





Comparing Graph Similarity Measures for Semantic Representations of Documents

Rubén Manrique^(✉) , Felipe Cueto-Ramirez , and Olga Mariño 

Systems and Computing Engineering Department, School of Engineering,
Universidad de los Andes, Bogotá, Colombia
{rf.manrique,f.cueto10,olmarino}@uniandes.edu.co

Abstract. Documents semantic representations built from open Knowledge Graphs (KGs) have proven to be beneficial in tasks such as recommendation, user profiling, and document retrieval. Broadly speaking, a semantic representation of a document can be defined as a graph whose nodes represent concepts and whose edges represent the semantic relationships between them. Fine-grained information about the concepts found in the KGs (e.g. DBpedia, YAGO, BabelNet) can be exploited to enrich and refine the representation. Although this kind of semantic representation is a graph, most applications that compare semantic representations reduce this graph to a “flattened” concept-weight representation and use existing well-known vector similarity measures. Consequently, relevant information related to the graph structure is not exploited. In this paper, different graph-based similarity measures are adapted to semantic representation graphs and are implemented and evaluated. Experiments performed on two datasets reveal better results when using the graph similarity measures than when using vector similarity measures. This paper presents the conceptual background, the adapted measures and their evaluation and ends with some conclusions on the threshold between precision and computational complexity.

1 Introduction

In recent years, great efforts have been made in the development of technologies and applications that incorporate semantic models exploiting the relational knowledge found in Knowledge Graphs (KG). A Knowledge Graph is defined as a large group of facts about a set of entities described by the classes that compose it and instances of these classes in a particular ontology [5]. KGs like DBpedia¹, Yago² and BabelNet³ incorporate knowledge, which is freely accessible and supported by the mature technologies of the Semantic Web, from multiple domains.

¹ <http://dbpedia.org/>.

² www.yago-knowledge.org/.

³ <http://babelnet.org/>.

To respond to the nature of this large, open, machine-readable knowledge, new representations of semantically-enriched documents have been used in different tasks such as content recommendation [10,14], user profiling [11,17], document retrieval [3,20] and query reformulation [12]. These representations are constructed from the concepts identified in the textual information of the document through Named Entity Recognition and Entity Linking tools. Using these concepts, novel algorithms have been developed to extract new, related relevant information from the KGs [10,15,18]. This extracted information is interconnected and expresses relationships between the concepts at the type level (classes), topics and hierarchies (categories), and characteristics expressed through the properties defined by the KG ontology.

In essence, semantic representations are graphs whose nodes represent concepts and whose edges represent the existence of a semantic relationship between the connected nodes. Instead of exploiting this multidimensional graph structure, most applications use a “flattened” version that considers the set of nodes identified in combination with some weighting measure [10,17,20]. Even though this weighting measure may consider the importance or interconnection of the concept in the graph, much of the structural information of the graph is discarded due to flattening into vectors. The previous problem can be attributed to the fact that applications require the computation of distances and similarities between the document representations. Vectorial representations can be implemented efficiently and it is also possible to apply simple algorithms to them, such as cosine similarity.

As a result of recent developments in graph matching, different algorithms have been proposed to compare graph-based representations [1]. These algorithms can be used to compare two semantic representations without flattening the representations. However, some of these proposals need to be adjusted to deal with the characteristics of semantic representations, particularly the absence of a common set of nodes between two representations. In this paper, we focus on comparing different graph similarity measures proposed in the literature about semantic representations. Using a semantic representation proposed by the authors in previous work, we implement different algorithms to calculate similarities. To compare these algorithms, we use two different datasets. The Lee50 dataset [9] consists of a set of short documents in which each pair of documents is scored according to their semantic relatedness by ten human annotators. The other set, a scholarly paper recommendation dataset [10], contains the profiles of eleven users and a corpus of more than 5000 academic papers both represented through semantic representations.

2 Related Work

With the growth and popularity of KGs, the use of semantic representations that exploit semantic content has been increasing. Semantic representations have been used to enrich vector spaces in information retrieval tasks. First, [20] shows that including a semantic layer that takes advantage of the connectivity and

hierarchical information of the concepts in the KG improves traditional text-based retrieval. Later, [3] proposes a representation for queries and documents using multiple semantic layers that exploit information from multiple KGs. These semantic layers include the unified resource identifier (URI) of each annotated concept to link them to DBpedia, YAGO and to a frame containing temporal values explicitly expressed in the text or associated with DBpedia concepts. In content-based recommendation tasks, semantic representations have been used for user modeling in social networks [17] and modeling user research interests [10,11]. These applications, which are based on semantic representation, have shown superior results in comparison with other representations such as the classical bag of words vector space model. Nevertheless, they do not exploit the structural information of the graph that is produced by the semantic connections of the concepts since their measures are based on a flattened representation of the graph.

