Creating a Probabilistic Graph for WordNet using Markov Logic Network

Lubomir Stanchev

Computer Science Department

California Polytechnic State University

San Luis Obispo, California, USA

stanchev@gmail.com

Abstract

The paper shows how to create a probabilistic graph for WordNet. A node is created for every word and phrase in WordNet. An edge between two nodes is labeled with the probability that a user that is interested in the source concept will also be interested in the destination concept. For example, an edge with weight 0.3 between “canine” and “dog” indicates that there is a 30% probability that a user who searches for “canine” will be interested in results that contain the word “dog”. We refer to the graph as *probabilistic* because we enforce the constraint that the sum of the weights of all the edges that go out of a node add up to one. Structural (e.g., the word “canine” is a hypernym (i.e., kind of) of the word “dog”) and textual (e.g., the word “canine” appears in the textual definition of the word “dog”) data from WordNet is used to create a Markov logic network, that is, a set of first order formulas with probabilities. The Markov logic network is then used to compute the weights of the edges in the probabilistic graph. We experimentally validate the quality of the data in the probabilistic graph on two independent benchmarks: Miller and Charles and WordSimilarity-353.

CCS Concepts

Computing methodologies → Probabilistic reasoning; *Semantic* *networks;*

Keywords

WordNet; Markov Logic Network; Probabilistic Graph; Semantic Search

1. INTRODUCTION

WordNet contains very high quality information [21]. Many scientists have spent decades to develop this lexical corpus that contains about 150,000 word forms from the English language. A *word form* is a word or a phrase (e.g., "sports utility vehicle"). WordNet also contains information about the relationship between word forms (e.g., "vitamin C" has the same meaning as "ascorbic acid", a "dog" is "canine", or a "house" has "windows"). Lastly, WordNet contains textual descriptions, such as the definition of a sense of a word form or an example use of a word form in a sentence. All this information can be very useful to a human who wants to understand the meaning of different word forms and how they are related. However, it is difficult for a computer program to process this information because computers are not as adept as humans in processing natural language. We address this challenge by creating a probabilistic model for WordNet that stores conditional probabilities of pairs of word forms. We refer to this model as a *probabilistic graph*. For example, an edge between "cat" and "pet" with weight 0.2 in the graph means that there is a 20% change that a user who searches for cats will find documents that contain the word "pet" relevant.

A probabilistic graph has many applications. For example, in this paper and in [34] we show that it can be used to compute the degree of semantic similarity between a pair of word forms. In [36] 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 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 order based on the probability of being relevant to the input query, where the probabilistic graph can help us compute these probabilities. Lastly, [37] shows how a probabilistic graph can be used to perform semantic document clustering. For example, an online store can use a probabilistic graph to cluster the products that are offered in different categories. Two products should appear in the same category if they have textual descriptions that contain words that are similar based on the semantic similarity metric that can be extracted from the probabilistic graph.

Converting WordNet in a computer-friendly format is a daunting task because it contains heterogeneous data. While there are plethora of algorithms that process structured information [14, 29] and textual information [2, 15], experimental results have shown that processing both types of information yields the best results [35]. 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., [7, 8, 19]), 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.

To the best our knowledge, our previous research in the area [34, 36] is the only study on how structured and unstructured information from WordNet can be combined to model the semantic relationship between word forms. However, all our previous research in the area [34, 33, 36, 37] considers a similarity graph that is not probabilistic, that is, the sum of the weights of the edges that go out of a node can add up to more than one. In this paper, we examine how a Markov Logic Network (MLN) [26] can be applied to the problem. The final result is a probabilistic graph where the weights of the edges that go out of a node add up to one. In Section 6, we show how applying a strict probabilistic model and using a MLN affects the quality of the data in the graph.

Our 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*. For example, the main sense of the word chair is "a seat for one person" (Note that a word form can have many senses and several word forms can represent the same sense.) We can model part of this information by creating the formula $rel(a\ seat\ for\ one\ person) \Rightarrow rel(seat)$ and assigning it a weight 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 a node add up to one.

