CoSimRank: A Flexible & Efficient Graph-Theoretic Similarity Measure

Sascha Rothe and Hinrich Schütze
Center for Information & Language Processing
University of Munich
sascha@cis.lmu.de

Abstract

We present *CoSimRank*, a graph-theoretic similarity measure that is efficient because it can compute a single node similarity without having to compute the similarities of the entire graph. We present equivalent formalizations that show CoSimRank's close relationship to Personalized PageRank and SimRank and also show how we can take advantage of fast matrix multiplication algorithms to compute CoSimRank. Another advantage of CoSimRank is that it can be flexibly extended from basic node-node similarity to several other graph-theoretic similarity measures. In an experimental evaluation on the tasks of synonym extraction and bilingual lexicon extraction, CoSimRank is faster or more accurate than previous approaches.

1 Introduction

Graph-theoretic algorithms have been successfully applied to many problems in NLP (Mihalcea and Radev, 2011). These algorithms are often based on PageRank (Brin and Page, 1998) and other centrality measures (e.g., (Erkan and Radev, 2004)). An alternative for tasks involving similarity is SimRank (Jeh and Widom, 2002). SimRank is based on the simple intuition that nodes in a graph should be considered as similar to the extent that their neighbors are similar. Unfortunately, SimRank has time complexity $\mathcal{O}(n^3)$ (where n is the number of nodes in the graph) and therefore does not scale to the large graphs that are typical of NLP.

This paper introduces CoSimRank,[1] a new graph-theoretic algorithm for computing node similarity that combines features of SimRank and PageRank. Our key observation is that to compute the similarity of two nodes, we need not consider all other nodes in the graph as SimRank does; instead, CoSimRank starts random walks from the two nodes and computes their similarity at each time step. This offers large savings in computation time if we only need the similarities of a small subset of all n^2 node similarities.

These two cases – computing a few similarities and computing many similarities – correspond to two different representations we can compute CoSimRank on: a vector representation, which is fast for only a few similarities, and a matrix representation, which can take advantage of fast matrix multiplication algorithms.

CoSimRank can be used to compute many variations of basic node similarity – including similarity for graphs with weighted and typed edges and similarity for sets of nodes. Thus, CoSimRank has the added advantage of being a flexible tool for different types of applications.

The extension of CoSimRank to *similarity across graphs* is important for the application of bilingual lexicon extraction: given a set of correspondences between nodes in two graphs A and B (corresponding to two different languages), a pair of nodes $(a \in A, b \in B)$ is a good candidate for a translation pair if their node similarity is high. In an experimental evaluation, we show that CoSimRank is more efficient and more accurate than both SimRank and PageRank-based algorithms.

This paper is structured as follows. Section 2 discusses related work. Section 3 introduces CoSimRank. In Section 4, we compare CoSimRank and SimRank. By providing some useful extensions, we demonstrate the great flexibility of CoSimRank (Section 5). We perform an experimental evaluation of CoSimRank in Section 6. Section 7 summarizes the paper.

2 Related Work

Our work is unsupervised. We therefore do not review graph-based methods that make extensive

[1] Code available at code.google.com/p/cistern

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1392–1402,
Baltimore, Maryland, USA, June 23-25 2014. ©2014 Association for Computational Linguistics

use of supervised learning (e.g., de Melo and Weikum (2012)).

Since the original version of *SimRank* (Jeh and Widom, 2002) has complexity $\mathcal{O}(n^4)$, many extensions have been proposed to speed up its calculation. A Monte Carlo algorithm, which is scalable to the whole web, was suggested by Fogaras and Rácz (2005). However, in an evaluation of this algorithm we found that it does not give competitive results (see Section 6). A matrix representation of SimRank called *SimFusion* (Xi et al., 2005) improves the computational complexity from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^3)$. Lizorkin et al. (2010) also reduce complexity to $\mathcal{O}(n^3)$ by selecting essential node pairs and using partial sums. They also give a useful overview of SimRank, SimFusion and the Monte Carlo methods of Fogaras and Rácz (2005). A non-iterative computation for SimRank was introduced by Li et al. (2010). This is especially useful for dynamic graphs. However, all of these methods have to run SimRank on the entire graph and are not efficient enough for very large graphs. We are interested in applications that only need a fraction of all $\mathcal{O}(n^2)$ pairwise similarities. The algorithm we propose below is an order of magnitude faster in such applications because it is based on a local formulation of the similarity measure.[2]