On the other hand, as a result of recent developments on graph matching, different algorithms have been proposed to solve the problem of comparing graph-based representation [1]. Though little has been explored in this regard for semantic representations, these graph matching algorithms can be adapted in order to support the calculation of similarities. Therefore, it is not necessary to flatten the representation, thus allowing the structure of the graph to be taken into account. Some of the most important measures are presented in the following paragraphs. It is also important to emphasise that there is not a single criterion to choose the best measure since their performance does greatly depend on the characteristics of the graph [16]. As such, experimentation is the most appropriate way to select the best algorithm for the problem at hand [8].

Since the nodes in the semantic representations are unambiguous concepts identified with URIs, we are interested in the algorithms that take advances of this known correspondence between nodes. A basic strategy known as VEO (vertex edge overlap) [16] measures the similarity between two graphs by calculating the overlap between their edges and nodes, ignoring the edge or node weights. GED (graph edit distance) is a more flexible similarity measure that contemplates the differences in edges and nodes as well as the set of associated weights [6]. There are many adaptations of GED; however, we use the bipartite variation of GED [4] to limit algorithm complexity as much as possible.

Another graph similarity measure we consider is signature similarity [16]. This method seeks to create a signature vector of 1 s and 0 s for each graph using the weights of the nodes. Then, it compares the vectors by counting the amount of matches between the two. It normalizes the result and provides measurements of similarity between 0 and 1. A different approach to graph similarity is the different variations that have been proposed for the MCS (Maximum common subgraph) algorithm [2]. The MCS is the largest sub-graph that is common in the considered graphs. Different metrics use the size of the MCS as an indicative of similarity. The size of a sub-graph can be measured in several ways, however, in this paper we will focus on the amount of nodes. This method is particularly useful in biological and chemical analysis [21].

There is little research in which measures of similarity based on graphs have been applied to semantic representations. Specifically, the work done in [18] is the closest to the purpose of our research. The authors present an approach to the calculation of document semantic similarity over graph-based structures. A variation of GED is used to measure the similarity of two semantic representations. Different from the previous work, we implement and compare different graph-based similarity measures on top of a more refined semantic representation that contemplates expansion and filtering processes.

3 Graph-Based Similarity Measures

To properly define and understand these algorithms, we must first address the issue of the set theory context of the problem and define the notation to explain the algorithms. It is important to clarify that vertex and node are one and the same. Therefore, we henceforth understand $G = (V, E)$ as a directed graph, with a set V of vertices and a set E of edges, with both edge and vertex weights. We define an edge as a 2-tuple $e = (o, d)$ with a origin o and a destination d . We will refer to the edge and vertex weight as $w(e)$ and $w(v)$ respectively.

For the complexity of each algorithm we will use Big O notation. The vertices and the edges are both kept in 2 hash tables. For the vertices, we use the label as the hashed key. For the edges we combine the origin and the destination to form a unique hashed key. The size of set V and set E are known. This will allow us to estimate the complexity of the most efficient version of each algorithm.

3.1 Vertex Edge Overlap

Among the simpler algorithms, we find VEO (Vertex edge overlap). This method seeks to simplify the problem of graph matching by counting the total number of vertices and edges that match and dividing the result by the sum of the total number of vertices and of the total number of edges on each graph. This factor is multiplied by 2 in order to normalize the result to the correct scale.

$$VEO(G, G') = 2 \frac{|V \cap V'| + |E \cap E'|}{|V| + |V'| + |E| + |E'|} \quad (1)$$

This algorithm can be applied on any graph structure since it only uses variables found on the graph. Even still, it is also an extremely narrow approach since it does not take information such as vertex or edge weight or path information into account. This approach is based on the simple form of the GED (Graph edit distance) algorithm and is normalized to a scale of 1 to 0, when 1 is completely similar, and 0 is completely dissimilar. The complexity of this algorithm is $O(V + E)$ since it only requires a single iteration over the sets of one of the graphs in order to find the matching pairs in both vertices and edges.

3.2 Node Graph Edit Distance

To properly take advantage of the information found on the semantic models, we devised a highly modified version of GED. The first issue was creating a method that takes the weight of the vertices into account. This algorithm would need to provide a normalized measure of similarity that used the weight of the nodes.

$$GED_{nodes}(G, G') = \frac{\sum w(V) + \sum w'(V') - \sum w(V \cap V') - \sum w'(V \cap V') + \sum |w(V \cap V') - w'(V \cap V')|}{\sum w(V) + \sum w'(V') - \sum w(V \cap V') - \sum w'(V \cap V') + \sum \max(w(V \cap V'), w'(V \cap V'))} \quad (2)$$

We can understand the dividend as the sum of vertex weights found only in G , plus the sum of vertex weights found only in G' , plus the sum of the differences in weights between the vertex intersections between G and G' . The divisor is the the total sum of vertex weights found only in G , plus the total sum of vertex weights found only in G' , plus the sum of the maximum weights between of the vertex intersections between G and G' . The complexity of this algorithm is $O(V + V')$ since it requires iterating over the vertices of both graphs in order to find the sum of weights in each one.