We experimentally validate the quality of the data in the probabilistic graph on two independent benchmarks: Miller and Charles [20] and WordSimilarity-353 [6]. The two benchmarks contain the degree of semantic relevance between pairs of words as determined by humans. We compare this data with the results of applying a similarity metric that uses the probabilistic graph. Roughly, we compute the semantic similarity between two nodes as the average of the conditional probability of the first node being relevant given the second node and the conditional probability of the second node being relevant given the first node. The conditional probability of one node given another is computed as a function of the sum of the conditional probabilities along all the cycleless paths between them, where we use the MLN model together with dynamic programming to compute the conditional probability along a path. The experimental results show that our approach produces good results on both benchmarks. We believe that the reason is that we process a lot of information, including natural language descriptions, and we are able to apply this information to build a mathematical model of the semantic relationships between word forms that is based on probability theory and MLNs. This approach is preferred to previous research (e.g. [34]) because it uses existing probabilistic theory algorithms.

In what follows, in Section 2 we review related research. The major contributions of the paper are in the next four sections. Section 3 shows our algorithm for creating logical formulas with weights from WordNet. Section 4 shows how this information can be transformed into a probabilistic graph. Section 5 shows a novel algorithm for measuring the semantic similarity between word forms based on the MLN model. Section 6 shows how our system compares with existing systems that measure word similarity, while concluding remarks and areas for future research are outlined in Section 7.

2. RELATED RESEARCH

Existing research that applies Bayesian networks to represent knowledge deals with the uncertain or probabilistic information in the knowledgebase [24, 22]. Our approach slightly differs because we do not store the probability that a word form is relevant given that an adjacent in the graph word form 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 [14, 29], while Glen Jeh and Jennifer Widom show how to approximate the similarity between words based on information about the structure of the graph in which they appear [11]. 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 [38] because the latter does not assign weights to the edges of the graph.

In this paper 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 [4] 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. Other approaches that rely on data from the Internet include papers by Danushka Bollegala, Yutaka Matsuo, and Mitsuru IshizukaRef [2] and by Swarnim Kulkami and Doina Caragea [15]. 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 [5]. 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 [16]. 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 highquality corpus, such as WordNet, is beneficial. Note as ell 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 [13] (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 is inconsequential. Conversely, if the word “ice” appears only in the definition of few senses, then the fact that the definition of one of the senses of the word “hockey” contains the word “ice” in its definition is statically meaningful.

Note that plenty of research effort has recently focused on using a description language, such as *ontology web language* (OWL) [41], to describe resources. A semantic query language, such as SPARQL [31] (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 [36] 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, our system for semantic search can return resources that are related to the input query in ranked order. In our approach 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, [28] examines how queries can be expanded based on the information from an OWL knowledgebase. Alternatively, [39] 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 [25] (stands for Resource Description Framework) with information retrieval approaches, such as BM25F [27]. For example, [1] 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.

Lastly, note that the probabilistic graph can be applied to the problem of query expansion in natural language search systems [30]. 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 LOGICAL FORMULAS FROM WORDNET

3.1. About WordNet

WordNet [21] 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 relationship between noun senses. The *meronym* relationship corresponds to the “partof” relationship. Note that WordNet provides three types of meronyms: *part*, *member*, and *substance*. The three types of meronyms can be explained with the following examples: a “tire” is part of a “car”, “car” is a member of “traffic jam”, and a “wheel” is made from “rubber”, respectively. The *holonym* relationship is the reverse of the meronym relationship. For example, “building” is a holonym of “window”. For verbs, WordNet defines the *hypernym* and *troponym* 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* relationship 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 Probability 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.

