 with increment of 0.01. The results are shown in Fig. 20. Again, the performance of the linear and the logarithmic algorithm are similar and both of them achieve correlation of 0.54 with the WordSimilarity-353 benchmark. Note that the correlation is lower than the correlation with the Millers and Charles benchmark because of the greater number of word pairs. The highest correlation is achieved for $\alpha = 0.087$ for the linear case and $\alpha = 0.127$ for the logarithmic case. These values are similar to the best performing α values from the previous benchmark. Indeed, if $\alpha = 0.1$, then the correlation drops to 0.9 and 0.92 for the linear and the logarithmic case, respectively, for the Miller and Charles benchmark. However, the correlation stays at 0.54 for both the linear and the logarithmic case on the WordSimilarity-353 benchmark.

Table 3 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 $\alpha = 0.1$. As the table suggests, our system produces better

Table 3. Correlation results with the WordSimilarity-353 benchamrk.

Algorithm	Correlation
Jarmasz[8]	0.27
Hirst and St-Onge[7]	0.34
Jiang and Conrath[10]	0.34
Strube and Ponzetto[36]	0.19–0.48
Leacock and Chodrow[14]	0.36
Lin[15]	0.36
Resnik[24]	0.37
Bollegala *et al.*[1]	0.50
$\|\cdot\|_{lin}$	0.54
$\|\cdot\|_{log}$	0.54

results than all other systems. Note that some algorithms use additional information from the web,[1] while our algorithm only uses information from WordNet. As our system is extended to use information from Wikipedia,[33] the quality of the data in the similarity graph is further improved.

7. Conclusion and Future Research

We presented an algorithm for building a similarity graph from WordNet. We verified the data quality of the algorithm 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 both the Miller and Charles and WordSimilarity-353 word pairs benchmarks. We believe that we outperformed existing algorithms because our algorithm processes not only structured data, but also natural language. Our algorithm also adopts several heuristics and we experimentally showed how these heuristics positively affect the quality of the data in the similarity graph. Augmenting these heuristics with new ones is an area for future research.

In this paper we experimentally verified the quality of the data in the similarity graph. Our plan for future research is to use the similarity graph to create a suite of semantic applications. We believe that the similarity graph can be used to not only find data that cannot be found by performing keywords-based search, but it can also help us achieve good ranking of the query result based on semantic relevance. Another application of the similarity graph is document clustering based on the similarity of the words and phrases in the documents.

We also plan on extending the similarity graph using freely available information sources. For example, Wikipedia can be used to compute the strength of the semantic relationship between common phrases. Available OWL ontologies can also be used to extend the similarity graph with domain-specific knowledge.

References

1. 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).
2. L. Burnard, Reference Guide for the British National Corpus (XML Edition) (2007), http://www.natcorp.ox.ac.uk.
3. R. L. Cilibrasi and P. M. Vitanyi, The Google Similarity Distance. *IEEE ITSOC Inforamtion Theory Workshop* (2005).
4. 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) (January 2002) 116–131.
5. C. Fox, Lexical analysis and stoplists, *Information Retrieval: Data Structures and Algorithms* (1992), pp. 102–130.
6. W. Frakes, Stemming algorithms, *Information Retrieval: Data Structures and Algorithms* (1992), pp. 131–160.
7. G. Hirst and D. St-Onge, Lexical chains as representations of context for the detection and correction of malapropisms, *Fellbaum* (1998), pp. 305–332.
8. M. Jarmasz, Roget's Thesaurus as a Lexical Resource for Natural Language Processing, Master's thesis, University of Ottawa (1993).
9. G. Jeh and J. Widom, SimRank: A measure of structural-context similarity, *Proc. of the Eight ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining* (2002), pp. 538–543.
10. J. Jiang and D. Conrath, Semantic similarity based on corpus statistics and lexical taxonomy, *Proc. on Int. Conf. on Research in Computational Linguistics* (1997), pp. 19–33.
11. K. Jones, A statistical interpretation of term specificity and its application in retrieval, *Journal of Documentation* **28**(1) (1972) 11–21.
12. R. Knappe, H. Bulskov and T. Andreasen, Similarity graphs, *Fourteenth Int. Symp. on Foundations of Intelligent Systems* (2003).
13. S. Kulkami and D. Caragea, Computation of the semantic relatedness between words using concept clouds, *Int. Conf. of Knowledge Discovery and Information Retrieval* (2009).
14. C. Leacock and M. Chodorow, Combining local context and WordNet similarity for word sense identification, *WordNet: An Electronic Lexical Database* (1998), pp. 265–283.
15. D. Lin, An Information-theoretic definition of similarity, *Proc. of the Fifteenth Int. Conf. on Machine Learning* (1998), pp. 296–304.
16. V. Lopez, M. Fernández, E. Motta and N. Stieler, PowerAqua: Supporting users in querying and exploring the semantic web content, *Semantic Web — Interoperability, Usability, Applicability an IOS Press Journal* (2010).
17. J. B. MacQueen, Some methods for classification and analysis of multivariate observations, *Proc. of 5th Berkeley Symp. on Mathematical Statistics and Probability* (1967), pp. 281–297.
18. M. F. Porter, An algorithm for suffix stripping, *Readings in Information Retrieval* (1997), pp. 313–316.
19. G. Miller and W. Charles, Contextual correlates of semantic similarity, *Language and Congnitive Processing* **6**(1) (1991) 1–28.
20. G. A. Miller, WordNet: A lexical database for English, *Communications of the ACM* **38**(11) (1995) 39–41.
21. Oracle, Berkeley DB (2014), http://www.oracle.com.