3.3 Edge Graph Edit Distance

Using a similar approach to node graph edit distance, we can obtain the edge graph edit distance. This formula works for both directed and non-directed graphs. With that in mind, we can convert a directed graph into a non-directed graph by adding the weights of corresponding opposite edges. This allows us to obtain a higher, and in some cases, more appropriate similarity measure.

$$GED_{edges}(G, G') = \frac{\sum w(E) + \sum w'(E') - \sum w(E \cap E') - \sum w'(E \cap E') + \sum |w(E \cap E') - w'(E \cap E')|}{\sum w(E) + \sum w'(E') - \sum w(E \cap E') - \sum w'(E \cap E') + \sum \max(w(E \cap E'), w'(E \cap E'))} \quad (3)$$

We can understand the dividend as the total sum of edge weights found only in G , plus the total sum of edge weights found only in G' , plus the total sum of the differences in weights between the edge intersections between G and G' . The divisor is the the total sum of edge weights found only in G , plus the total sum of edge weights found only in G' , plus the sum of the maximum weights of the edge intersections between G and G' . The complexity of this algorithm is $O(E + E')$ since it requires iterating over the edges of both graphs in order to find the total sum of weights in each one.

3.4 Total Graph Edit Distance

Given the values of node similarity and edge similarity, we can create a more complete measure by adding them together.

$$GED(G, G') = \frac{GED_{nodes}(G, G') + GED_{edges}(G, G')}{2} \quad (4)$$

This formula creates a new measurement that equates the similarity weight of vertex similarity and edge similarity. It provides a normalized value between 0 and 1. The complexity of this algorithm is $O(V + V' + E + E')$ since it is the sum of both the edge graph edit distance and the node graph edit distance.

3.5 Maximum Common Sub Graph

The MCS (Maximum Common Subgraph) of two graphs can be calculated by finding the common sub-graph with most nodes. In order to do this, MCS algorithm finds all common sub-graphs, and then calculates the amount of nodes in the largest one. To normalize the result, it divides this amount by the number of nodes in the graph with the most nodes.

$$MCS_{nodes}(G, G') = \frac{|MCS(G, G')|}{\max(|V|, |V'|)} \quad (5)$$

This formula creates a value between 1 and 0, where 1 is completely similar, and 0 is completely dissimilar. The MCS is calculated via Algorithm 1.

Algorithm 1. Maximum common subgraph

```

Require:  $G = \{V, E\}, G' = \{V', E'\}$ 
function  $mcsNodes(G, G')$ 
   $currentNodeMaximum := 0$ 
  for all  $v \in V$  do
     $count := 0$ 
     $visited = []$  ▷ Keeps track of explored nodes in current subgraph
     $result := mcsNodesRecursor(count, visited, v, G, G')$ 
    if  $result[0] > currentNodeMaximum$  then
       $currentNodeMaximum := count$ 
    end if
  end for
  return  $currentNodeMaximum$ 
end function
function  $mcsNodesRecursor(count, visited, v, G, G')$ 
   $outwardEdges := E[v]$  ▷ Gets all out-edges of a node
   $inwardEdges := E[v]$  ▷ Gets all in-edges of a node
  if  $v \notin visited$  then
    if  $v \in V'$  then
       $count := count + 1$ 
       $visited := visited + v$ 
      for all  $e \in outwardEdges$  do
        if  $e \in E'$  then
           $r = e[1]$  ▷ Extracts destination node from edge
           $result := mcsNodesRecursor(count, visited, r, G, G')$ 
           $count := result[0]$ 
           $visited := result[1]$ 
        end if
      end for
    end if
    for all  $e \in inwardEdges$  do
      if  $e \in E'$  then
         $r = e[0]$  ▷ Extracts origin node from edge
         $result := mcsNodesRecursor(count, visited, r, G, G')$ 
         $count := result[0]$ 
         $visited := result[1]$ 
      end if
    end for
  end if
  return  $(count, visited)$ 
end function

```

In order to find a subgraph, Algorithm 1 recursively travels through the common connections between the two considered graphs. The algorithm starts iterating over all the nodes in one of the graphs. Once it finds a matching node, the algorithm adds it to the current subgraph hash-table and iterates through its edges looking for common connections. If an edge match is found, the algorithm adds the destination node and subsequently processes the edges of this new node. The MCS algorithm is usually defined in the context of non-directed graphs [2]. Since our graphs are directed, we adapted the original algorithm to explore nodes through outgoing and incoming edges. We accomplish this by iterating over outgoing edges and then iterating over incoming edges. We keep track of explored nodes by adding their labels to a hash-table. If the two nodes in a pair are connected by both an incoming and an outgoing edge, we only explore the subsequent node once and exclude it upon the second inspection. Once a subgraph has been identified and there are no more matching edges found on the border nodes, the algorithm detects the subgraph size and compares it with the largest subgraph previously found. This algorithm will provide us with the size of the largest possible common subgraph, in terms of nodes included, between two graphs.

Finding the MCS is a np-complete problem. Nevertheless, it is possible to calculate the worst case time complexity. As mentioned in [2], the worst case time complexity of the MCS algorithm is $O((V * V')^V)$.