[Insert Equation 1]

Following the MLN model [26], the weight of a logical formula is equal to the natural logarithm of the odds of the formula being true, that is $ln(\frac{\mathrm{p}}{(1-\mathrm{p})})$. 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 given the word “chair” we are interested in the word “table”, we want this evidence to increase the conditional probability of the word “table” given the word “chair”. 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.

[Insert Equation 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*. In our 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 equal to $ln(0.55/0.45) = 0.2$.

Note that the same formula can appear multiple times in our knowledgebase. In the next section we will show how we can apply the MLN model to aggregate multiple evidence about the conditional probability between two concepts. In the remaining of this section, we describe how we model WordNet as a set of Horn clauses with weights. Note that, for the most part, we will only describe how to compute the evidence probabilities, where the weight of each formula can be computed using Equation 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 office who presides at meetings". Suppose that WordNet gives a frequency of 35, 2, and 1, respectively, to the three senses. We will then crate the following formula and probability for the first sense (Similar formulas and weights will also be created for the other two senses.)

$rel\,(chair) \Rightarrow rel\,(a\ seat\ for\ one\ person)$, (35/38)

Note that the word "chair" has three meanings. Based on the frequencies that we are given, the evidence probability for the relationship to the first meaning is 35/38. 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. In our example, we will also add the following formula and weight. Since there are no parentheses, the expression shows a weight and not an evidence probability.

$rel\,(a\ seat\ for\ one\ person) \Rightarrow rel\,(chair\,),\ 10$

In general, we always add a formula with weight 10 between a sense and all the word forms that it represents. The reason in that we have a very high degree of confidence that if a sense is relevant, then so are all the word form that denote the sense. A weight of 10 means that we have a very high degree of confidence (above 99.99%) of the formula being true. Note that in a MLN we cannot assign an evidence probability of one to a formula because this translates in a weight that is equal to infinity.

3.4. Processing the Definition of a Sense

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. 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 as *coef* $\ast$*computeMinMax*(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[2] of non-noise words in the sense. The variable denotes the importance of the word form in the definition of the sense. For example, if there are only two words in the definition of the sense, then they are both very important. However, if there are 20 words in the definition of the sense, then each individual word is less important. In our example, *ratio* = ½ for both formulas. However, *coef* = 1.0 for the first formula and *coef* = 0.8 for the second formula. The *computeMinMax* function returns a number that is in most cases between the first two arguments, where the magnitude of the number is determined by the third argument. Since the appearance of a word in the definition of a sense is not a reliable source of evidence about the relationship between the word and the sense, the value of the second argument is set to 0.6. The constant 0.6 is related to the probability that someone who is interested in a sense will be also interested in one of the words in the definition of the sense.

The *computeMinMax* function smoothens the value of the *ratio* parameter. For example, a word that appears as one of the 20 non-noise words in the definition of a sense is not ten times less important than a word that appears as one of the two non-noise word in the definition of a sense. The function makes the difference between the two cases less extreme. Using this function, the 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. Formally, the *computeMinMax* function is defined as follows.

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

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

In our example, *ratio* = ½ and therefore *computeMin-Max(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 Sense

WordNet also includes example use for each sense. In this subsection we show how to represent this information as logical 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 an example use does not have as strong as significance as the definition of a sense, we will calculate the evidence probability as $computeMinMax(0, 0.2, ratio)$. 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 word forms 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 $\frac{1}{5}$. Unlike the case with the definition of a sense, the first words in the example use are not more important. Therefore, we ignore the order of the words in the example use of a sense. The precise calculation for the evidence probability is $0.2 * (\frac{-1}{\log_2(0.2)})$ = 0.09.

3.6. Processing the Backward Relationships

We also create a formula for the conditional probability of a sense node given a word form for every word form that appears in the definition of the sense. The evidence probability of the formula is computed as *computeMinMax* (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 label of the sense. The constant 0.3 relates to the probability that someone who is interested in a word form will also be interested in one of the senses that have the word form in their definition. Here, we assume that the backward relationship is not as strong as the forward relationship. As an example, if the word "position" occurred as part of the definition of only three senses and exactly once in each definition, then we will add the following formula for the second sense of the word "chair". The evidence probability is computed as *computeMinMax* (0, 0.3, 1/3) = 0.19.