Apart from SimRank, many other similarity measures have been proposed. Leicht et al. (2006) introduce a similarity measure that is also based on the idea that nodes are similar when their neighbors are, but that is designed for bipartite graphs. However, most graphs in NLP are not bipartite and Jeh and Widom (2002) also proposed a SimRank variant for bipartite graphs.

Another important similarity measure is cosine similarity of *Personalized PageRank* (PPR) vectors. We will refer to this measure as *PPR+cos*. Hughes and Ramage (2007) find that PPR+cos has high correlation with human similarity judgments on WordNet-based graphs. Agirre et al. (2009) use PPR+cos for WordNet and for cross-lingual studies. Like CoSimRank, PPR+cos is efficient when computing single node pair similarities; we therefore use it as one of our baselines below. This method is also used by Chang et al. (2013) for semantic relatedness. They also experimented with Euclidean distance and KL-divergence. Interestingly, a simpler method performed best when comparing with human similarity judgments. In this method only the entries corresponding to the compared nodes are used for a similarity score. Rao et al. (2008) compared PPR+cos to other graph based similarity measures like shortest-path and bounded-length random walks. PPR+cos performed best except for a new similarity measure based on commute time. We do not compare against this new measure as it uses the graph Laplacian and so cannot be computed for a single node pair.

One reason CoSimRank is efficient is that we need only compute a few iterations of the random walk. This is often true of this type of algorithm; cf. (Schütze and Walsh, 2008).

LexRank (Erkan and Radev, 2004) is similar to PPR+cos in that it combines PageRank and cosine; it initializes the sentence similarity matrix of a document using cosine and then applies PageRank to compute lexical centrality. Despite this superficial relatedness, applications like lexicon extraction that look for *similar entities* and applications that look for *central entities* are quite different.

In addition to faster versions of SimRank, there has also been work on extensions of SimRank. Dorow et al. (2009) and Laws et al. (2010) extend SimRank to edge weights, edge labels and multiple graphs. We use their Multi-Edge Extraction (MEE) algorithm as one of our baselines below. A similar graph of dependency structures was built by Minkov and Cohen (2008). They applied different similarity measures, e.g., cosine of dependency vectors or a new algorithm called *path-constrained* graph walk, on synonym extraction (Minkov and Cohen, 2012). We compare CoSimRank with their results in our experiments (see Section 6).

Some other applications of SimRank or other graph based similarity measures in NLP include work on document similarity (Li et al., 2009), the transfer of sentiment information between languages (Scheible et al., 2010) and named entity disambiguation (Han and Zhao, 2010). Hoang and Kan (2010) use SimRank for related work summarization. Muthukrishnan et al. (2010) combine link based similarity and content based similarity for document clustering and classification.

These approaches use at least one of cosine similarity, PageRank and SimRank. CoSimRank can either be interpreted as an efficient version of Sim-

[2] A reviewer suggests that CoSimRank is an efficient version of SimRank in a way analogous to SALSA's (Lempel and Moran, 2000) relationship to HITS (Kleinberg, 1999) in that different aspects of similarity are decoupled.

Rank or as a version of Personalized PageRank for similarity measurement. The novelty is that we compute similarity for vectors that are induced using a new algorithm, so that the similarity measurement is much more efficient when an application only needs a fraction of all $\mathcal{O}(n^2)$ pairwise similarities.

3 CoSimRank

We first first give an intuitive introduction of CoSimRank as a Personalized PageRank (PPR) derivative. Later on, we will give a matrix formulation to compare CoSimRank with SimRank.