4 Semantic Representation

In this section, we explain the semantic representation building process in general. The essential information is taken from a KG using the concepts found in the document text. A KG consists of a set of resources⁴ C and literals L that are interrelated through a set of properties/predicates P . Under an RDF model, KG data consists of a set of statements $S \subset C \times P \times (C \cup L)$. Each $s \in S$ is a triplet composed of a subject, a predicate, and an object/literal. For this paper, DBpedia was employed as KG; however, other KGs can also be employed or combined to build the representation.

Once the KG is defined, the representation of a document is constructed following the process depicted in Fig. 1. The process begins with the extraction of the concepts mentioned in the text (i.e. annotations) contained by the document. DBpedia Spotlight⁵ and Babelfy⁶ two automatic entity linking and word sense disambiguation tools were used for this task. Then, the Expansion Module receives the initial set of annotations and expands it through the rich number of relationships in the KG. In this module, new expanded concepts that are not

⁴ Hereafter, we use concept and entity interchangeably to refer to resources of the KG.

⁵ <http://www.dbpedia-spotlight.org/>.

⁶ <http://babelfy.org/>.

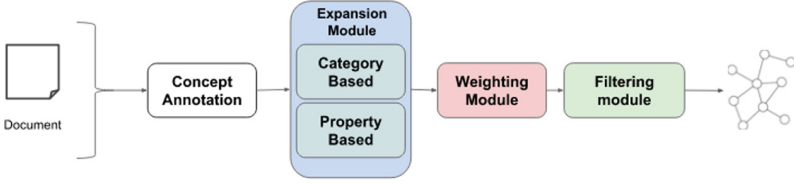


Fig. 1. General overview of the semantic representation process

found in the text, but are related with the annotation, are incorporated into the representation. We follow two different expansion approaches:

- Category-based expansion: We add the hierarchical information of each concept. We find such information in DBpedia through the Dublin Core dct:subject property.
- Property-based expansion: The semantic representation is enriched with the set of resources recovered by following the set of properties of the KG ontology.

As a result of the expansion, an initial set of nodes for the representation is obtained. A weight for each node is assigned by the Weighting Module that checks the importance of each concept for the document. For annotations and expanded concepts, different weighting strategies are employed. Finally, in the Filtering Module, we apply a filtering technique to select concepts that are highly connected such that weakly connected concepts are discarded. The strategy seeks connection paths of length l between annotations because it uses these to create edges in the representation and assign the corresponding edge weight. Using previous results, we limit the path length to a maximum of $l = 2$. From these processes, a graph whose nodes are concepts and edges expressing the existence of a linkage between two concepts in the KG is built. A more detailed description of each of these modules can be found in previous works by the authors [10, 11]. The resulting representation follows Definition 1.

Definition 1. *The semantic representation G_i of a document r_i is a directed weighted graph $G_i = (V_i, E_i, w(r_i, c), w(r_i, e))$, where both nodes and edges have an associated weight defined by the functions $w(r_i, c) : V \rightarrow \mathbb{R}^+$ and $w(e) : E \rightarrow \mathbb{R}^+$. The set of nodes $V_i = \{c_1, c_2, \dots, c_k\}$ are concepts belonging to the space of a KG ($c_k \in C$). The node weight $w(r_i, c)$ denotes how relevant the node c is for the document. A connection edge between two nodes (c_a, c_b) represents the existence of almost one statement s in the KG that links both concepts. The weight of the edge $w(e)$ denotes how relevant this linkage in G_i is.*

The definition above refers to a directed graph to the extent that the direction of the relationships found in the KG are preserved. Nevertheless, it is also possible to build a non-directed version by unifying the vertices that share the same nodes but go in opposite directions. The weight of the resulting non-directed vertex is the sum of the directed opposite vertices. We also use this non-directed version

of the semantic representation to evaluate the contribution of the direction in the similarity calculation. Additionally, even when there is a loss of information, this non-directed version is lighter and reduces the computational cost.

Finally, we define the flattened version of the representation as that which only preserves the set of nodes and their weights (Definition 2). Put simply, edges are removed from the representation. Since it is easy to transform this flattened version into a vector representation, measures such as the cosine similarity, L2-norm or Manhattan distance can be used for the calculation of similarity between documents. For the flattened version, and following previous results [11], we use the cosine similarity.

Definition 2. *The flattened semantic representation R_i of a document r is a set of weighted KG entities/concepts. A weighted concept is a pair $(c, w(r_i, c))$ where the weight $w(r_i, c)$ denotes how important the concept c is for the document r_i , and is computed by a certain function w .*

$$R_i = \{(c, w(r_i, c)) | c \in C\} \quad (6)$$

5 Evaluation

5.1 Datasets

Our evaluation aims to compare the graph similarity measures described above when the graphs are semantic representations of documents. To this end, we select two different datasets that have been used in the past. The first one, **Lee50** [9] is a compilation of 50 short documents collected from the Australian Broadcasting Corporations news mail service. Each possible pair of documents was scored by ten human judges on their semantic relatedness. The final similarity judgment for every pair is obtained by averaging all annotation of the judges, so the final collection contains 1225 relatedness scores. With this dataset we can compare how well the combination of the representation and the graph similarity measures approximate the human notion of similarity. The second dataset, **Man17** [10], was developed for the scholarly paper recommendation task. It contains eleven researcher profiles built from the concepts found in their open publications. For this dataset, the semantic representation is built for research profiles and the candidate corpus of documents (>5000 documents). Different from Lee50, documents in **Man17** are larger since they are academic papers that usually contain more than 2000 words. Hence, the dataset has greater number of concepts in the text and a larger graph is produced in terms of the number of nodes and edges. For each profile, this dataset contains the set of relevant papers from the candidate set. The task here is try to recover relevant papers by comparing the research profile with the candidate corpus (i.e. content base recommendation) using the different graph similarity measures.