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

Similarly, we will create a formula that shows the conditional probability between a sense and a word form that appears in its example use. The weight of an edge in this case will be equal to *computeMinMax* (0, 0.1, *ratio*), where *ratio* is the number of the times the word form appears in the example use of the sense divided by the total number of occurrences of the word form in the example use of all senses. The constant 0.1 relates to the probability that someone who is interested in a word form will also be interested in one of the senses that have the word form in their example use. This value is smaller than the value for the definition of a sense because the words in the definition of a sense are closely related to the meaning of the sense. As an example, if the word "coat" occurred as part of the example use of only three senses and exactly once in each sense, then we will add the following formula for the first sense of the word "chair". The evidence probability is computed as *computeMinMax*(0, 0.1, 1/3) = 0.06.

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

3.7. Populating the Frequency of the Senses

So far, we have shown how to extract information from textual sources, such as the text for the definition and example use of a word sense. We will next show how structured knowledge, such as the hyponym (a.k.a. kind-of) relationship between senses, can be represented in the similarity graph. Most existing approaches [23] 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 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 crosssection of British English, both spoken and written, from the late twentieth century [3].

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

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 gives the word "president" a frequency of 9781, the contribution of the word "president" to the size of the sense *s* will be equal to $BNC\ (president) * p_s(president) = 9781 * 0.096 = 938.976$. 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", "wheelchair", and so on. 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*) ⇒*rel* (*chair with support ...*), (0.49)

rel(*a seat for one person*) ⇒*rel*(*a moveable chair ...*), (0.41)

The first formula shows the conditional probability for the sense "chair with a support on each side for arms" of the word "armchair", while the second formula shows the conditional probability for the sense "a movable chair on large wheels" for the word "wheelchair". 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. 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 *computeMinMax* function. The reason is that the function is only relevant for words in 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. Here is a formula from our example that will be added.

$rel(\textit{chair with support} \ldots) \Rightarrow rel(\textit{a seat for one person}),\ (0.3)$

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 formulas.

$rel\ (\textit{a seat for one person}) \Rightarrow rel\ (\textit{a support that} \ldots),\ (0.3)$

$rel(\textit{a seat for one person}) \Rightarrow rel(\textit{one of the supports} \ldots),\ (0.3)$

In general, we compute the evidence probability as $0.6/n$, where n is the number of meronyms of the sense. The constant 0.6 represents the probability that a user that is interested in a sense of a word form is also interested in one of its meronyms. In our system, this coefficient is set to 0.6 because the meronym relationship is not as strong as the hyponym relationship. The reasoning behind the formula is that the more meronyms a sense has, the less likely is that we are interested in a specific 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.

$rel\,(a\ support\ that\ ...) \Rightarrow rel\,(a\ seat\ for\ one\ person),\ (0.15)$

$rel(one\ of\ the\ supports\ ...) \Rightarrow rel(a\ seat\ for\ one\ person),\ (0.15)$

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, respectively, then we will create the following formulas.

$rel(an\ exchange\ of\ ideas\ via...) \Rightarrow rel(talk\ with\ a\ lisp),\ (0.14)$

$rel(an\ exchange\ of\ ideas\ via...) \Rightarrow rel(talk\ pompously),\ (0.01)$

$rel(an\ exchange\ of\ ideas\ via...) \Rightarrow rel(carry\ on\ ...)\,(0.75)$

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”, which have definitions “talk with a lisp”, “talk pompously”, and “carry on a conversation”, respectively. The first formula express 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. We will all add formulas for the reverse relationship with evidence probability of 0.3. For example, we will ad the following formula.

rel (*talk with a lisp*) $\Rightarrow$ *rel* (*an exchange of ideas via...*), (0.3)

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

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 be equal to 0.9. This probability is high because, for example, someone who is interested in perceiving is probably also interested in one of the ways to perceive. In order to determine the 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.

