Stanchev, 2016. Published in *International Journal for Computers & Their Applications*, 23(3):176-194.

Creating a Probabilistic Model for WordNet

Lubomir Stanchev *
California Polytechnic State University
San Luis Obispo, California, 93407, USA

Abstract

We present a probabilistic model for extracting and storing information from WordNet and the British National Corpus. We map the data into a directed probabilistic graph that can be used to compute the conditional probability between a pair of words from the English language. For example, the graph can be used to deduce that there is a 10% probability that someone who is interested in dogs is also interested in the word "canine". We propose three ways for computing this probability, where the best results are achieved when performing multiple random walks in the graph. Unlike existing approaches that only process the structured data in WordNet, we process all available information, including natural language descriptions. The available evidence is expressed as simple Horn clauses with probabilities. It is then aggregated using a Markov Logic Network model to create the probabilistic graph. We experimentally validate the quality of the data on five different benchmarks that contain collections of pairs of words and their semantic similarity as determined by humans. In the experimental section, we show that our random walk algorithm with logarithmic distance metric produces higher correlation with the results of the human judgment on three of the five benchmarks and better overall average correlation than the current state-of-the-art algorithms.

Key Words: Semantic similarity, probability-based semantic similarity and distances, Markov logic network for representing WordNet data, semantic similarity benchmarks for WordNet.

1 Introduction

Tens of scientists have spent decades to develop WordNet [22]. This word corpus contains very accurate information about 150,000 word forms from the English language and their senses. A *word form* is a word or a short phrase, such as "sports utility vehicle". Every word form can have multiple senses and every sense can be represented by multiple word forms. For example "a seat for one person" is the most popular sense of the word "chair".

The first problem that we will solve in this article is to show how to map the data from WordNet into a probabilistic graph. The graph will contain a node for each word form and each sense in WordNet. A directed edge between two nodes will be labeled with the probability that a user who is interested in the source node would also be interested in the destination node. For example, based on the definition of the first sense of the word "chair", we can create an edge between this sense and the word "seat". WordNet also contains information about the relationships between senses, where this information will also be used in creating the probabilistic graph.

The second problem that we will address in the article is how to measure the semantic similarity between two word forms in the graph. We will show two algorithms that consider disjoint paths between the two nodes. The first algorithm simply multiplies the weights of the edges along a path, while the second algorithm is more sophisticated and uses the Markov Logic Network (MLN) model [30]. The third algorithm is a Monte Carlo approximation algorithm that performs random walks in the graph.

The third and last problem that we will examine is how to experimentally validate the quality of the data in the graph and the quality of the semantic similarity algorithm using multiple benchmarks. As we will show in the next paragraph, the probabilistic graph has many applications. However, its usefulness is limited by the quality of the data in the graph. We will examine five benchmarks that have 201, 28, 65, 65, and 665 pairs of words, respectively. Each pair of words was given to multiple people and the average of the semantic similarity, as determined by their judgment, was recorded. We will compare the results of our three algorithms to that of 16 algorithms that form the current state-of-the-art in computing semantic similarity between words. Specifically, we will show that one of our algorithms produces higher correlation than the other 16 algorithms on three of the five benchmarks. Moreover, this algorithm has the highest average correlation over the five benchmarks.

The probabilistic graph has multiple applications. For example, in this article and in [43, 46], we focus on computing the degree of semantic similarity between a pair of word forms. In [45], we show how a probabilistic graph can be used to perform semantic search. This means that given a textual query, we can return documents that contain related words. For example, if we know that there is a 20% change that a

*Computer Science Department. Email: stanchev@gmail.com.

user who searches for cats will find documents that contain the word "pet" relevant, then we can return such documents as part of the query result. The documents that are returned are ranked based on the probability of being relevant to the input query, where the probabilistic graph can help us compute these probabilities. Lastly, [47] shows how a probabilistic graph can be used to perform semantic document clustering. For example, an online store can use the probabilistic graph to cluster the products that are offered in different categories. Two products should appear in the same category only when they have textual descriptions that contain words that are similar based on the semantic similarity distance that can be computed from the probabilistic graph.

Note that all the applications of the probabilistic graph rely on an efficient and precise algorithm for computing the conditional probability of a word form in the graph being relevant given that a different word form in the graph is relevant. The semantic similarity between two nodes can be computed as a function of the average of the probability of the first word form being relevant given that the second word form is relevant and the reverse. Since most of the relationships in the graph are between senses and not between word forms, we will present an algorithm that explores all the paths between two word form nodes in the graph in order to compute the conditional probability of the second word form being relevant given that the first word form is relevant.

Converting WordNet in a computer-friendly format is a daunting task because WordNet contains heterogeneous data. While there are a plethora of algorithms that process structured information [15, 37] and textual information [3, 16], experimental results have shown that processing both types of information yields the best results (e.g., [44]). The fact that processing natural language is intrinsically hard for computers makes the problem even harder. Although significant effort has been put in automated natural language processing (e.g., [9, 10, 25]), current approaches fall short of understanding the precise meaning of human text. In fact, the questions of whether computers will ever become as proficient as humans in understanding natural language text is an open problem. Lastly, note that the problem of computing the conditional probability of a word form in the graph being relevant given that a different word form in the graph is relevant is not trivial. As [46] shows, we can use the MLN model to compute the conditional probability along a single path. However, when there are multiple interweaving paths between the two nodes, exact computation of the conditional probability becomes computationally intractable.

To the best of our knowledge, our previous research in the area [46, 43, 45] is the only study on how structured and unstructured information from WordNet can be combined to capture the semantic relationship between word forms in a probabilistic model. Other approaches that extract information from WordNet (e.g., [18, 15, 37]) only consider the structured information in WordNet. However, we believe it is beneficial to capture all the information in WordNet, including the natural language text descriptions for the definition and example use of senses. Most existing research does not consider this information because natural language text is intrinsically hard to process.

The algorithm for creating the probabilistic graph first examines WordNet and creates a node for each word form and each sense. The label of a word form node is the word form and the label of a sense node is the definition of the sense. Next, we represent the relationship between nodes using logical formulas with weights. Following the MLN approach, the weight of a formula is equal to the natural logarithm of the odds of the formula being true. We slightly modify this expression to ensure that all the weights are positive. Our probability space consists of a random variable for each node in the graph and a single predicate called *rel* (stands for relevant). For example, we can model the relationship between the main sense of the word chair ("a seat for one person") and the first word in the definition of the sense using the Horn clause $rel(a\,seat\,for\,one\,person) \Rightarrow rel(seat)$. A weight will also be assigned to the formula and it will be based on how strongly we believe that someone who is interested in the sense will also be interested in the first non-noise word in its definition. After all the formulas are created, we draw edges between each pair of nodes that participate in a formula. We use the MLN model to aggregate the evidence about the conditional probability for each of these pairs. The resulting weight of an edge between two nodes is a normalized probability value that assures that the sum of the weights of all the edges that leave each node add up to one.

This article presents three algorithms for computing the conditional probability that a node is relevant given that a different node in the graph is relevant. Computing this probability using the MLN model without approximating the result is possible, but computationally intractable. Although [46] shows how to compute this probability along a single path, we are not aware of a practical algorithm that computes the probability when there are interweaving paths between the two nodes. In this article, we introduce a randomized Monte Carlo algorithm that performs multiple random walks, where the algorithm contains parameters for tuning the expected accuracy of the result.

In what follows, in Section 2 we cover related research. In Section 3, we present an overview of WordNet and our algorithm for creating the probabilistic graph. The main contributions of the article are in the next two sections. In Section 4, we present two existing algorithms and a novel Monte Carlo algorithm for computing the conditional probability between two nodes in the probabilistic graph. Section 5 shows previously unpublished experimental results that test the quality of the data in the probabilistic graph and the accuracy of the different algorithms for finding the conditional probability between two nodes on five different benchmarks. Lastly, Section 6 summarizes the article and suggests avenues for further research.

2 Related Research

First, note that this article builds on several previous papers by the same author. The paper [43] proposes how to build a similarity graph, where the weights of the edges in the graph correspond to the degree of directional semantic similarity between the nodes. However, the weight of the edges are not probabilities in the strict sense. The paper [46] extends this working by showing how to use the MLN model to convert the weights of the edges in the graph to strict probabilities. This article extends [46] by presenting more detailed introduction and related research sections. The algorithm for computing the probabilistic graph is slightly modified. However, the most important contribution is the new Monte Carlo algorithm for computing the conditional probability between two nodes in the probabilistic graph and the new experimental results that test the quality of the proposed model on five independent benchmarks and compare the results to 16 algorithms that form the current state-of-the-art in algorithms that compute semantic word similarity.

Existing research that applies Bayesian networks to represent knowledge deals with the uncertain or probabilistic information in the knowledgebase [26, 23]. Our approach slightly differs because we do not store the probability that a word form is relevant given that an adjacent word form in the graph is unrelated. We only store a single number along every edge (the conditional probability that the destination concept is relevant given that the source concept is relevant) and we do not store all the information that is needed to create the full joint distribution of the word forms. Our model is more compact and, as we will show in the experimental section, contains high quality data.