In order to evaluate the performance on **Lee50**, we report the Pearson (r) and Spearman correlation (ρ). According to [7], these correlation metrics are appropriate to evaluate relatedness measures and have been used in related work, so

we are able to compare our results to other approaches. For the **Man17** dataset, we use the following typical metrics for the evaluation of Top-N recommender tasks [19]: MRR (Mean Reciprocal Rank), MAP@10 (Mean Average Precision), and NDCG@10 (Normalized Discounted Cumulative Gain). Following the original paper, we select $N = 10$ as the recommendation objective [10]. In this data set, the relevance measures are binaries (i.e. the recommended documents are relevant to the user or not), so we use a binary relevance scale for the calculation of NDCG. The final NDCG is calculated averaging the results for each user profile.

5.2 Semantic Annotators and Path Length

We use DBpedia Spotlight (DBS) and Babelfy to annotate text documents. DBpedia Spotlight allows us to configure the level of annotation (precision/recall trade-off) by confidence/support parameters. Support parameters specify the minimum number of inlinks a DBpedia resource has to have in order to be annotated, while the confidence parameter controls the topical pertinence and the contextual ambiguity to avoid incorrect annotations as much as possible [13]. We define 5 different configurations for DBS: DBS1 (support: 5, confidence: 0.35), DBS2 (support: 5, confidence: 0.40), DBS3 (support: 5, confidence: 0.45), DBS4 (support: 10, confidence: 0.40), and DBS5 (support: 20, confidence: 0.40). We explore the influence of the confidence parameter with the first three configurations. Values higher than 0.45 are not considered since this would significantly reduce the number of annotations obtained. This can be particularly detrimental in short documents such as those handled in Lee50. For the support parameter we use values of 5, 10 and 20; our hypothesis is that the identification of highly specialized concepts may be affected by this parameter. For Babelfy, no special configuration was used and the complete set of annotations recovered are used in the semantic representation building process.

As previously mentioned, the semantic representation input parameter is the path length (l) that specifies the maximum depth to look for connections between concepts in the KG. This parameter affects the edge composition, and thereby the graph structure. We want to explore the effect of this parameter in the graph similarity measures, so we define values of path length of $l = 1$ and $l = 2$.

6 Results

We report our results on the **Lee50** dataset in Table 1. Each column in the table represents one of the similarity measures presented above. To differentiate the results obtained for the directed and non-directed versions of the semantic representations, we use the letter D to indicate directed or U to mean undirected. The column *Flattened* presents the results obtained with the flattened version of the representation (Definition 2). Finally, the column *NodeAvg* presents the average of nodes in each case. For each column, the best result at the level of each correlation measure is highlighted in bold.

Table 1. Lee50 dataset results. Correlation measure: Pearson (r) and Spearman (ρ). l : path length parameter