$rel(to\ become\ aware\ ...) \Rightarrow rel(pay\ attention\ to\ sound),$ (0.23)

$rel\ (to\ become\ aware\ ...) \Rightarrow rel\ (percieve\ by\ sight),$ (0.67)

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 $0.9*1241/4865 = 0.23$. The general idea is that the conditional probabilities to "bigger" senses will be bigger because it is more likely that they are relevant.

We will use an evidence probability of 0.3 for the hypernym relationship. Here is an example.

$rel\ (pay\ attention\ to\ sound\ ) \Rightarrow rel\ (to\ become\ aware...),$ (0.3)

$rel\ (percieve\ by\ sight\ ) \Rightarrow rel\ (to\ become\ aware...),$ (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.

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.

$rel(not\ moving\ quickly) \Rightarrow rel(acting\ or\ moving...),$ (0.6)

$rel(acting\ or\ moving...) \Rightarrow rel(not\ moving\ quickly),$ (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.

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 or habitually" and the sense for the word "prevailing": "most frequent or common" are similar to each other. We will therefore create the following formulas.

$rel(coming\ at\ short\ intervals...) \Rightarrow rel(most\ frequent...),$ (0.8)

$rel(most\ frequent...) \Rightarrow rel(coming\ at\ short\ intervals...),$ (0.8)

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

4. BUILDING THE PROBABILISTIC GRAPH

In this section we describe how to convert the logical formulas from the previous section into a probabilistic graph. Fist, 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 that are generated in the previous section. 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 formulas.

$rel(X) \Rightarrow rel(Y)$, 2.3

$rel(X) \Rightarrow rel(Y)$, 1.1

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

$rel\ (X) \Rightarrow rel\ (Y)$, 3:4

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 weight of the old formula). Lastly, note that adding the weights is consistent with the MLN model. A probability of a world X is computed using the following formula.

[Insert Equation 3]

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 $\omega\ (F)$ 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 will not change the value of the above formula in any world.

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

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

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

$$p = \frac{1}{1+e^{-\omega}}$$

$$edge\ weight = \frac{2 * p - 1}{total}$$

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 so constructed probabilistic graph, 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.

5. MEASURING THE SEMANTIC DISTANCE BETWEEN WORD FORMS

One of the applications of the probabilistic graph is to compute the semantic distance between two word forms in the graph. We will show two ways of performing this calculation as a function of the conditional probability between the word forms.

5.1. Conditional Probability Between Nodes

Let A_0 and A_n be two nodes in the probabilistic graph. We will next describe an efficient way of computing the probability that A_n is relevant given that A_0 is relevant. From probability theory, we have the following formula.

[Insert Equation 4]

[Insert Figure 1]

Let there be a path from A_n to A_0 in the graph – see Figure 1. We will show how to compute the numerator and denominator of the above expression using the probabilities along the path. In other words, for these calculations we assume that we only know the conditional probabilities along a single path. Let ω_i be the weight of the edge from $A_i + 1$ to A_i. We can compute these weights from probabilities using Equations 1 and 2. Let $f00(i)$ be the non-normalized probability from Equation 3 (i.e., we do not divide by *total*) that $A_i + 1$ and A_0 are both irrelevant. Similarly, let $f01(i)$ be the non-normalized probability that $A_i + 1$ is irrelevant and A_0 is relevant, $f10(i)$ be the non-normalized probability that $A_i + 1$ is relevant and A_0 is irrelevant, and $f11(i)$ be the nonnormalized probability that both $A_i + 1$ and A_0 are relevant. In order to understand why we need these functions, note that Equation 4 can be rewritten as follows.