The idea of creating a graph that stores the degree of semantic similarity between word forms is not new. For example, Simone Ponzetto and Michael Strube show how to create a graph that only represents inheritance of words [15, 37]. Specifically, [28] proposed one of the first models that computes the information content by counting the number of occurrences of different words in the WordNet hierarchy. Alternatively, 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 [13]. 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 [48] because the latter does not assign weights to the edges of the graph.

In this article, we show a method that uses the probabilistic graph to measure the semantic similarity between word forms. However, there are alternative methods to measure the semantic similarity between word forms. The most notable approach is the Google approach [6] in which the similarity between two word forms is measured as a function of the number of Google results that are returned by each word form individually and the two word forms combined. Note that there is a second relevant paper by Google research [20]. The paper explains how input text can be used to train a two-layer neural network. Once trained, the neural network can be used to predict what words will appear together in a text. This differs from our approach because we are interested in the semantic similarity between words, where similar words do not necessarily appear in the same sentence.

Other approaches that rely on data from the Internet include papers by Danushka Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka [3] and by Swarnim Kulkami and Doina Caragea [16]. 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 computes the semantic distance between two words as a function of the distance between their 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 (i.e., is-a) relationship in WordNet. Second, they do not provide evidence about the strength of the relationship between the two word forms that are compared. In contrast, our approach can show the paths in the probabilistic graph between the two word forms, which serves as evidence that supports the similarity score.

Since the early 1990s, research on LSA (stands for *latent semantic analysis*) has been carried out [7]. The approach has the advantage of not relying on external information. Instead, it considers the closeness of word forms in text documents as proof of their semantic similarity. For example, LSA can be used to detect words that are synonyms [17]. This differs from our approach because we do not consider the closeness of the words in a document. For the most part, we process natural language text as a bag of terms, where the main exception is that we consider the order of the words in the definition of a WordNet sense when we create the logical formulas. The reason is that we assume that the first words in the definition of a sense are more important. The other difference is that our algorithm can extract overlapping terms from a text source. Although the LSA approach has its applications, we believe that using a high-quality word corpus, such as WordNet, is beneficial. Note as well that the LSA approach cannot be directly used to process structured knowledge.

Research from information retrieval is also relevant to creating and using the probabilistic 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 uses a model that is similar to TF-IDF [14] (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 two. 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 in its definition is inconsequential. Conversely, if the word "ice" appears only in the definition of a 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 a plethora of research effort has recently focused on using a description language, such as the *ontology web language* (OWL) [51], to describe resources. A semantic query language, such as SPARQL [39] (a recursive acronym that stands for SPARQL Protocol and RDF Query Language), can be used to search for relevant items. This research differs from our approach to semantic search in [45] because it does not provide ranking of the query result. At the same time, a SPARQL query returns exactly the resources that fulfill the query description. Alternatively, [45] returns resources that are related to the input query in ranked order. 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, there are papers that consider a hybrid approach for information retrieval using both an ontology and keyword matching. For example, [32] examines how queries can be expanded based on the information from an OWL knowledgebase. Alternatively, [49] proposes a ranking function that depends on the length of the logical derivation of the result, where the assumption is that shorter derivations will produce more relevant documents. Unfortunately, these approaches are only useful in the presence of an ontology and research on automatic annotation of resources with OWL descriptions is still in its early stages of development.

There has also been research in the area of combining a subset of OWL called RDF [27] (stands for Resource Description Framework) with information retrieval approaches, such as BM25F [31] (a version of the TF-IDF approach). For example, [2] shows how to use natural language to query RDF stores. Note that this is a keywords-matching search approach and it does not take into account that the same query can be phrased differently using different words and terms. There have also been several papers that explore how to rank the result of queries over RDF data. For example, [5] uses the TF-IDF algorithm to rank the result of an RDF query.

Lastly, note that the probabilistic graph can be applied to the problem of query expansion in natural language search systems [38]. 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 probabilistic graph.

3 Building the Probabilistic Graph

3.1 About WordNet

WordNet [22] 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 process these textual descriptions to extract evidence about the strength of the relationship between the initial word forms and the word forms that appear in the definition and example use of their 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 relationships between nouns. The *hypernym* relationship corresponds to the "kind-of" relationship. For example, "canine" in a hypernym of "dog". The *hyponym* relationship is the reverse. For example, "dog" is a hyponym of "canine". WordNet also provides information about the meronym and holonym relationships 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", a "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* relationships. 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* relationships between adjective senses, which are self explanatory. We will use all this structured information from WordNet as evidence about the degree of conditional probability between senses.

3.2 The Probabilistic Model

We create a random variable for each sense and each word form in WordNet. We will refer to a random variable by its label, where the label of a word form variable is the word form and the label of a sense variable is the definition of the sense. In order to avoid ambiguity, we convert all labels to lower case. In this model, each random variable will have a string label and no two random variables will have the same label.

We add a single predicate to the model. The name of the predicate is *rel* and it tells us if a word form or sense is relevant in the current world. Our model contains only logical formulas that are Horn clauses of the form: $rel(X) \Rightarrow rel(Y)$. We will add a weight to each logical formula, where the weight will be computed using the following expression.

$$w(rel(X) \Rightarrow rel(Y)) = ln(\frac{P^+(Y|X)}{1-P^+(Y|X)}) \quad (1)$$

Following the MLN model [30], the weight of a logical formula is equal to the natural logarithm of the odds of the formula being true, that is $ln(\frac{p}{1-p})$. However, this will allow formulas with negative weights, which is undesirable. When aggregating evidence, a MLN works by interpreting formulas with positive weights as positive reinforcement and formulas with negative weights as evidence why the formula does not hold. By making all weights positive, we ensure that all the formulas will have a positive contribution to the aggregated conditional probability between two concepts. Note that when we say that there is a 10% probability that the word "table" is relevant given that the word "chair" is relevant, we want this evidence to increase the conditional probability of the word "table" being relevant given the word "chair" is relevant. We make the weights positive by performing a linear transformation of the probability to the range $[0.5, 1]$. Specifically, we define P^+ as follows.

$$P^+(Y|X) = 0.5 + \frac{P^e(Y|X)}{2} \quad (2)$$

We use $P^e(Y|X)$ to denote our confidence of the formula being true and refer to this value as the *evidence probability*. For example, if we know that the evidence probability is 0.10 (i.e., we are 0.10 confident that someone who is interested in the word "chair" will also be interested in the word "table"), then $P^+(table|chair) = 0.55$ and the weight of the formula will be calculated as $ln(0.55/0.45) = 0.2$.

Note that the same formula can appear multiple times in our knowledgebase, but possibly with different weights. At the end of this section, we will show how we can apply the MLN model to aggregate multiple evidence about the conditional probability between two concepts. Before that, we describe an algorithm that models WordNet as a set of Horn clauses with weights. Note that, for the most part, we will only describe how to compute the evidence probability, where the weight of each formula can be computed using Equations 1 and 2.

3.3 Processing the Senses

We first show how to create logical formulas that show the relationship between a word form and all its senses. Consider the word chair and its three meanings: "a seat for one person", "the position of a professor" and "the officer who presides at meetings". Suppose that WordNet gives a frequency of 35, 2, and 1, respectively, for the three senses. We will then crate the following formulas and probabilities.

$$rel(chair) \Rightarrow rel(a\,seat\,for\,one\,person), (35/38)$$
$$rel(chair) \Rightarrow rel(the\,position\,of\,a\,professor), (2/38)$$
$$rel(chair) \Rightarrow rel(the\,officer\,who\,presides\,at\,meetings), (1/38)$$

Note that the word "chair" has three meanings. Based on the frequencies that we are given, the evidence probabilities for the three relationships are $35/38$, $2/38$, and $1/38$, respectively. Note that for each formula, we put the evidence probability in parentheses. We can then compute the weight of the formula using Equations 1 and 2. When we assign an actual weight to a formula, we omit the parenthesis around the number. In general, we will compute the evidence probability as the frequency of the sense divided by the sum of the frequencies of all the senses for the word form. Here is the general formula, where $\{sense_i\}_{i=1}^n$ are all the senses of the word form.

$$rel(word\,form) \Rightarrow rel(sense\,of\,the\,word\,form), \\ (\frac{frequency(sense)}{\sum_i^n frequency(sense_i)}) \quad (3)$$

In our example, we will also add the following formulas and weights. Since there are no parentheses, the expressions show weights and not evidence probabilities.

$$rel(a\,seat\,for\,one\,person) \Rightarrow rel(chair), 10$$
$$rel(the\,position\,of\,a\,professor) \Rightarrow rel(chair), 10$$
$$rel(the\,officer\,who\,presides\,at\,meetings) \Rightarrow rel(chair), 10$$

In general, we always add a formula with weight 10 between a sense and all the word forms that it represents. The general formula is shown next.

$$rel(sense\,of\,a\,word\,form) \Rightarrow rel(word\,form), 10 \quad (4)$$

The reason for this formula is that we have a very high degree of confidence that if a sense is relevant, then so are all the word forms that represent the sense. A weight of 10 corresponds to evidence probability of above 99.99%. Note that in a MLN we cannot assign an evidence probability of one to a formula because this translates to a weight that is equal to infinity.