		$DGED_{node}$	$DGED_{edge}$	$DGED$	$UGED_{node}$	$UGED_{edge}$	$UGED$	$UVEO$	$DVEO$	MCS	<i>Flattened</i>	<i>NodeAvg.</i>	<i>EdgeAvg.</i>		
r	$l=1$	<i>Babelfy</i>	0.4629	0.3702	0.4225	0.4629	0.3321	0.3907	0.416	0.415	0.3427	0.398	289	313	
		<i>DBS1</i>	0.6582	0.5733	0.6244	0.6582	0.5572	0.6005	0.6138	0.613	0.5548	0.593	254	277	
		<i>DBS2</i>	0.624	0.546	0.5922	0.624	0.5171	0.5601	0.5819	0.5812	0.5300	0.564	242	264	
		<i>DBS3</i>	0.5631	0.4803	0.5291	0.5631	0.4709	0.5169	0.5414	0.5405	0.4770	0.509	242	265	
		<i>DBS4</i>	0.604	0.5261	0.5723	0.604	0.5172	0.5602	0.5819	0.5812	0.5300	0.551	237	260	
	$l=2$	<i>DBS5</i>	0.6035	0.5265	0.5722	0.6035	0.5176	0.5603	0.5817	0.581	0.5299	0.551	230	253	
		<i>Babelfy</i>	0.4629	0.371	0.4262	0.4629	0.3282	0.3935	0.4159	0.4149	0.3427	0.399	289	314	
		<i>DBS1</i>	0.6592	0.5667	0.6256	0.6592	0.5254	0.601	0.6136	0.6128	0.5543	0.594	254	278	
		<i>DBS2</i>	0.6239	0.5401	0.5936	0.6239	0.5125	0.5613	0.5819	0.5811	0.5297	0.563	242	265	
		<i>DBS3</i>	0.563	0.4784	0.5323	0.563	0.471	0.5207	0.5414	0.5405	0.4768	0.509	242	266	
	ρ	$l=1$	<i>DBS4</i>	0.6039	0.5202	0.5736	0.6039	0.5128	0.5615	0.5819	0.5811	0.5297	0.550	237	261
			<i>DBS5</i>	0.6035	0.5217	0.574	0.6035	0.5147	0.5624	0.5816	0.5809	0.5296	0.551	230	254
			<i>Babelfy</i>	0.3057	0.0909	0.2315	0.3057	0.1012	0.1841	0.2233	0.2219	0.0870	0.194	289	313
			<i>DBS1</i>	0.5164	0.3724	0.5149	0.5164	0.4714	0.5168	0.5146	0.5139	0.1063	0.472	254	277
			<i>DBS2</i>	0.4874	0.3614	0.4871	0.4874	0.4281	0.4883	0.4863	0.4857	0.1081	0.444	242	264
$l=2$		<i>DBS3</i>	0.4535	0.3263	0.4534	0.4535	0.4258	0.454	0.4529	0.4525	0.0716	0.414	242	265	
		<i>DBS4</i>	0.4875	0.3614	0.4872	0.4875	0.4281	0.4883	0.4863	0.4858	0.1082	0.444	237	260	
		<i>DBS5</i>	0.4843	0.3614	0.4839	0.4843	0.428	0.4851	0.483	0.4825	0.1141	0.442	230	253	
		<i>Babelfy</i>	0.3057	0.0892	0.2374	0.3057	0.0979	0.1898	0.2232	0.2217	0.0870	0.196	289	314	
		<i>DBS1</i>	0.5165	0.3723	0.515	0.5165	0.4645	0.5163	0.5147	0.5141	0.1063	0.471	254	278	
$l=2$		<i>DBS2</i>	0.4875	0.3613	0.4872	0.4875	0.4207	0.4875	0.4862	0.4859	0.1081	0.443	242	265	
		<i>DBS3</i>	0.4535	0.3264	0.4534	0.4535	0.4196	0.4534	0.4529	0.4526	0.0717	0.414	242	266	
		<i>DBS4</i>	0.4876	0.3613	0.4873	0.4876	0.4207	0.4875	0.4863	0.486	0.1082	0.444	237	261	
		<i>DBS5</i>	0.4843	0.3613	0.484	0.4843	0.4211	0.4846	0.4831	0.4827	0.1141	0.441	230	254	

The best results were obtained using the $DGED_{node}$. Since there is no difference in nodes between the directed and undirected version of the graph, $DGED_{node}$ is equivalent to $UGED_{node}$. We prefer $DGED_{node}$ because the additional step of merging edges is avoided. The superiority of this similarity measure is independent, in most cases, of other elements such as the annotation service and the path length. Regarding the contribution of the edges as the similarity measures $DGED_{edge}$ and $UGED_{edge}$, the following is shown: (a) the direction seems to favor the Pearson correlation; however, better results were obtained at the Spearman correlation level using the undirected version; and (b) surprisingly, when combining the contribution of the edges and nodes ($DGED$ and $UGED$), there is no improvement in the results obtained compared to $DGED_{node}$ and $UGED_{node}$. The behavior described in (a) can be attributed to increases in non-equivalent magnitude among the variables that are evaluated, in particular in the undirected version the changes of similarity are very low in comparison with its directed version. While the Pearson correlation is usually strongly affected by this, the Spearman correlation does not. Furthermore, (b) seems to indicate that the contribution of the edges is not so important compared to the node contribution, at least when compared to the similarity established by the human annotators. However, we believe that edges provide additional semantic information that increases the relatedness between documents on a deeper level, and this might be overlooked by human annotators when comparing text documents.

The results obtained with the VEO measure, for both the directed (DVEO) and undirected (UVEO) versions, are interesting since both have an excellent trade-off between the computational complexity and the performance. Since VEO does not consider the edge or node weight, we also highlight that it is

possible to reduce the time to construct the semantic representation by discarding the weighting module. Thus, VEO is an interesting alternative for critical response time applications.

In accordance with the aforementioned, another suitable way to reduce the computational cost is to consider unitary path lengths $l = 1$. Results in Table 1 are not conclusive about an improvement when a longer path length is selected. In the cases where the correlation measures are improved by the selection of $l = 2$, the difference in the values obtained by its counterpart $l = 1$ does not seem to be significant (i.e. less than 2%). In contrast, the selection of connections path of $l = 1$ significantly reduces the complexity and/or the number of queries that must be sent to the KG.

When considering the Spearman correlation, the MCS algorithm performs poorly. The similarity measures obtained via this algorithm present the highest variance, so more appropriate ways to normalize *MCS* (Eq. 5) should be explored.