[Insert Equation 5]

The numerator expresses the non-normalized probability that both A_0 and

A_n are relevant. The non-normalized probability of A_n being relevant is computed as $f\,10(n\ 1)+f\,11(n\ 1)$. The reason is that this formula computes the probability of A_n being relevant and A_0 being irrelevant plus the probability of A_n being relevant and A_0 being relevant, which is equal to exactly the probability of A_n being 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, the value for *total* in Equation 3 will be canceled out if added to the formulas.

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

$f00(0) = e\omega^0$

$f01(0) = e\omega^0$

$f10(0) = 1$

$f11(0) = e\omega^0$

The four values follow from Equation 3. Note that we have the following formula and weight.

$rel\,(A_1) \Rightarrow rel\,(A_0),\ \omega_0$

For example, if A_1 is not relevant and A_0 is not relevant, then the formula $rel\,(A_1) \Rightarrow rel\,(A_0)$ will be true and according to Equation 3 the non-normalized probability for this world will be equal to $e\omega^0$. However, if A_1 is relevant and A_0 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.

[Insert Equations 6, 7, 8, 9]

Let us examine the first formula in details. In this case, we want to compute the non-normalized probability ofthe world where both A_0 and A_i $+1$ are irrelevant. We have two sub-cases: when A_i is relevant and when A_i is irrelevant. When A_i is relevant, the following formula will be true.

[Insert Equation 10]

We will therefore add to the probability $f00(i-1) * e^{\omega i}$ in this case. We use the

expression $f00(i-1)$ because we know that both A_i and A_0 are irrelevant in this sub-case. The second sub-case is when A_i is irrelevant. The above formula will be true again and therefore we add to the probability $f10(i-1) * e^{\omega i}$.

Next, let us examine Equation 7. In this case, we want to compute the non-
$-\quad *$
normalized probability of the world where A_0 is irrelevant, but A_i $+1$ is relevant. We have two subcases: when A_i is relevant and when A_i is irrelevant. When A_i is irrelevant, Equation 10 does not hold and therefore will add the probability $f00(i-1)*e^0$. The second subcase is when A_i is relevant and we will add the probability $f10(i-1) * e^{\omega i}$ because Equation 10 holds. Equations 8 and 9 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 0 and then it applies Equations 6 through 9 with values for i from 2 to $n-1$.

5.2. Linear and Logarithmic Distance Metrics

Consider two nodes X and Y in the probabilistic graph. There can be multiple paths between them. Our approach is to compute $P(rel(Y)|rel(X))$ using

Equation 5 as explained in the previous section for each acyclic path Pt between X and Y . We then convert each probability to a weight and then we add the weights. As we explained in last section, adding the weights of identical formulas follows the MLN approach. Finally, the total weight of all paths between X and Y can be converted back to a probability using the following formula.

[Insert Equation 11]

From now on, we will use the shorthand $P\ (Y|X)$ to denote the result of computing $P(rel(Y)|rel(X))$ using the algorithm that was just presented. Note that one may consider our approach coarse because the paths from X to Y can overlap. A more precise approach would be to compute $P(rel(Y)|rel(X))$ from first principles similar to the way we computed the conditional probability along a path in the previous section. However, the computations can get very involved and we leave this approach for future research.

Next, we present two functions for measuring semantic similarity between word forms. The linear function is shown in Equation 12.

[Insert Equation 12]

The minimum function is used in order to to cap the value of the similarity function at one. α is a coefficient that amplifies the available evidence ($\alpha \leq 1$). The experimental section of the article shows how the value of α 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 13. 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.

[Insert Equation 13]

Given two nodes, the similarity between them is computed by performing a depth-first traversal of the graph from one of the nodes. The algorithm runs in linear time relative to the number of visited nodes. Given a path Pt with conditional probabilities $\{p\}_{i=1}^{n}$, we prune it out if $\prod_{i=1}^{n} p_i$ falls bellow 0.0002. This helps us eliminate paths that contain very weak evidence. It turns out that this is the only pruning condition that we need because the pruning value decreases pretty quickly as we add more edges to a path.