3.1 Personalized PageRank

Haveliwala (2002) introduced Personalized PageRank – or topic-sensitive PageRank – based on the idea that the uniform damping vector $p^{(0)}$ can be replaced by a personalized vector, which depends on node i. We usually set $p^{(0)}(i) = e_i$, with e_i being a vector of the standard basis, i.e., the i^{th} entry is 1 and all other entries are 0. The PPR vector of node i is given by:

$$p^{(k)}(i) = dAp^{(k-1)}(i) + (1-d)p^{(0)}(i) \quad (1)$$

where A is the stochastic matrix of the Markov chain, i.e., the row normalized adjacency matrix. The damping factor $d \in (0,1)$ ensures that the computation converges. The PPR vector after k iterations is given by $p^{(k)}$.

To visualize this formula, one can imagine a random surfer starting at node i and following one of the links with probability d or jumping back to the starting node i with probability $(1-d)$. Entry i of the converged PPR vector represents the probability that the random surfer is on node i after an unlimited number of steps.

To simulate the behavior of SimRank we will simplify this equation and set the damping factor $d = 1$. We will re-add a damping factor later in the calculation.

$$p^{(k)} = Ap^{(k-1)} \quad (2)$$

Note that the personalization vector $p^{(0)}$ was eliminated, but is still present as the starting vector of the iteration.

3.2 Similarity of vectors

Let $p(i)$ be the PPR vector of node i. The cosine of two vectors u and v is computed by dividing the inner product $\langle u, v\rangle$ by the lengths of the vectors. The cosine of two PPR vectors can be used as a similarity measure for the corresponding nodes (Hughes and Ramage, 2007; Agirre et al., 2009):

$$s(i,j) = \frac{\langle p(i), p(j)\rangle}{|p(i)||p(j)|} \quad (3)$$

This measure $s(i,j)$ looks at the probability that a random walker is on a certain edge after an *unlimited number of steps*. This is potentially problematic as the example in Figure 1 shows. The PPR vectors of *suit* and *dress* will have some weight on *tailor*, which is good. However, the PPR vector of *law* will also have a non-zero weight for *tailor*. So *law* and *dress* are similar because of the node *tailor*. This is undesirable.

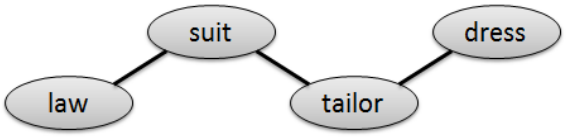

Figure 1: Graph motivating CoSimRank algorithm. Whereas PPR gives relatively high similarity to the pair (law,suit), CoSimRank assigns the pair similarity 0.

We can prevent this type of spurious similarity by *taking into account the path the random surfer took to get to a particular node*. We formalize this by defining CoSimRank $s(i,j)$ as follows:

$$s(i,j) = \sum_{k=0}^{\infty} c^k \langle p^{(k)}(i), p^{(k)}(j)\rangle \quad (4)$$

where $p^{(k)}(i)$ is the PPR vector of node i from Eq. 2 after k iterations. We compare the PPR vectors at each time step k. The sum of all similarities is the value of CoSimRank, i.e., the final similarity. We add a damping factor c, so that early meetings are more valuable than later meetings.

To compute the similarity of two vectors u and v we use the inner product $\langle\cdot,\cdot\rangle$ in Eq. 4 for two reasons:

1. This is similar to cosine similarity except that the 1-norm is used instead of the 2-norm. Since our vectors are probability vectors, we have

$$\frac{\langle p(i), p(j)\rangle}{|p(i)||p(j)|} = \langle p(i), p(j)\rangle$$

for the 1-norm.[3]

2. Without expensive normalization, we can give a simple matrix formalization of CoSimRank and compute it efficiently using fast matrix multiplication algorithms.