Clearly, the results show the superiority of *DBS1* as annotation service. Independent of the graph similarity measure or path length selected, *DBS1* outperforms all the annotation services considered. Babelfy has a high number of false positives that negatively influence the representation. Although it seems not to be properly documented, Babelfy provides confidence measures that are associated with each recovered annotation that can be exploited for future filtering strategies. In the case of Spotlight service, small increases in the level of confidence strongly affect the number of concepts in the final representation. The support parameter, on the other hand, does not seem to be so decisive in the final representation and thus the similarity in the results obtained.

Table 2 lists the performance for our best-performing similarity measures (obtained via *DGED_{node}* similarity measure, *DBS1* as annotation tool and a semantic representation build with $l = 2$ as input parameter), as well as for the following related baselines:

- Salient Semantic Analysis (*SSA*): a concept-base strategy which incorporates a similar semantic abstraction and interpretation of words, by using the linkage of concepts in Wikipedia [7].
- Graph-based document similarity (*GDS*): Similar to this work, a semantic graph using KGs is constructed. The representation in this case is basically a KG subgraph on the basis of the annotated concepts. No refinement processing is performed.
- Vector Space Model (*VSM*): the cosine distance of a standard bag-of-words Vector Space Model. We carried out typical text processing operations including tokenizer, stop word removal and stemming.

In general, very competitive results of our best measure of similarity are observed. At the level of the Spearman correlation (ρ) we present the best results; however, *SSA* is superior in terms of the Pearson correlation (r). There is a relative improvement of 15.4% over *VSM* and 4.6% over *GDS* at the Pearson correlation level.

Table 2. Comparison with related work for Lee50 dataset. Correlation measures: Pearson (r) and Spearman (ρ)

	r	ρ
SSAs [7]	0.684	0.488
$DGED_{node}$	0.659	0.516
GDS [18]	0.63	-
VSM	0.571	0.402

Table 3 shows the results obtained from the **Man17** dataset. For this dataset, we report the results obtained by the semantic representation using *DBS1* as annotation service. The results of the flattened version were taken from [11] when the full text of the paper was used as input for the semantic representation. We also report the results obtained via VSM.

Table 3. Man17 dataset results.

		$DGED_{node}$	$DGED_{edge}$	$DGED$	$UGED_{node}$	$UGED_{edge}$	$UGED$	$UVEO$	$DVEO$	MCS	Flattened	VSM
$l = 1$	MRR	0.4781	0.4750	0.5000	0.4781	0.2019	0.2786	0.4047	0.4100	0.418	0.5820	0.401
	MAP	0.5102	0.5037	0.5092	0.5102	0.3029	0.3492	0.4871	0.4641	0.330	0.4978	0.3456
	NDCG	0.7060	0.6391	0.6478	0.7060	0.4071	0.4121	0.6174	0.6449	0.424	0.6621	0.4234
$l = 2$	MRR	0.4674	0.4386	0.503	0.4674	0.2719	0.3355	0.4902	0.4478	0.311	0.4297	0.401
	MAP	0.5064	0.4838	0.5007	0.5064	0.3257	0.3568	0.4903	0.4764	0.269	0.4534	0.3456
	NDCG	0.6766	0.5808	0.6620	0.6766	0.4216	0.4602	0.6154	0.647	0.326	0.6161	0.4234

Consistent with the results obtained in Lee50, $DGED_{node}$ presents the best results in terms of MAP and NDCG. The flattened version is better in terms of the MRR. In this dataset, the direction of the edges are more relevant for the recommendation quality. Indeed, there is a significant difference between the values obtained by $DGED_{edge}$ and $UGED_{edge}$. The results also show that a path of size $l = 1$ is more appropriate for this task.

7 Conclusion

In this paper we have compared the performance of different graph-based similarity algorithms with two different datasets that employ semantic representations. One of the datasets is focused on the similarity between short documents, while the second is focused on the recommendation of academic papers. For each dataset, different evaluation measures were used. The graph yielded better results in comparison with the flattened version of the semantic representation.

The results suggest that GED_{nodes} , an algorithm based on comparing the weighted nodes of both graphs, is an appropriate measure and that it is not necessary to consider the edges of the graph. This goes slightly against our initial hypothesis which suggested that the edges connecting the nodes in the semantic

graph express relevant information of the described document and should thus be taken into account. However, as the baseline used was the similarity explicitly indicated by humans, two hypotheses should be further explored: either edges do not add significant information; or, when comparing documents, humans look at the broad picture and if examining a more detailed relation, they might reduce the initial similarity degree.

The computational complexity will also be taken into account in order to select the most appropriate similarity measure. It should be noted that the current implementation of the algorithms discussed was done on `networkx`, a graph library for python. Consequently, the complexity achieved was much higher than theoretically possible. For VEO, we obtained a complexity of $O(V + V' + E + E')$ since the algorithm had to create the hash tables to avoid an even higher processing cost. For node graph edit distance, we obtained a complexity of $O(2V + 2V')$, and for edge graph edit distance, $O(V + V' + 2E + 2E')$ for the same reasons.

Our future work will focus on exploring other graph matching algorithms, in particular, those with a low computational complexity that can be implemented in real recommendation scenarios and/or information retrieval applications. Additionally, we would like to explore other ways to evaluate the edge correspondence based on paths analysis.