22. R. Pan, Z. Ding, Y. Yu and Y. Peng, A Bayesian network approach to ontology mapping, *Proc. of the Fourth International Semantic Web Conference* (2005).
23. J. Pearl, Bayesian networks: A model of self-activated memory for evidential reasoning, *Proc. of the 7th Conf. of the Cognitive Science Society* (University of California, Irvine, CA, 1985), pp. 329–334.
24. P. Resnik, Using information content to evaluate semantic similarity in a taxonomy, *Int. Joint Conf. on Artificial Intelligence* (1995), pp. 448–453.
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) (1989) 17–30.
26. Q. Rajput and S. Haider, Use of Bayesian networks in information extraction from unstructured data sources, *Proc. of Int. Conf. on Ontological and Semantic Engineering* (2009), pp. 325–331.
27. RDF Wordking Group, Resource Description Framework (RDF) (2014), http://www.w3.org/RDF/.
28. Simone Paolo Ponzetto and Michael Strube. Deriving a Large Scale Taxonomy from Wikipedia. *22nd International Conference on Artificial Intelligence*, 2007.
29. S. Simske, I. Boyko and G. Koutrika, Multi-engine search and language translation, *ExploreDB* (2014).
30. E. Sirin and B. Parsia, SPARQL-DL: SPARQL query for OWL-DL, *3rd OWL: Experiences and Directions Workshop* (*OWLED*) (2007).
31. B. Spell, Java API for WordNet Searching (JAWS) (2009), http://lyle.smu.edu/∼tspell/jaws/index.html.
32. L. Stanchev, Similarity Software (2012), http://softbase.ipfw.edu:8080/Similarity.
33. L. Stanchev, Creating a phrase similarity graph from Wikipedia, *Eight IEEE Int. Conf. on Semantic Computing* (2014).
34. L. Stanchev, Creating a similarity graph from WordNet, *Fourth Int. Conf. on Web Intelligence, Mining and Semantics* (2014).
35. M. Steyvers and J. Tenenbaum, The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth, *Cognitive Science* **29**(1) (2005) 41–78.
36. M. Strube and S. P. Ponzetto, Wikirelate! Computing semantic relatedness using wikipedia, *Association for the Advancement of Artificial Intelligence Conference* (2006).
37. The World Wide Web Consortium, OWL Web Ontology Language Guide (2014), http://www.w3.org/TR/owl-guide/.
38. J. Webber and I. Robinson, *Graph Databases* (O'Reilly, 2013).
39. Z. Wu and M. Palmer, Verb semantics and lexcial selection, *Annual Meeting of the Association for Computational Linguistics* (1994), pp. 133–138.
40. D. Yang and D. M. Powers, Measuring semantic similarity in the taxonomy of WordNet, *Australian Computer Science Conference* (2005), pp. 315–322.