Later on, the following iterative computation of CoSimRank will prove useful:

$$s^{(k)}(i,j) = c^k \langle p^{(k)}(i), p^{(k)}(j) \rangle + s^{(k-1)}(i,j) \tag{5}$$

3.3 Matrix formulation

The matrix formulation of CoSimRank is:

$$\begin{aligned} S^{(0)} &= E \\ S^{(1)} &= cAA^T + S^{(0)} \\ S^{(2)} &= c^2 A^2 (A^T)^2 + S^{(1)} \\ &\dots \\ S^{(k)} &= c^k A^k (A^T)^k + S^{(k-1)} \end{aligned} \tag{6}$$

We will see in Section 5 that this formulation is the basis for a very efficient version of CoSimRank.

3.4 Convergence properties

As the PPR vectors have only positive values, we can easily see in Eq. 4 that the CoSimRank of one node pair is monotonically non-decreasing. For the dot product of two vectors, the Cauchy-Schwarz inequality gives the upper bound:

$$\langle u, v \rangle \leq \|u\| \, \|v\|$$

where $\|x\|$ is the norm of x. From Eq. 2 we get $\left\|p^{(k)}\right\|_1 = 1$, where $\|\cdot\|_1$ is the 1-norm. We also know from elementary functional analysis that the 1-norm is the biggest of all p-norms and so one has $\left\|p^{(k)}\right\| \leq 1$. It follows that CoSimRank grows more slowly than a geometric series and converges if $|c| < 1$:

$$s(i,j) \leq \sum_{k=0}^{\infty} c^k = \frac{1}{1-c}$$

If an upper bound of 1 is desired for $s(i,j)$ (instead of $1/(1-c)$), then we can use s':

$$s'(i,j) = (1-c)s(i,j)$$

[3]This type of similarity measure has also been used and investigated by Ó Séaghdha and Copestake (2008), Cha (2007), Jebara et al. (2004) (probability product kernel) and (Jaakkola et al., 1999) (Fisher kernel) among others.

4 Comparison to SimRank

The original SimRank equation can be written as follows (Jeh and Widom, 2002):

$$r(i,j) = \begin{cases} 1, & \text{if } i = j \\ \frac{c}{|N(i)||N(j)|} \sum\limits_{\substack{k \in N(i) \\ l \in N(j)}} r(k,l), & \text{else} \end{cases}$$

where $N(i)$ denotes the nodes connected to i. SimRank is computed iteratively. With A being the normalized adjacency matrix we can write SimRank in matrix formulation:

$$\begin{aligned} R^{(0)} &= E \\ R^{(k)} &= \max\{cAR^{(k-1)}A^T, R^{(0)}\} \end{aligned} \tag{7}$$

where the maximum of two matrices refers to the element-wise maximum. We will now prove by induction that the matrix formulation of CoSimRank (Eq. 6) is equivalent to:

$$S'^{(k)} = cAS'^{(k-1)}A^T + S^{(0)} \tag{8}$$

and thus very similar to SimRank (Eq. 7).

The base case $S^{(1)} = S'^{(1)}$ is trivial. Inductive step:

$$\begin{aligned} S'^{(k)} &\overset{(8)}{=} cAS'^{(k-1)}A^T + S^{(0)} \\ &= cA(c^{k-1}A^{k-1}(A^T)^{k-1} + S^{(k-2)})A^T + S^{(0)} \\ &= c^k A^k (A^T)^k + cAS^{(k-2)}A^T + S^{(0)} \\ &= c^k A^k (A^T)^k + S^{(k-1)} \overset{(6)}{=} S^{(k)} \end{aligned}$$ □

Comparing Eqs. 7 and 8, we see that SimRank and CoSimRank are very similar except that they initialize the similarities on the diagonal differently. Whereas SimRank sets each of these entries back to one at each iteration, CoSimRank adds one. Thus, when computing the two similarity measures iteratively, the diagonal element (i,i) will be set to 1 by both methods for those initial iterations for which this entry is 0 for $cAS^{(k-1)}A^T$ (i.e., before applying either max or add). The methods diverge when the entry is $\neq 0$ for the first time.