3.4 Processing the Definitions of the Senses

We next show how to model the relationship between a sense and the non-noise word forms in its definition. Note that our algorithm uses a list of about one hundred noise words, such as "who", "where", "at", "about" and so on. 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 be left with two words: "position" and "professor". As a result, we will create the following formulas.

$$rel(the\,position\,of\,a\,profeesor) \Rightarrow rel(position), (0.6)$$
$$rel(the\,position\,of\,a\,profeesor) \Rightarrow rel(professor), (0.48)$$

The formulas represent the connection between a sense and the non-noise words in its definition. We assume that the first words in the definition of a sense are far more important than the later words. We will therefore multiply the probability by $coef = 1.0$ for the first non-noise word form and keep decreasing this coefficient by 0.2 for each sequential word form until the value of the coefficient reaches 0.2. We compute the evidence probability of each formula using the equation $coef * minMax(0, 0.6, ratio)$, where the variable *ratio* is calculated as the number of times the word form appears in the definition of the sense divided by the total number of non-noise words in the sense.

$$rel(sense) \Rightarrow rel(word\,form\,in\,the\,sense\,definition), (coef * minMax(0, 0.6, \frac{frequency\,of\,word\,form}{sum\,of\,frequencies})) \quad (5)$$

The third parameter of the *minMax* function expresses the importance of the word form in the definition of the sense. For example, if there are only two word forms in the definition of the sense, then they are both very important. However, if there are 20 word forms in the definition of the sense, then each individual word form is less important. The *minMax* function makes the difference between the two cases less extreme. Using this function, the evidence probability of the formula in the second case will be only roughly four times smaller than the evidence probability of the formula 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. We use the *minMax* function every time we compare the number of occurrences of a word form in a document compared to the total number of words in the document.

The *minMax* 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 form in the definition of a sense is not a reliable source of evidence about the relationship between the word form and the sense, the value of the second argument is set to 0.6. The constant 0.6 is related to the probability that someone who is interested in a sense will also be interested in one of the word forms in the definition of the sense. Note that throughout this paper we introduce multiple constants. In [44], we give experimental evidence why these constants are meaningful and produce good results.

Formally, the *minMax* function is defined as follows.

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

Note that when $ratio = 0.5$, 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 evidence probability to the formula.

In our example, $ratio = \frac{1}{2}$ and therefore $minMax(0, 0.6, ratio) = 0.6$. Therefore, the evidence probability of the first formula is $coef * 0.6 = 1 * 0.6 = 0.6$ and for the second formula: $coef * 0.6 = 0.8 * 0.6 = 0.48$. To summarize, we assume that the probability that a user is interested in a word form will be higher if : (1) the word form appears multiple times in the definition of the sense, (2) the word form is one of only few words in the definition of the sense, and (3) the word form is one of the first word forms of the definition of the sense.

3.5 Processing the Example Uses of a Senses

WordNet also includes example uses for each sense. In this subsection, we show how to represent this information as formulas with weights. For example, in WordNet the sentence "he put his coat over the back of the chair and sat down" is shown as an example use of the first sense of word "chair". Since the example use represents evidence that is weaker than the evidence from the definition of a sense, we will calculate the evidence probability as $minMax(0, 0.2, ratio)$. Here, the variable *ratio* is the number of times the word form 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 word forms in the example use of the sense. The following formulas are created from the first sense of the word "chair" and its example use. Note that the noise words have

been omitted.

$$rel(a\,seat\,for\,one\,person) \Rightarrow rel(put),\ (0.09)$$
$$rel(a\,seat\,for\,one\,person) \Rightarrow rel(coat),\ (0.09)$$
$$rel(a\,seat\,for\,one\,person) \Rightarrow rel(back),\ (0.09)$$
$$rel(a\,seat\,for\,one\,person) \Rightarrow rel(sat),\ (0.09)$$
$$rel(a\,seat\,for\,one\,person) \Rightarrow rel(down),\ (0.09)$$

The evidence probability is the same for all edges because all words appear once in the example use. For all words, the value of *ratio* is equal to $\frac{1}{5}$. Unlike the case with the definition of a sense, the first words in the example use are not considered to be more important. Therefore, we ignore the order of the words in the example use of a sense. The precise calculation for the evidence probability is $0.2 * (\frac{-1}{log_2(0.2)}) = 0.09$. The general formula is shown next.

$$rel(sense) \Rightarrow rel(word\,form\,in\,the\,example\,use\,of\,the\,sense),$$
$$(coef * minMax(0, 0.2, \frac{frequency\,of\,word\,form\,in\,example\,use}{sum\,of\,frequencies})) \quad (6)$$

3.6 Processing the Backward Relationships

We also create formulas for the probability that a sense is relevant given that a word form that appears in its definition is relevant. The evidence probability of the formula is computed as $minMax(0, 0.3, ratio)$, where 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 definition of all senses. 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" occurs as part of the definition of only three senses and exactly once in each definition, then we will add the following formula for the second sense of the word "chair". The evidence probability is computed as $minMax(0, 0.3, ratio) = 0.3 * \frac{-1}{log_2(\frac{1}{3})} = 0.19$ and the formula is as follows.

$$rel(position) \Rightarrow rel(the\,position\,of\,a\,professor),\ (0.19)$$

The general formula is shown next.

$$rel(a\,word\,form\,in\,a\,sense) \Rightarrow rel(sense), (minMax(0, 0.3,$$
$$\frac{frequency\,of\,word\,form\,in\,sense\,definition}{sum\,of\,frequencies}) \quad (7)$$

Similarly, we will create a formula that shows the conditional probability between a word form and a sense that contains the word form in its example use. The weight of an edge in this case will be computed as $minMax(0, 0.1, ratio)$. Here, the *ratio* parameter is the number of times the word form appears in the example use of the sense divided by the total number of occurrences in the example uses 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 more closely related to the meaning of the sense. As an example, if the word "coat" occurs as part of the example use of only three senses and exactly once in each sense, then we will add the following formula for the first sense of the word "chair". The evidence probability is computed as $minMax(0, 0.1, \frac{1}{3}) = 0.1 * \frac{-1}{log_2(\frac{1}{3})} = 0.06$. Recall that the example use of this sense is: "he put his coat over the back of the chair and sat down".

$$rel(coat) \Rightarrow rel(a\,seat\,for\,one\,person),\ (0.06)$$

The general formula is shown next.

$$rel(a\,word\,form\,in\,the\,example\,use\,of\,a\,sense) \Rightarrow rel(sense),$$
$$minMax(0, 0.1, \frac{frequency\,of\,word\,form\,in\,example\,use}{sum\,of\,frequencies}) \quad (8)$$

3.7 Populating the Frequencies 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 sense. We will next show how structured knowledge, such as the hyponym (a.k.a. kind-of) relationship between senses, can be represented as logical formulas. Most existing approaches [28] 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. The description of this corpus, as presented in [4], is: "The British National Corpus is a 100 million word collection of samples of written and spoken language from a wide range of sources, designed to represent a wide cross-section of British English, both spoken and written, from the late twentieth century."

Definition 1. *Let s be a sense. Let* $\{wf_i\}_{i=1}^{n}$ *be the word forms for that sense. We will use* $BNC(wf)$ *to denote the frequency of the word form in the British National Corpus. Let* $p_s(wf)$ *be the frequency of use of the sense s of the word form wf, as specified in WordNet, divided by the sum of the frequencies of use of all senses of wf (also as defined in WordNet). Then we define the* size *of s to be equal to* $|s| = \sum_{i=1}^{n} (BNC(wf_i) * p_s(wf_i))$.

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 *s*. According to Definition 1, $p_s(president) = 3/31 =$

0.096 because the frequency of s is 3 and the sum of all the frequencies is 31. Since the British National Corpus shows the frequency of the word "president" as 9781, the contribution of the word "president" to the size of the sense s is equal to $|s| = BNC(president) * p_s(president) = 9781 * 0.096 = 938.98$. Other word forms that represent the sense s, such as "chairman", will also contribute to the size of the sense.

3.8 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" and "wheelchair". We will add formulas that show the conditional probability 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 evidence probability of each formula, 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 we need to add the following formulas.

$$rel(a\,seat\,for\,one\,person) \Rightarrow rel(chair\,with\,support\,on\,each\,side\,for\,arms), (0.49)$$
$$rel(a\,seat\,for\,one\,person) \Rightarrow rel(a\,moveable\,chair\,mounted\,on\,large\,wheels), (0.41)$$

The evidence probabilities were computed as $0.9 * 657/1208 = 0.49$ and $0.9 * 551/1208 = 0.41$. In general, the evidence probability is computed as 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.

$$rel(sense) \Rightarrow rel(hyponym\,of\,the\,sense), \left(0.9 * \frac{|hyponym\,of\,the\,sense|}{\sum_{s\,is\,a\,hyponym\,of\,the\,sense} |s|}\right) \quad (9)$$

The idea is that the conditional probability for "bigger" senses will be bigger because it is more likely that a bigger sense is relevant. Note that here we do not apply the *minMax* function. The reason is that the function is only relevant when computing the ratio of the number of occurrences of a word form in text relative to the size of the text.