6. EXPERIMENTAL EVALUATION

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 [32]. All experiments were performed on a multi-core machine. 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 and the average time for computing the similarity distance between two word forms is about 100 milliseconds.

6.1. The Millers and Charles Benchmark

The goal of this section is to evaluate the quality of the data in the similarity graph. We use the system to compute the similarity of 28 pairs of words from the Miller and Charles study [20]. The study presented the words to humans and computed the mean score of the human ranking.

We ran both the linear and the logarithmic algorithm with different value for α. We got best results for $\alpha = 0.28$ for both the linear and logarithmic metrics. The results are shown in Table 1. Note that if the average of the conditional probability is below 0.00001, then we make it equal to 0.00001 because we believe that evidence with such low level of confidence is just noise. Note that this rounding up is done before we apply the logarithmic or linear metric.

[Insert Table 1]

Table 2 show the correlation of the results of the Miller and Charles study with the results of different algorithms. As the table suggests, our approach produces results of better quality then previous approaches that consider only the structured or unstructured data in WordNet. Our approach is also comparable to the result from [34]. In that paper, we also showed a linear and a logarithmic algorithm. The difference is that the graph was not probabilistic and we did not use the MLN approach. In [34], both the linear and logarithmic approach gave correlation of 0.93 on the Miller and Charles benchmark. The results in this paper are similar and we believe that the difference is due to better parameter adjustment in [34].

6.2. The WordSimilarity-353 Benchmark

[Insert Table 2]

[Insert Table 3]

Next, we discuss the results of applying our algorithm to the WordSimilarity-353 dataset [6]. It contains 353 word pairs. Thirteen humans were used to rate the similarity between each pair of words and give a score between 1 and 10 (10 meaning that the words have the same meaning and 1 meaning that the words are unrelated). The average similarity rating for each word pair was recorded.

We ran the experiments with value for $\alpha = 0.28$, which is the value that produced the best results for the Miller and Charles benchmark. Table 3 shows how our system compares with nine existing systems that have documented their performance on the WordSimilarity-353 benchmark. As the table suggests, our system produces better results than all systems that examine only structured or only unstructured information. Note that some algorithms use additional information from the web [2], while our algorithm only uses information from WordNet. The results are similar to the results from our previous work [34] and we believe that the difference is due to the way the parameters of the algorithm are adjusted.

7. CONCLUSION AND FUTURE RESEARCH

We presented an algorithm for building a proababalistic graph from WordNet. We verified the data quality of the algorithm on both the Miller and Charles and WordSimilarity-353 word pairs benchmarks. We believe that the high quality of the data is due to the fact that our algorithm processes not only structured data, but also natural language information from WordNet. The experimental results are similar to the results in [34]. However, this paper uses the MLN model and a sound probabilistic approach.

In the future, we plan on continuing our work with the MLN model. Specifically, we want to extend our algorithm to compute the conditional probability between two nodes in the graph from first principles. This will eliminate the double counting that we currently do when different paths between two nodes share edges. The other area for future research is extending the probabilistic graph with information from Wikipedia. The biggest challenge there will be the efficiency of the algorithm because Wikipedia contains about 10GB of data as compared to about 100 MB of data in WordNet.

8. REFERENCES

[1] R. Blanco, P. Mika, and S. Vigna. Effective and efficient entity search in RDF data. The Semantic Web? ISWC, pages 83–97, 2011.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1998.

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

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

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

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

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

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

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

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

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

[27] S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond, foundations and trends in information retrieval. Foundations and Trends in

Information Retrieval, 3(4), 2009.

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

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

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

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

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

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

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

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

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

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

[38] M. Steyvers and J. Tenenbaum. The Large-Scale Structure of Semantic Networks:

Statistical Analyses and a Model of Semantic Growth. Cognitive Science, 29(1):41–78, 2005.