Acknowledgment. This work was partially supported by COLCIENCIAS PhD scholarship (Call 647-2014).

References

1. Bunke, H.: Recent developments in graph matching. In: Proceedings 15th International Conference on Pattern Recognition, ICPR-2000, vol. 2, pp. 117–124 (2000)
2. Bunke, H., Shearer, K.: A graph distance metric based on the maximal common subgraph. *Pattern Recognit. Lett.* **19**(3), 255–259 (1998)
3. Corcoglioniti, F., Dragoni, M., Rospocher, M., Apro시오, A.P.: Knowledge extraction for information retrieval. In: Sack, H., Blomqvist, E., d’Aquin, M., Ghidini, C., Ponzetto, S.P., Lange, C. (eds.) *ESWC 2016*. LNCS, vol. 9678, pp. 317–333. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-34129-3_20
4. Fankhauser, S., Riesen, K., Bunke, H.: Speeding up graph edit distance computation through fast bipartite matching. In: Jiang, X., Ferrer, M., Torsello, A. (eds.) *GbRPR 2011*. LNCS, vol. 6658, pp. 102–111. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20844-7_11
5. Färber, M., Ell, B., Menne, C., Rettinger, A.: A comparative survey of DBpedia, freebase, OpenCyc, Wikidata, and YAGO. *Semant. Web J.* 1–26 (2015)
6. Gao, X., Xiao, B., Tao, D., Li, X.: A survey of graph edit distance. *Pattern Anal. Appl.* **13**(1), 113–129 (2010)
7. Hassan, S., Mihalcea, R.: Semantic relatedness using salient semantic analysis. In: *AAAI* (2011)
8. Jouili, S., Tabbone, S., Valveny, E.: Comparing graph similarity measures for graphical recognition. In: Ogier, J.-M., Liu, W., Lladós, J. (eds.) *GREC 2009*. LNCS, vol. 6020, pp. 37–48. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13728-0_4

9. Lee, M.D., Welsh, M.: An empirical evaluation of models of text document similarity. In: CogSci 2005, pp. 1254–1259. Erlbaum (2005)
10. Manrique, R., Herazo, O., Mariño, O.: Exploring the use of linked open data for user research interest modeling. In: Solano, A., Ordoñez, H. (eds.) CCC 2017. CCIS, vol. 735, pp. 3–16. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66562-7_1
11. Manrique, R., Mariño, O.: How does the size of a document affect linked open data user modeling strategies? In: Proceedings of the International Conference on Web Intelligence, WI 2017, pp. 1246–1252. ACM, New York (2017)
12. Manrique, R., Mariño, O.: Diversified semantic query reformulation. In: Rózewski, P., Lange, C. (eds.) KESW 2017. CCIS, vol. 786, pp. 23–37. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69548-8_3
13. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: Dbpedia spotlight: shedding light on the web of documents. In: Proceedings of the 7th International Conference on Semantic Systems, I-Semantics 2011, pp. 1–8. ACM, New York (2011)
14. Musto, C., Lops, P., de Gemmis, M., Semeraro, G.: Semantics-aware recommender systems exploiting linked open data and graph-based features. *Knowl.-Based Syst.* **136**, 1–14 (2017)
15. Nunes, B.P., Fetahu, B., Kawase, R., Dietze, S., Casanova, M.A., Maynard, D.: Interlinking documents based on semantic graphs with an application. In: Tweedale, J.W., Jain, L.C., Watada, J., Howlett, R.J. (eds.) Knowledge-Based Information Systems in Practice. SIST, vol. 30, pp. 139–155. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-13545-8_9
16. Papadimitriou, P., Dasdan, A., Garcia-Molina, H.: Web graph similarity for anomaly detection. *J. Internet Serv. Appl.* **1**(1), 19–30 (2010)
17. Piao, G., Breslin, J.G.: Analyzing aggregated semantics-enabled user modeling on google+ and twitter for personalized link recommendations. In: Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, UMAP 2016, pp. 105–109. ACM, New York (2016)
18. Schuhmacher, M., Ponzetto, S.P.: Knowledge-based graph document modeling. In: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, WSDM 2014, pp. 543–552. ACM, New York (2014)
19. Sugiyama, K., Kan, M.Y.: A comprehensive evaluation of scholarly paper recommendation using potential citation papers. *Int. J. Digit. Libr.* **16**(2), 91–109 (2015)
20. Waitelonis, J., Exeler, C., Sack, H.: Enabled generalized vector space model to improve document retrieval. In: Proceedings of the Third NLP & DBpedia Workshop (NLP & DBpedia 2015) Co-located with the 14th International Semantic Web Conference 2015 (ISWC 2015), 11 October 2015, Bethlehem, Pennsylvania, USA, pp. 33–44 (2015)
21. Willett, P.: Matching of chemical and biological structures using subgraph and maximal common subgraph isomorphism algorithms. In: Truhlar, D.G., Howe, W.J., Hopfinger, A.J., Blaney, J., Dammkoehler, R.A. (eds.) Rational Drug Design, vol. 108, pp. 11–38. Springer, New York (1999). https://doi.org/10.1007/978-1-4612-1480-9_3