We will also create formulas 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 evidence probability for each formula will be the same and equal to the constant 0.3. This represents the probability that someone who is interested in a sense will be also interested in the hypernym of the 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. Next, we show the formula from our example.

$$rel(chair\,with\,support\,on\,each\,side\,for\,arms) \Rightarrow rel(a\,seat\,for\,one\,person), (0.3)$$

The general formula is shown below.

$$rel(sense) \Rightarrow rel(hypernym\,of\,the\,sense), (0.3) \quad (10)$$

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. For example, 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. Part of this information can be represented using the following equations.

$$rel(a\,seat\,for\,one\,person) \Rightarrow rel(a\,support\,that\,you\,can\,lean\,against), (0.3)$$
$$rel(a\,seat\,for\,one\,person) \Rightarrow rel(one\,of\,the\,supports\,for\,a\,piece\,of\,furniture), (0.3)$$

In general, we compute the evidence probability as $0.6/n$, where n is the number of meronyms of the sense. Here is the general formula.

$$rel(sense) \Rightarrow rel(meronym\,of\,the\,sense), \left(\frac{0.6}{number\,of\,meronyms\,of\,the\,sense}\right) \quad (11)$$

The constant 0.6 represents the probability that a user who 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 provides weaker evidence than the hyponym relationship. The reasoning behind the formula is that the more meronyms a sense has, the less likely it is that we are interested in a specific meronym.

We also represent the *holonym* (a.k.a. contains) relationship. 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 evidence probabilities 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 create the following formulas.

$$\begin{aligned} &rel(\textit{a support that you can lean against}) \Rightarrow \\ &\quad rel(\textit{a seat for one person}), (0.15) \\ &rel(\textit{one of the supports for a piece of furniture}) \Rightarrow \\ &\quad rel(\textit{a seat for one person}), (0.15) \end{aligned} \tag{12}$$

The general formula is shown next.

$$rel(sense) \Rightarrow rel(\textit{holonym of the sense}), (0.15)$$

3.9 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 (as determined by the formula in Definition 1), respectively, then we will create the following equations.

$$\begin{aligned} &rel(\textit{an exchange of ideas via conversation}) \Rightarrow \\ &\quad rel(\textit{talk with a lisp}), (0.14) \\ &rel(\textit{an exchange of ideas via conversation}) \Rightarrow \\ &\quad rel(\textit{talk pompously}), (0.01) \\ &rel(\textit{an exchange of ideas via conversation}) \Rightarrow \\ &\quad rel(\textit{carry on a conversation})\,(0.75) \end{aligned}$$

The left side of the formulas contains the first sense of the word "talk": "an exchange of ideas via conversation", while the right side of the formulas contains the senses for "lisp", "orate" and "converse". The first formula expresses the conditional probability between the senses for "talk" (an exchange of ideas via conversation) and "lisp". The evidence probability for the formula is equal to $0.9 * \frac{18}{114} = 0.14$. The constant 0.9 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 arrive at the expression $18/114$ by dividing the size of the sense by the sum of the sizes of all the troponym senses. The general formula is shown next.

$$rel(sense) \Rightarrow rel(\textit{troponym of the sense}), \\ (0.9 * \frac{|sense|}{\sum_{s \textit{ is a troponym of the sense}} |s||}) \tag{13}$$

We will also add formulas for the reverse relationship with evidence probability of 0.3. For example, we will add the following formula.

$$rel(\textit{talk with a lisp}) \Rightarrow rel(\textit{an exchange of ideas via} \ldots), (0.3)$$

This means that if someone is interested in one of the troponyms, then there is a 30% chance that they are also interested in the original verb. The general formula is shown next.

$$rel(\textit{troponym of the sense}) \Rightarrow rel(sense), (0.3) \tag{14}$$

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 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 "perceive".

We will assume that the probability that someone who is interested in a verb sense is also interested in one of the hyponym senses is 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 evidence probabilities of the formulas, 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 we will create the following formulas.

$$\begin{aligned} &rel(\textit{to become aware of thought and senses}) \Rightarrow \\ &\quad rel(\textit{pay attention to sound}), (0.23) \\ &rel(\textit{to become aware of thought and senses}) \Rightarrow \\ &\quad rel(\textit{percieve by sight}), (0.67) \end{aligned}$$

The evidence probability for each formula 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. For example, the evidence probability of the first formula is $0.9 * 1241/4865 = 0.23$ and the evidence probability of the second formula is $0.9 * 3624/4865 = 0.67$. The idea behind the formula is that the conditional probabilities to "bigger" senses will be bigger because it is more likely that they are relevant. The general formula is shown next.

$$rel(sense) \Rightarrow rel(\textit{hyponym of the sense}), \\ (0.9 * \frac{|sense|}{\sum_{s \textit{ is a hyponym of the sense}} |s|}) \tag{15}$$

We will use an evidence probability of 0.3 for the hypernym (the reverse of the hyponym) relationship. For example, the main sense of the verb "perceive" is a hypernym of the main senses of the verbs "listen" and "see". This information can be

expressed using the following formulas.

$$rel(\textit{pay attention to sound}) \Rightarrow rel(\textit{to become aware of thought and senses}), (0.3)$$
$$rel(\textit{percieve by sight}) \Rightarrow rel(\textit{to become aware of thought and senses}), (0.3)$$

The number 0.3 represents the probability that someone who is interested in a sense will also be interested in the hypernym of the 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. The general formula is shown next.

$$rel(\textit{sense}) \Rightarrow rel(\textit{hypernym of the sense}), (0.3) \tag{16}$$

3.10 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 represent this relationship using the following formulas.

$$rel(\textit{not moving quickly}) \Rightarrow rel(\textit{acting or moving quickly}), (0.6)$$
$$rel(\textit{acting or moving quickly}) \Rightarrow rel(\textit{not moving quickly}), (0.6)$$

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 probability is high because the "related to" relationship represents relatively strong semantic similarity. The general formula is shown below.

$$rel(\textit{sense}) \Rightarrow rel(\textit{related to sense}), (0.6) \tag{17}$$

WordNet also defines the *similar to* relationship between adjectives. We create formulas with evidence probability of 0.8 for this relationship 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" and the sense for the word "prevailing": "most frequent or common" are similar to each other. We will therefore create the following formulas.

$$rel(\textit{coming at short intervals}) \Rightarrow rel(\textit{most frequent} \ldots), (0.8)$$
$$rel(\textit{most frequent or common}) \Rightarrow rel(\textit{coming at} \ldots), (0.8) \tag{18}$$

Note that both the "similar to" and "related to" relationships are symmetric and therefore the evidence probability for each formula and its reverse is the same. The general formula is shown next.

$$rel(\textit{sense}) \Rightarrow rel(\textit{similar to sense}), (0.6) \tag{19}$$

3.11 Building the Probabilistic Graph

Equations 3-19 from the previous subsections show how to create Horn clauses from WordNet. Once the formulas are extracted, they are converted into a probabilistic graph. In order to do so, first, we create a node for each random variable, that is, for each word form and each sense. Next, we convert the evidence probabilities of the formulas to weights using Equation 1 and 2. Note that there can be several identical formulas with possibly different weights that are generated. When this is the case, we will merge all such formulas into a single formula. The weight of the new formula is equal to the sum of the weights of the old formulas. For example, consider the following two formulas.

$$\begin{aligned} rel(X) &\Rightarrow rel(Y), 2.3 \\ rel(X) &\Rightarrow rel(Y), 1.1 \end{aligned} \tag{20}$$

The old formulas will be removed and the following new formula will be created.

$$rel(X) \Rightarrow rel(Y), 3.4 \tag{21}$$

First, note that we are adding the weights of the formulas and not the probabilities and therefore the evidence probability of the formula will always stay below 1.0. Second, note that since the evidence probabilities are always above 0.5, our model is monotonic (i.e, adding a new formula will always increase the evidence probability of the final formula). Lastly, note that adding the weights is consistent with the MLN model. Specifically, the probability of a world X is computed using the following formula.

$$P(X) = \frac{1}{total} e^{\left(\sum_F w(F) * |F(X)|\right)} \tag{22}$$

In the formula, *total* is a normalizing constant that is used to make sure that all the probabilities over all worlds add up to one. The sum is over all formulas F in our knowledgebase. The expression $w(F)$ is used to denote the weight of the formula F and $|F(X)|$ is equal to one when the formula F is true in the world X and is equal to 0 otherwise. Obviously, merging identical formulas by adding up their weights follows the above formula.

Next, we add an edge between X and Y in the graph for each logical formula of the following type.

$$rel(X) \Rightarrow rel(Y), w$$

The weight of the edge will be converted to a probability and will be computed using the following formulas.

$$p = \frac{1}{1+e^{-w}}$$
$$edge\,weight = \frac{2*p-1}{total}$$

The first formula converts the weight to a probability. The second formula maps the probability from the interval [0.5,1] back to the interval [0,1] and divides the result by the sum of the weights of all edges that leave the source node X. This guarantees that the sum of the weights of all the edges that leave a node will be equal to one.

In the probabilistic graph that was constructed, the weight of each edge is equal to the probability that a user is interested in the destination concept given that they are interested in the source concept, where we assume that the user is interested in only one of the destination concepts.

4 Measuring the Semantic Distance Between Word Forms

We will next show how to compute the semantic similarity between two arbitrary word form nodes in the graph. Our algorithm will return a number that is between zero and one. One will be returned when the two word forms are the same. Also, note that it is perfectly reasonable for two word forms to represent completely unrelated concepts and the semantic similarity between the word forms to be equal to zero. The semantic distance will be computed as a function of the average of the probability that the first word form is relevant given the second word form is relevant and the probability that the second word form is relevant given the first word form is relevant.