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

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

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

Equation 1

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

Equation 2

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

Equation 3

$$P(X) = \frac{1}{total} e^{(\sum_{F} w(F) * |F(X)|)}$$

Equation 4

$$P(rel(A_0)|rel(A_n)) = \frac{P(rel(A_0) \wedge rel(A_n))}{P(rel(A_n))}$$

Figure 1: Example path in the graph

$$\begin{tikzcd} A_n \arrow[r, "w_{n-1}"] & A_{n-1} & \cdots & A_1 \arrow[r, "w_0"] & A_0 \end{tikzcd}$$

Equation 5

$$\frac{P(rel(A_0) \wedge rel(A_n))}{P(rel(A_n))} = \frac{f11(n-1)}{f10(n-1) + f11(n-1)}$$

Equation 6

$$f00(i) = f00(i-1) * e^{w_i} + f10(i-1) * e^{w_i}$$

Equation 7

$$f10(i) = f00(i-1) * 1 + f10(i-1) * e^{w_i}$$

Equation 8

$$f01(i) = f01(i-1) * e^{w_i} + f11(i-1) * e^{w_i}$$

Equation 9

$$f11(i) = f01(i-1) * 1 + f11(i-1) * e^{w_i}$$

Equation 10

$$rel(A_i + 1) \Rightarrow rel(A_i), w_i$$

Equation 11

$$p = 2 * (\frac{1}{1 + e^{-w}}) + 1$$

Equation 12

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

Equation 13

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

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

word 1	*word 2*	*M&C*	*Linear*	*Logarithmic*
car	automobile	3.92	0.99	0.99
gem	jewel	3.84	1.0	1.0
journey	voyage	3.84	1.00	1.00
boy	lad	3.76	0.72	0.80
coast	shore	3.7	0.95	0.96
asylum	madhouse	3.61	1.0	1.00
magician	wizard	3.5	1.0	1.00
midday	noon	3.42	0.95	0.96
furnace	stove	3.11	0.98	0.98
food	fruit	3.08	0.03	0.26
bird	cock	3.05	0.63	0.73
bird	crane	2.97	0.76	0.82
tool	implement	2.95	0.78	0.83
brother	monk	2.82	0.41	0.59
crane	implement	1.68	0.00004	0.11
lad	brother	1.66	0.53	0.67
journey	car	1.16	0.10	0.36
monk	oracle	1.1	0.00004	0.18
food	rooster	0.89	0.42	0.60
coast	hill	0.87	0.00004	0.11
forest	graveyard	0.84	0.00004	0.11
monk	slave	0.55	0.00004	0.11
coast	forest	0.42	0.00004	0.11
lad	wizard	0.42	0.00004	0.11
chord	smile	0.13	0.00004	0.11
glass	magician	0.11	0.00004	0.11
noon	string	0.08	0.00004	0.11
rooster	voyage	0.08	0.00004	0.11

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

algorithm	*correlation*
Hirst and St-Onge [9]	0.74
Leacock and Chodorow citeLC98	0.82
Resnik [23]	0.77
Jiang and Conrath [12]	0.85
Lin [18]	0.83
$\lvert\cdot\rvert_{lin}$ [34]	0.93
$\lvert\cdot\rvert_{log}$ [34]	0.93
$\lvert\cdot\rvert_{lin}$	0.89
$\lvert\cdot\rvert_{log}$	0.90

Table 3: Correlation results with the WordSimilarity-353 benchmark.

algorithm	*correlation*
Jarmasz[10]	0.27
Hirst and St-Onge[9]	0.34
Jiang and Conrath [12]	0.34
Strube and Ponzetto[40]	0.19-0.48
Leacock and Chodrow[17]	0.36
Lin[18]	0.36
Resnik[23]	0.37
Bollegala et al.[2]	0.50
$\|\cdot\|_{lin}$ [34]	0.54
$\|\cdot\|_{log}$ [34]	0.54
$\|\cdot\|_{lin}$	0.52
$\|\cdot\|_{log}$	0.53