Complexity of computing all n^2 similarities. The matrix formulas of both SimRank (Eq. 7) and CoSimRank (Eq. 8) have time complexity $\mathcal{O}(n^3)$ or – if we want to take the higher efficiency of computation for sparse graphs into account – $\mathcal{O}(dn^2)$ where n is the number of nodes and d the

average degree. Space complexity is $\mathcal{O}(n^2)$ for both algorithms.

Complexity of computing $k^2 \ll n^2$ similarities. In most cases, we only want to compute k^2 similarities for k nodes. For CoSimRank, we compute the k PPR vectors in $\mathcal{O}(kdn)$ (Eq. 2) and compute the k^2 similarities in $\mathcal{O}(k^2n)$ (Eq. 5). If $d < k$, then the time complexity of CoSimRank is $\mathcal{O}(k^2n)$. If we only compute a single similarity, then the complexity is $\mathcal{O}(dn)$. In contrast, the complexity of SimRank is the same as in the all-similarities case: $\mathcal{O}(dn^2)$. It is not obvious how to design a lower-complexity version of SimRank for this case. Thus, we have reduced SimRank's cubic time complexity to a quadratic time complexity for CoSimRank or – assuming that the average degree d does not depend on n – SimRank's quadratic time complexity to linear time complexity for the case of computing few similarities.

Space complexity for computing k^2 similarities is $\mathcal{O}(kn)$ since we need only store k vectors, not the complete similarity matrix. This complexity can be exploited even for the all similarities application: If the matrix formulation cannot be used because the $\mathcal{O}(n^2)$ similarity matrix is too big for available memory, then we can compute all similarities in batches – and if desired in parallel – whose size is chosen such that the vectors of each batch still fit in memory.

In summary, CoSimRank and SimRank have similar space and time complexities for computing all n^2 similarities. For the more typical case that we only want to compute a fraction of all similarities, we have recast the global SimRank formulation as a local CoSimRank formulation. As a result, time and space complexities are much improved. In Section 6, we will show that this is also true in practice.

5 Extensions

We will show now that the basic CoSimRank algorithm can be extended in a number of ways and is thus a flexible tool for different NLP applications.

5.1 Weighted edges

The use of weighted edges was first proposed in the PageRank patent. It is straightforward and easy to implement by replacing the row normalized adjacency matrix A with an arbitrary stochastic matrix P. We can use this edge weighted PageRank for CoSimRank.

5.2 CoSimRank across graphs

We often want to compute the similarity of nodes in *two different graphs* with a known node-node correspondence; this is the scenario we are faced with in the lexicon extraction task (see Section 6). A variant of SimRank for this task was presented by Dorow et al. (2009). We will now present an equivalent method for CoSimRank. We denote the number of nodes in the two graphs U and V by $|U|$ and $|V|$, respectively. We compute PPR vectors $p \in \mathbb{R}^{|U|}$ and $q \in \mathbb{R}^{|V|}$ for each graph. Let $S^{(0)} \in \mathbb{R}^{|U| \times |V|}$ be the known node-node correspondences. The analog of CoSimRank (Eq. 4) for two graphs is then:

$$s(i,j) = \sum_{k=0}^{\infty} c^k \sum_{(u,v) \in S^{(0)}} p_u^{(k)}(i) q_v^{(k)}(j) \quad (9)$$

The matrix formulation (cf. Eq. 6) is:

$$S^{(k)} = c^k A^k S^{(0)} (B^T)^k + S^{(k-1)} \quad (10)$$

where A and B are row-normalized adjacency matrices. We can interpret $S^{(0)}$ as a change of basis. A similar approach for word embeddings was published by Mikolov et al. (2013). They call $S^{(0)}$ the translation matrix.