Consider two nodes n_1 and n_k in the probabilistic graph. We will show three different ways to compute the probability that n_k is relevant given that n_1 is relevant. In Section 5, we will compare the accuracy of the different approaches.

4.1 Multiplication Approach

A version of this approach was initially published in [43]. Consider a node sequence $n_1 \cdots n_k$ that forms a directed acyclic path in the graph. Let A_i be a random variable that represents the event that n_i is relevant for $i = 1$ to k. From probability theory, we have the following equation.

$$\begin{aligned} P(A_2 \cdots A_k|A_1) &= \frac{P(A_1 \cdots A_k)}{P(A_1)} = \\ &= \frac{P(A_1)P(A_2|A_1)P(A_3|A_1A_2)\cdots P(A_k|A_1 \cdots A_{k-1})}{P(A_1)} = \\ &= P(A_2|A_1)P(A_3|A_1A_2)\cdots P(A_k|A_1 \cdots A_{k-1}) \end{aligned} \quad (23)$$

Next, we will simplify the formula by assuming some level of independence. Suppose that the event A_i only depends on the preceding event A_{i-1}. This is the same assumptions that is made in Bayesian networks. Given this assumption, we can rewrite the equation as follows.

$$P(A_2 \cdots A_k|A_1) = P(A_2|A_1)P(A_3|A_2)\cdots P(A_k|A_{k-1}) \quad (24)$$

The idea of this approach is that if n_k is relevant because n_1 is relevant and there is an acyclic directed path $n_1 \cdots n_k$ in the graph, then the nodes $n_2, \ldots, n_k$ must also be relevant. Next, we can use the above formula and compute the probability that n_k is relevant given that n_1 is relevant by simply multiplying the weights of the edges along the path. If there are multiple paths between n_1 and n_k in the graph, then we can add the conditional probability from each path. The result will be a probability because the weights of the edges are normalized. (Note that this is not the case in [43].) The formulas for computing the conditional probability are shown next.

$$P(A_k|A_1) = \sum_{Pt \text{ is acyclic path from node } n_1 \text{ to node } n_k} P(Pt) \quad (25)$$

$$P(Pt) = \prod_{(n_i,n_j) \text{ is an edge in the path } Pt} edgeWeight(n_i,n_j) \quad (26)$$

The *edgeWeight* function simply returns the weight of the edge. Note that the algorithm is not deterministic because there are different ways to select disjoint paths between two nodes in the graph. In our experiments we use the depth-first algorithm that is shown in Figure 1. Before calling the method, $totalDistance$ is set to zero. After the method is called, the variable contains the result. When the method is initially called, $distance$ is equal to one and $depth$ is equal to zero. As the method is recursively called, the distance decreases and the depth is incremented by one after every call. In order to find the probability that n_k is relevant given that n_1 is relevant, we will call the method as follows: $depthFirst(n_1, n_k, 1, 0)$. The method starts at n_1 and recursively calls itself on all adjacent nodes in the graph. The recursion terminates when we have reached n_k, we have reached a node that has already been visited, we are on a path of more than 20 edges, or the value for the conditional probability for the path has dropped below the threshold of 0.0001.

4.2 Markov Logic Network Approach

A version of this approach was initially published in [46]. This approach is similar to the previous algorithm in the sense that the conditional probabilities over the different paths are aggregated. However, this approach uses the MLN approach to compute the conditional probability along a single path.

Let n_1 and n_k be two nodes in the probabilistic graph. We will next describe an efficient way of computing the probability that n_k is relevant given that n_1 is relevant using only the evidence along the path $n_1 \cdots n_k$. From probability theory, we have the following formula.

$$P(rel(n_k)|rel(n_1)) = \frac{P(rel(n_1) \wedge rel(n_k))}{P(rel(n_1))} \quad (27)$$

Algorithm 1 $depthFirst(currentNode, endNode, distance, depth)$

if $currentNode = endNode$ **then**
  $totalDistance \leftarrow totalDistance + distance$
  return
end if
if $depth > 20$ or $distance < 0.0001$ or $currentNode$ is visited **then**
  return
end if
for all neighbors $neighbor$ of $currentNode$ **do**
  $depthFirst(neighbor, endNode, distance * edgeWeigth(currentNode, neighbor)\ ,\ depth+1)$
end for

Figure 1: Recursive method for finding disjoint paths between two nodes and computing the conditional probability

We will next show how to compute the numerator and denominator of the above expression using the weights of the edges along the path $n_1 \cdots n_k$.

Let $f00(i)$ be the non-normalized probability from Equation 22 (i.e., we do not divide by *total*) that n_i and n_k are both irrelevant. Similarly, let $f01(i)$ be the non-normalized probability that n_i is irrelevant and n_k is relevant, $f10(i)$ be the non-normalized probability that n_i is relevant and n_k is irrelevant, and $f11(i)$ be the non-normalized probability that both n_i and n_k are relevant. In order to understand why we need these functions, note that Equation 27 can be rewritten as follows.

$$\frac{P(rel(n_1) \wedge rel(n_k))}{P(rel(n_1))} = \frac{f11(1)}{f10(1)+f11(1)} \tag{28}$$

The numerator expresses the non-normalized probability that both n_1 and n_k are relevant. The non-normalized probability of n_1 being relevant is computed as $f10(1)+f11(1)$. The reason is that this formula computes the probability that n_1 is relevant and n_k is irrelevant plus the probability that n_1 is relevant and n_k is relevant, which is equal to exactly the probability that n_1 is relevant. Lastly, note that the fact that the probabilities are not-normalized will not affect the result because we divide a non-normalized probability by a non-normalized probability. That is, if the probabilities are normalized, then we will divide both the numerator and the denominator of the expression by the same constant *total* from Equation 22 and the result will not change.

We will compute $f00$, $f01$, $f10$, and $f11$ using dynamic programming. Using MLN theory, we have the following base case.

$$\begin{aligned} f00(k-1) &= \frac{1+edgeWeight(n_{k-1},n_k)}{1-edgeWeight(n_{k-1},n_k)} \\ f01(k-1) &= \frac{1+edgeWeight(n_{k-1},n_k)}{1-edgeWeight(n_{k-1},n_k)} \\ f10(k-1) &= 1 \\ f11(k-1) &= \frac{1+edgeWeight(n_{k-1},n_k)}{1-edgeWeight(n_{k-1},n_k)} \end{aligned} \tag{29}$$

The four values follow from Equation 22 and Equations 1 and 2. Note that we have the following formula and evidence probability.

$$rel(n_{k-1}) \Rightarrow rel(n_k), (edgeWeight(n_{k-1},n_k))$$

The weight of the formula can be computed using Equations 1 and 2 as $ln(\frac{0.5+\frac{edgeWeight(n_{k-1},n_k)}{2}}{1-(0.5+\frac{edgeWeight(n_{k-1},n_k)}{2})})$, which is equal to $ln(\frac{1+edgeWeight(n_{k-1},n_k)}{1-edgeWeight(n_{k-1},n_k)})$. Now, if n_{k-1} is not relevant and n_k is not relevant, then the formula $rel(n_{k-1}) \Rightarrow rel(n_k)$ will be true and according to Equation 22 the non-normalized probability for this world will be equal to $e^{ln(\frac{1+edgeWeight(n_{k-1},n_k)}{1-edgeWeight(n_{k-1},n_k)})} = \frac{1+edgeWeight(n_{k-1},n_k)}{1-edgeWeight(n_{k-1},n_k)}$. However, if n_{k-1} is relevant and n_k is irrelevant, then the formula will be false and the non-normalized probability will be equal to $e^0 = 1$.

Next, we present the recursive formulas for computing the four functions.

$$\begin{aligned} f00(i) &= f00(i+1) * \frac{1+edgeWeight(n_i,n_{i+1})}{1-edgeWeight(n_i,n_{i+1})} + \\ &\quad f10(i+1) * \frac{1+edgeWeight(n_i,n_{i+1})}{1-edgeWeight(n_i,n_{i+1})} \\ f10(i) &= f00(i+1) * 1 + f10(i+1) * \frac{1+edgeWeight(n_i,n_{i+1})}{1-edgeWeight(n_i,n_{i+1})} \\ f01(i) &= f01(i+1) * \frac{1+edgeWeight(n_i,n_{i+1})}{1-edgeWeight(n_i,n_{i+1})} + \\ &\quad f11(i+1) * \frac{1+edgeWeight(n_i,n_{i+1})}{1-edgeWeight(n_i,n_{i+1})} \\ f11(i) &= f01(i+1) * 1 + f11(i+1) * \frac{1+edgeWeight(n_i,n_{i+1})}{1-edgeWeight(n_i,n_{i+1})} \end{aligned} \tag{30}$$

Let us examine the first formula in detail. In this case, we want to compute the non-normalized probability of the world where both n_i and n_k are irrelevant. We have two sub-cases: when n_{i+1} is relevant and when n_{i+1} is irrelevant. When n_{i+1} is relevant, the following formula will be true.

$$rel(n_i) \Rightarrow rel(n_{n+1}), (edgeWeight(n_i,n_{i+1})) \tag{31}$$

We will therefore add to the probability $f00(i+1) * e^{ln(\frac{1+edgeWeight(n_i,n_{i+1})}{1-edgeWeight(n_i,n_{i+1})})}$ in this case, which is equal to $f00(i+1) * \frac{1+edgeWeight(n_i,n_{i+1})}{1-edgeWeight(n_i,n_{i+1})}$. We use the expression $f00(i+1)$ because

we know that both n_{i+1} and n_k are irrelevant in this sub-case. The second sub-case is when n_{i+1} is irrelevant. The above formula will be true again and therefore we add to the probability the expression $f10(i+1) * \frac{1+edgeWeight(n_i,n_{i+1})}{1-edgeWeight(n_i,n_{i+1})}$.

Next, let us examine the second formula from Equation 30. In this case, we want to compute the non-normalized probability of the world where n_i is relevant, but n_k is irrelevant. We have two sub-cases: when n_{i+1} is relevant and when n_{i+1} is irrelevant. When n_{i+1} is irrelevant, Equation 31 does not hold and therefore will add the probability $f00(i+1) * e^0$. The second sub-case is when n_{i+1} is relevant and we will add the probability $f10(i+1) * \frac{1+edgeWeight(n_i,n_{i+1})}{1-edgeWeight(n_i,n_{i+1})}$ because Equation 31 holds. The last two formulas from Equation 30 can be derived similarly.

Note that our program for computing the f functions uses dynamic programming instead of recursion and runs in linear time relative to the size of the path. It first computes the value for the functions with input $k-1$ and then it applies the formulas from Equation 30 with values for i from $k-2$ up to 1. At the end, Equation 28 can be applied to find the conditional probability along the path $n_1 \cdots n_k$. If there are multiple paths along n_1 and n_k, then the conditional probabilities from the disjoint paths are aggregated using the algorithm from Figure 1.

4.3 Markov Logic Network Combined with Random Walk

Our experimental section (Section 5) shows that this approach produces the most accurate results. The drawback of the two previous approaches is that only disjoint paths between the nodes that are compared are explored. However, in most cases there are multiple interweaving paths between the two nodes and looking at only disjoint paths is not a very accurate approximation of the conditional probability. Here, we propose a simple alternative using a random walk. The algorithm from Figure 2 starts at *currentNode* and randomly visits 20 nodes in the search of *endNode*. If *endNode* is found, then the algorithm returns 1. Otherwise, it returns 0. We call this algorithm 10,000 times for the two nodes that we are comparing and aggregate the result. If we divide the total by 10,000, then we will get the conditional probability that the second node is relevant given that the first node is relevant, where the accuracy will be 4 digits after the decimal dot. We chose to look at paths of at most 20 nodes because we believe that longer paths give very little evidence about the semantic relationship between the word forms that the nodes represent.

Note that it is possible for the *randomWalk* algorithm to reach a dead end. For example, if we reach a node and there are no adjacent nodes that are not visited, the algorithm will return 0. This means that the random walk was unable to find the *endNode*. Specifically, the algorithm tries 100 times to find an adjacent node that is not visited and it gives up if it is unable to find such a node.

Algorithm 2 *randomWalk*(*currentNode*, *endNode*)

```
for i ← 0 to 20 do
  if currentNode = endNode then
    return 1
  end if
  repeat
    nextNode ← getRandomNextNode(currentNode)
  until nextNode is not already visited or loop has run for
  100 times
  if above loop ran 100 times then
    return 0
  end if
  curentNode ← nextNode
end for
return 0
```

Figure 2: The method takes a random walk from *currentNode* and it returns 1 if it reaches *endNode* and 0 otherwise

4.4 Linear and Logarithmic Distance Metrics

Let $P(Y|X)$ denote the result of computing $P(rel(Y)|rel(X))$ using one of the three algorithms that we presented in the last three subsections. Next, we present two functions for measuring semantic similarity between two word forms. The linear function is shown in Equation 32.

$$|wf_1, wf_2|_{lin} = min(\alpha, \frac{P(wf_1|wf_2) + P(wf_2|wf_1)}{2}) * \frac{1}{\alpha} \quad (32)$$

The minimum function is used in order to cap the value of the similarity function at one. The coefficient α amplifies the available evidence ($\alpha \leq 1$). The experimental section of the article shows how the value for α is picked. Note that when α is equal to one, then the function simply takes the average of the two numbers and caps the result at one.

The second semantic similarity function is inverse logarithmic, that is, it amplifies the smaller values. It is shown in Equation 33. 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. Again, the experimental section of the article shows how the value for α is picked.

$$|wf_1, wf_2|_{log} = norm(\frac{-1}{log_2(min(\alpha, \frac{P(wf_1|wf_2)+P(wf_2|wf_1)}{2}))}) \quad (33)$$

5 Experimental Validation

The system consists of two programs: one that creates the probabilistic graph and one that queries the graph. We used the Java API for WordNet Searching (JAWS) to connect to WordNet. The interface was developed by Brett Spell [40]. All experiments were performed on a laptop with Intel i7 CPU and 16GB of main memory. It takes about three minutes to build the probabilistic graph and save it to the hard disk. The size of the graph file is 81MB and it easily fits in main memory. It takes about 5 seconds to load the graph in main memory. We will refer to the three algorithms for finding the conditional probability between two nodes as the Multiplication, MLN, and MLN+Random Walk. The average time for computing the similarity distance between two word forms is about 100 milliseconds for the first two algorithms and about 1 second for the MLN+Random Walk algorithm. It takes about three minutes to build the initial probabilistic graph.

We evaluated our system on five different benchmarks. For each benchmark, experiments with human subjects were conducted and the average human judgment for each pair of words was recorded. The *RG65 data set* was created by Rubenstein and Goodenough and contains 65 pairs of words ([33]). The *MC28 dataset* contains 28 pairs of words and was created by Miller and Charles [21]. The *Agirre201 dataset* contains 201 pairs of words and was developed by Agirre et al. [1]. It is a subset of the *WordSim-353 dataset* that contains 353 pairs or words and was created by Finkelstein et al. [8]. Pierro and Euzenant recently ran a new study on the RG65 dataset and got slightly different results ([24]) – we will refer to this benchmark as the $P\&S_{full}$ *dataset*. Lastly, the *SimLex665 dataset* contains 665 pairs of words and was introduced by Hill et al. [12]. This happens to be the largest and most recent word similarity benchmark in literature.

For each dataset, we computed the Person and Spearman correlation between the data from the studies and the data that was produced by our system. The Person correlation is computed as shown in Equation 34. Note that we have used $\bar{X}$ to define the average of the numbers in the vector. We assume that the two vectors: $X = \langle x_1, \ldots, x_n \rangle$ and $Y = \langle y_1, \ldots, y_n \rangle$ are the input to the formula.

$$PearsonCorrelation(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{Y})^2}} \tag{34}$$

A notable property of the Pearson correlation is that it is invariant as regards to any Euclidean operation, such as scaling, translation, or rotation of the data.

The formula for the Spearman correlation is shown in Equation 35.

$$SpearmanCorrelation(X,Y) = 1 - \frac{6\sum_{i=1}^{n}(rank(x_i) - rank(y_i))^2}{n(n^2-1)} \tag{35}$$

The $rank(x_i)$ expression returns the position of the number x_i in the sorted version of the list X. A notable property of the Spearman correlation is that it is rank invariant, that is, a monotonic transformation would not affect its value.

In Tables 1, 2, and 3 we show the Pearson, Spearman, and the average of the two correlations, respectively, for our algorithms. We compare our results to the current state-of-the-art, which includes 16 algorithms. The correlation data for these 16 algorithms was taken as report by Lastra-Diaz and Garcia-Serrano in [18].

Note that both our linear and logarithmic similarity metrics take as input the parameter α (see Equations 32 and 33). Table 4 shows the values for α that were used to create our experimental results. These values were selected because they produce the highest Pearson correlation for the RG65 dataset. It turns out that they are close to optimal (i.e., produce the highest correlation) for both the Pearson and Spearman correlation on the other benchmarks as well.

Looking at Table 1, we see that our MLN+RandomWalk algorithm that uses the logarithmic similarity metric gives us the highest Pearson correlation on three of the five benchmarks. More over, this algorithm also gives us the highest value for the average of the Pearson correlation over the five benchmarks. These are significant results that demonstrate the high quality of the data inside the probabilistic graph. It is also worth noting that the MLN+RandomWalk algorithm produces higher average Pearson correlation than the MLN algorithm, which in turns produces higher average correlation than the Multiplication algorithm. The reason is that the MLN+RandomWalk algorithm is based on strict probabilistic theory and is able to take into account the interweaving paths in the graph between the two nodes that we are comparing. Note as well that the logarithmic similarity metric produces slightly better results than the linear case. Specifically, the average over the three algorithms is 0.7513 for the logarithmic similarity metric and 0.7440 for the linear one.

Next, consider Table 2. Again, the MLN+RandomWalk algorithm that uses the logarithmic similarity metric produces Spearman correlation that is higher than the current state-of-the-art algorithms on three of the five benchmarks. In addition, the algorithm produces the highest value for the average of the Spearman correlation over the five benchmarks. Again, the logarithmic similarity metric produces a little higher correlation: 0.6931 average Spearman correlation for the linear case and 0.6996 average Spearman correlation for the logarithmic case. Note that the Spearman correlation for the MLN algorithm is the same for the linear and logarithmic similarity distance metric. The reason is that $\alpha = 0.3$ for both algorithms. This number means that results that are equal to above 0.3 for both metrics are mapped to 1. In other words, the ranking

Table 1: Pearson correlation on the five different benchmarks (the highest values are in bold)

Algorithm/Data Set	*RG65*	*MC28*	*Agirre201*	*P&S$_{full}$*	*SimLex665*	*Average*
Resnik$_{ic-treebank-add1}$ [29]	0.8653	0.8809	0.6913	0.9003	0.5955	0.7867
Yuan et al. [52]	0.8675	0.8407	0.7061	0.9082	**0.6106**	0.7866
Seco et al. [36]	0.8642	0.8557	0.6969	0.9042	0.6048	0.7852
Sanchez et al. [34]	0.8752	0.8595	0.6946	0.9025	0.5941	0.7852
Meng et al. [19]	0.8723	0.8393	0.7039	0.9057	0.6010	0.7844
Harispe et al. [11]	0.8589	0.8575	0.6960	0.9003	0.6056	0.7836
Resnik$_{ic-semcorraw-add1}$ [29]	0.8658	0.8621	0.6955	0.8997	0.5930	0.7832
Sanchez et al.[35]	0.8616	0.8507	0.6973	0.9042	0.5995	0.7827
CondProbCosine [18]	0.8634	0.8562	0.6902	0.9015	0.5964	0.7815
CondProbHypo [18]	0.8658	0.8552	0.6874	0.9015	0.5940	0.7808
CondProbLeaves [18]	0.8635	0.8511	0.6891	0.9008	0.5934	0.7796
CPCorpus$_{ic-treebank-add1}$ [18]	0.8633	0.8678	0.6807	0.8987	0.5863	0.7794
CPCorpus$_{ic-semcorraw-add1}$ [18]	0.8647	0.8504	0.6792	0.8979	0.5843	0.7753
Zhou et al. [53]	0.8589	0.8403	0.6848	0.8905	0.5985	0.7746
CondProbLogistic$_{k8}$ [18]	0.8692	0.8142	0.6809	0.9064	0.5972	0.7736
Hadj Taieb et al. [50]	0.7933	0.6899	0.6490	0.8167	0.4921	0.6570
$Multiplication\,(linear)$	0.8690	0.8391	0.6256	0.8993	0.3995	0.7265
$Multiplication\,(log)$	0.8536	0.8220	0.5962	0.8996	0.4392	0.7221
$MLN\,(linear)$	0.8173	0.8653	0.7115	0.8475	0.4438	0.7371
$MLN\,(log)$	0.8160	0.8661	**0.7273**	0.8382	0.4575	0.7410
$MLN + RandomWalk\,(linear)$	0.8874	0.8913	0.7002	0.9152	0.4472	0.7683
$MLN + RandomWalk\,(log)$	**0.8992**	**0.9290**	0.7105	**0.9237**	0.4914	**0.7908**

Table 2: Spearman correlation on the five different benchmarks (the highest values are in bold)

Algorithm/Data Set	*RG65*	*MC28*	*Agirre201*	*P&S$_{full}$*	*SimLex665*	*Average*
Resnik$_{ic-treebank-add1}$ [29]	0.7831	0.8882	0.6461	0.7783	0.5810	0.7353
Yuan et al. [52]	0.8206	0.8274	0.6656	0.8199	**0.6027**	0.7473
Seco et al. [36]	0.8012	0.8727	0.6643	0.7919	0.5901	0.7441
Sanchez et al. [34]	0.8034	0.8492	0.6576	0.8003	0.5906	0.7402
Meng et al. [19]	0.8166	0.8296	0.6581	0.8127	0.5957	0.7426
Harispe et al. [11]	0.7977	0.8697	0.6539	0.7904	0.5918	0.7407
Resnik$_{ic-semcorraw-add1}$ [29]	0.7922	0.8712	0.6505	0.7835	0.5782	0.7351
Sanchez et al.[35]	0.7911	0.8551	0.6590	0.7854	0.5850	0.7351
CondProbCosine [18]	0.7896	0.8606	0.6524	0.7834	0.5828	0.7337
CondProbHypo [18]	0.8017	0.8554	0.6466	0.7910	0.5806	0.7350
CondProbLeaves [18]	0.7877	0.8389	0.6478	0.7808	0.5799	0.7270
CPCorpus$_{ic-treebank-add1}$ [18]	0.7722	0.8502	0.6364	0.7691	0.5735	0.7203
CPCorpus$_{ic-semcorraw-add1}$ [18]	0.7916	0.8247	0.6389	0.7813	0.5712	0.7216
Zhou et al. [53]	0.8051	0.8244	0.6591	0.7999	0.5945	0.7366
CondProbLogistic$_{k8}$ [18]	0.7993	0.8034	0.6460	0.7921	0.5791	0.7240
Hadj Taieb et al. [50]	0.7417	0.6961	0.6175	0.7463	0.4833	0.6570
$Multiplication\,(linear)$	0.7365	0.7653	0.4893	0.7348	0.3964	0.6245
$Multiplication\,(log)$	0.7424	0.7859	0.4969	0.7456	0.4140	0.6370
$MLN\,(linear)$	0.7704	0.8420	**0.6953**	0.7687	0.4552	0.7063
$MLN\,(log)$	0.7704	0.8420	**0.6953**	0.7687	0.4552	0.7063
$MLN + RandomWalk\,(linear)$	0.8375	0.9236	0.6789	0.8235	0.4794	0.7486
$MLN + RandomWalk\,(log)$	**0.8392**	**0.9423**	0.6801	**0.8253**	0.4909	**0.7556**

Table 3: Average of Pearson and Spearman correlation on the five benchmarks (the highest values are in bold)

Algorithm/Data Set	*RG65*	*MC28*	*Agirre201*	*P&S$_{full}$*	*SimLex665*	*Average*
Resnik$_{ic-treebank-add1}$ [29]	0.8242	0.8846	0.6687	0.8393	0.5883	0.7610
Yuan et al. [52]	0.8441	0.8341	0.6859	0.8641	**0.6067**	0.7670
Seco et al. [36]	0.8327	0.8642	0.6806	0.8481	0.5975	0.7647
Sanchez et al. [34]	0.8393	0.8544	0.6761	0.8514	0.5924	0.7627
Meng et al. [19]	0.8445	0.8345	0.6810	0.8592	0.5984	0.7635
Harispe et al. [11]	0.8283	0.8636	0.6750	0.8454	0.5987	0.7622
Resnik$_{ic-semcorraw-add1}$ [29]	0.8290	0.8667	0.6730	0.8416	0.5856	0.7592
Sanchez et al.[35]	0.8264	0.8529	0.6782	0.8448	0.5923	0.7589
CondProbCosine [18]	0.8265	0.8584	0.6713	0.8425	0.5896	0.7576
CondProbHypo [18]	0.8338	0.8553	0.6670	0.8463	0.5873	0.7579
CondProbLeaves [18]	0.8256	0.8450	0.6685	0.8408	0.5867	0.7533
CPCorpus$_{ic-treebank-add1}$ [18]	0.8178	0.8590	0.6586	0.8339	0.5799	0.7499
CPCorpus$_{ic-semcorraw-add1}$ [18]	0.8282	0.8376	0.6591	0.8396	0.5778	0.7485
Zhou et al. [53]	0.8320	0.8324	0.6720	0.8452	0.5965	0.7556
CondProbLogistic$_{k8}$ [18]	0.8343	0.8088	0.6635	0.8493	0.5882	0.7488
Hadj Taieb et al. [50]	0.7675	0.6930	0.6333	0.7815	0.4877	0.6570
$Multiplication\,(linear)$	0.8028	0.8022	0.5575	0.8171	0.3980	0.6755
$Multiplication\,(log)$	0.7980	0.8040	0.5466	0.8226	0.4266	0.6795
$MLN\,(linear)$	0.7939	0.8537	0.7034	0.8081	0.4495	0.7217
$MLN\,(log)$	0.7932	0.8541	**0.7113**	0.8035	0.4564	0.7237
$MLN + RandomWalk\,(linear)$	0.8625	0.9075	0.6896	0.8694	0.4633	0.7584
$MLN + RandomWalk\,(log)$	**0.8692**	**0.9357**	0.6953	**0.8745**	0.4912	**0.7732**

Table 4: Values for α

Algorithm	α *linear metric*	α *log metric*
Multiplication	0.002	0.1
MLN	0.3	0.3
MLN+RandomWalk	0.006	0.015

is the same after applying the linear or logarithmic similarity distance metric and therefore the Spearman correlation is the same. Lastly, note that again the MLN+RandomWalk algorithm produces the highest average correlation, followed by the MLN and the Multiplication algorithm. However, the MLN algorithm produces the best results on the Agirre201 benchmark.

Lastly, consider Table 3 that shows the average of the Pearson and Spearman correlation. Again, the MLN+RandomWalk algorithm that uses the logarithmic similarity measure produces higher correlation than previous algorithms on four of the five benchmarks and the highest average correlation over the five benchmarks. The logarithmic similarity metric produces a little higher correlation than the linear one: 0.7185 average correlation for the linear case and 0.7255 average correlation for the logarithmic case. The MLN+RandomWalk algorithm produces higher average correlation than the MLN algorithm, which produces higher average correlation than the Multiplication algorithm.

The Java source code and all text files that are needed to reproduce the experimental results can be found at [41].

6 Conclusion and Future Research

In this article, we presented a new Markov Logic Network algorithm that uses a random walk to compute the semantic similarity between two word forms of the English language. We showed that the logarithmic version of the algorithm produces higher average correlation over five benchmarks than the current state-of-the-art algorithms. We believe that these results are due to the fact that our algorithm processes not only structured data, but also natural language information from WordNet. Moreover, unlike our previous work, the algorithm considers all the evidence from the probabilistic graph and not only the disjoint paths between the nodes that are compared.

Although the random walk algorithm gives very accurate results, it is not necessarily the most efficient way of computing the semantic similarity between two nodes in the probabilistic graph. In the future, we plan to explore alternative methods for computing the semantic similarity distance between two nodes, such as Gibbs sampling, belief propagation, and approximation via pseudolikelihood. We also plan on conducting experiments on the full-blown version of the probabilistic graph that includes data from Wikipedia ([42]) and determining if this can improve the correlation values with the five benchmarks.

References

[1] Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., and Soroa, A. “A Study on Similarity and Relatedness using Distributional and WordNet-based Approaches”.

Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 19–27, 2009.

[2] Blanco, R., Mika, P., and Vigna, S. "Effective and Efficient Entity Search in RDF Data". Tenth International Conference on The Semantic Web, pp. 83–97, 2011.

[3] Bollegala, D., Matsuo, Y., and Ishizuka, M. "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.

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

[5] Castells, P., Fernandez, M., and Vallet, D. "An Adaptation of the Vector-space Model for Ontology-based Information retrieval". IEEE Transactions on Knowledge and Data Engineering, 19(2):pp. 261–272, 2007.

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

[7] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. "Indexing by Latent Semantic Analysis". Journal of the Society for Information Science, 41(6):pp. 391–407, 1990.

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

[9] Fox, C. "Lexical Analysis and Stoplists". Information Retrieval: Data Structures and Algorithms, pp. 102–130, 1992.

[10] Frakes, W. "Stemming Algorithms". Information Retrieval: Data Structures and Algorithms, pp. 131–160, 1992.

[11] Harispe, S., Ranwez, S., Janaqi, S., and Montmain, J. "Semantic Similarity from Natural Language and Ontology Analysis". Synthesis Lectures on Human Language Technologies, pp. 81–254, 2015.

[12] Hill, F., Reichart, R., and Korhonen, A. "SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation". arXiv:1408.3456, 2014.

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

[14] Jones, K. "A Statistical Interpretation of Term Specificity and its Application in Retrieval". Journal of Documentation, 28(1):pp. 11–21, 1972.

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

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

[17] Landauer, T. K., Foltz, P., and Laham, D. "Introduction to Latent Semantic Analysis". Discourse Processes, pp. 259–284, 1998.

[18] Lastra-Diaz, J. J. and Garcia-Serrano, A. "A New Family of Information Content Models with an Experimental Survey on WordNet". Elsevier Knowledge-Based Systems, 89(1):pp. 509–526, 2015.

[19] Meng, L., Gu, J., and Zhou, Z. "A New Model of Information Content Based on Concept's Topology for Measuring Semantic Similarity in WordNet". International Journal on Grid Distributed Computing, 5(3):pp. 81–93, 2012.

[20] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. "Distributed Representations of Words and Phrases and their Compositionality". Advances in Neural Information Processing Systems, 26(1):pp. 3111–3119, 2013.

[21] Miller, G. and Charles, W. "Contextual Correlates of Semantic Similarity". Language and Cognitive Processing, 6(1):pp. 1–28, 1991.

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

[23] Pan, R., Ding, Z., Yu, Y., and Peng, Y. "A Bayesian Network Approach to Ontology Mapping". Proceedings of the Fourth International Semantic Web Conference, pp. 563–577, 2005.

[24] Pirro, G. and Euzenat, J. "A Feature and Information Theoretic Framework for Semantic Similarity and Relatedness". Proceedings of the ninth international semantic web conference on the semantic web, pp. 615–630, 2010.

[25] Porter, M. F. "An Algorithm for Suffix Stripping". Readings in Information Retrieval, pp. 313–316, 1997.

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

[27] RDF Wordking Group. “Resource Description Framework (RDF)”. http://www.w3.org/RDF/, 2014.

[28] Resnik, P. “Using Information Content to Evaluate Semantic Similarity in a Taxonomy”. International Joint Conference on Artificial Intelligence, pp. 448–453, 1995.

[29] Resnik, P. “Semantic Similarity in a Taxonomy: An Information-based measure and its application to problems of ambiguity in natural language”. Journal of Artificial Intelligence, 11:pp. 95–130, 1999.

[30] Richardson, M. and Domingos, P. “Markov Logic Networks”. Machine Learning, 62(1-2):pp. 107–136, 2006.

[31] Robertson, S. and Zaragoza, H. “The Probabilistic Relevance Framework: BM25 and Beyond, Foundations and Trends in Information Retrieval”. Foundations and Trends in Information Retrieval, 3(4):pp. 333–389, 2009.

[32] Rocha, C., Schwabe, D., and Aragao, M. “A Hybrid Approach for Searching in the Semantic Web”. Thirteenth International World Wide Web Conference (WWW 2004), pp. 374–383, 2004.

[33] Rubenstein, H. and Goodenough, J. B. “Contextual Correlates of Synonymy”. Communications of the ACM, 8(10):pp. 627–633, 1965.

[34] Sanchez, D., Baret, M., and Isern, D. “Ontology-based Information Content Computation”. Knowledge-Based Systems, 24(2):pp. 297–303, 2011.

[35] Sanchez, D. and Batet, M. “A New Model to Compute the Information Content of Concepts from Taxonomic Knowledge”. International Journal on Semantic Web Information Systems, 8(2):pp. 34–50, 2012.

[36] Seco, N., Veale, T., and Hayes, J. “An Intrinsic Information Content Metric for Semantic Similarity in WordNet”. Sixteenth European Conference on Artificial Intelligence, 16:pp. 1089–1094, 2014.

[37] Simone Paolo Ponzetto and Michael Strube. “Deriving a Large Scale Taxonomy from Wikipedia”. 22nd International Conference on Artificial Intelligence, 2007.

[38] Simske, S., Boyko, I., and Koutrika, G. “Multi-Engine Search and Language Translation”. ExploreDB, 2014.

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

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

[41] Stanchev, L. “Experimental Results”. http://users.csc.calpoly.edu/∼lstanche/kbs.

[42] Stanchev, L. “Creating a Phrase Similarity Graph from Wikipedia”. Eight IEEE International Conference on Semantic Computing, 2014.

[43] Stanchev, L. “Creating a Similarity Graph from WordNet”. Fourth International Conference on Web Intelligence, Mining and Semantics, 2014.

[44] Stanchev, L. “Fine-Tuning an Algorithm for Semantic Search Using a Similarity Graph”. International Journal on Semantic Computing, 9(3):pp. 283–306, 2015.

[45] Stanchev, L. “Semantic Search using a Similarity Graph”. Ninth IEEE International Conference on Semantic Computing, 2015.

[46] Stanchev, L. “Creating a Probabilistic Graph for WordNet using Markov Logic Network”. Sixth International Conference on Web Intelligence, Mining and Semantics, 2016.

[47] Stanchev, L. “Semantic Document Clustering Using a Similarity Graph”. Tenth IEEE International Conference on Semantic Computing, 2016.

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

[49] Stojanovic, N. “On Analyzing Query Ambiguity for Query Refinement: The Librarian Agent Approach”. Twenty Second International Conference on Conceptual Modeling, pp. 490–505, 2003.

[50] Taieb, M. A. H., Aouicha, M. B., and Hamadou, A. B. “Ontology-based Approach for Measuring Semantic Similarity”. Elsevier Engineering Applications of Artificial Intelligence, 36:pp. 238–261, 2014.

[51] The World Wide Web Consortium. “OWL Web Ontology Language Guide”. http://www.w3.org/TR/owl-guide/, 2014.

[52] Yuan, Q., Yu, Z., and Want, K. “A New Model of Information Content for Measuring the Semantic Similarity Between Concepts”. International Conference on Cloud Computing and Big Data, pp. 141–146, 2013.

[53] Z.Zhou, Want, Y., and Gu, J. “A New Model of Information Content for Semantic Similarity in WordNet”. Second International Conference on Future Generation Communication and Networking Symposia, 3:pp. 85–89, 2008.

Lubomir Stanchev is an Associate Professor at California Polytechnic State University, San Luis Obispo, California. Before joining Cal Poly, he was an Assistant and then an Associate Professor at Indiana-University Purdue-University Fort Wayne. He got his Ph.D. in Computer Science from the University of Waterloo in Canada. His research interests include databases, parallel algorithms, and semantic computing. He has published more than thirty papers in peer-reviewed conferences and journals.