5.3 Typed edges

To be able to directly compare to prior work in our experiments, we also present a method to integrate a set of typed edges $\mathcal{T}$ in the CoSimRank calculation. For this we will compute a similarity matrix for each edge type τ and merge them into one matrix for the next iteration:

$$S^{(k)} = \left(\frac{c}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} A_\tau S^{(k-1)} B_\tau^T \right) + S^{(0)} \quad (11)$$

This formula is identical to the random surfer model where two surfers only meet iff they are on the same node and used the same edge type to get there. A more strict claim would be to use the same edge type at any time of their journey:

$$S^{(k)} = \frac{c^k}{|\mathcal{T}|^k} \sum_{\tau \in \mathcal{T}^k} \left(\prod_{i=1}^{k} A_{\tau_i} \right) S^{(0)} \left(\prod_{i=0}^{k-1} B_{\tau_{k-i}}^T \right) + S^{(k-1)} \quad (12)$$

We will not use Eq. 12 due to its space complexity.

5.4 Similarity of sets of nodes

CoSimRank can also be used to compute the similarity $s(V, W)$ of two sets V and W of nodes, e.g., short text snippets. We are not including this method in our experiments, but we will give the equation here, as traditional document similarity measures (e.g., cosine similarity) perform poorly on this task although there also are known alternatives with good results (Sahami and Heilman, 2006). For a set V, the initial PPR vector is given by:

$$p_i^{(0)}(V) = \begin{cases} \frac{1}{|V|}, & \text{if } i \in V \\ 0, & \text{else} \end{cases}$$

We then reuse Eq. 4 to compute $s(V, W)$:

$$s(V, W) = \sum_{k=0}^{\infty} c^k \langle p^{(k)}(V), p^{(k)}(W) \rangle$$

In summary, modifications proposed for SimRank (weighted and typed edges, similarity across graphs) as well as modifications proposed for PageRank (sets of nodes) can also be applied to CoSimRank. This makes CoSimRank a very flexible similarity measure.

We will test the first three extensions experimentally in the next section and leave similarity of node sets for future work.

6 Experiments

We evaluate CoSimRank for the tasks of synonym extraction and bilingual lexicon extraction. We use the basic version of CoSimRank (Eq. 4) for synonym extraction and the two-graph version (Eq. 9) for lexicon extraction, both with weighted edges. Our motivation for this application is that two words that are synonyms of each other should have similar lexical neighbors and that two words that are translations of each other should have neighbors that correspond to each other; thus, in each case the nodes should be similar in the graph-theoretic sense and CoSimRank should be able to identify this similarity.

We use the English and German graphs published by Laws et al. (2010), including edge weighting and normalization. Nodes are nouns, adjectives and verbs occurring in Wikipedia. There are three types of edges, corresponding to three types of syntactic configurations extracted from the parsed Wikipedias: adjective-noun, verb-object and noun-noun coordination. Table 1 gives examples and number of nodes and edges.

Edge types

relation	entities	description	example
amod	a, v	adjective-noun	a fast car
dobj	v, n	verb-object	drive a car
ncrd	n, n	noun-noun	cars and busses

Graph statistics

nodes		nouns	adjectives	verbs
	de	34,544	10,067	2,828
	en	22,258	12,878	4,866
edges		ncrd	amod	dobj
	de	65,299	417,151	143,905
	en	288,878	686,069	510,351

Table 1: Edge types (above) and number of nodes and edges (below)

6.1 Baselines

We propose CoSimRank as an efficient algorithm for computing the similarity of nodes in a graph. Consequently, we compare against the two main methods for this task in NLP: SimRank and extensions of PageRank.

We also compare against the MEE (Multi-Edge Extraction) variant of SimRank (Dorow et al., 2009), which handles labeled edges more efficiently than SimRank:

$$S'^{(k)} = \frac{c}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} A_\tau S^{(k-1)} B_\tau^T$$

$$S^{(k)} = \max\{S'^{(k)}, S^{(0)}\}$$

where A_τ is the row-normalized adjacency matrix for edge type τ (see edge types in Table 1).

Apart from SimRank, extensions of PageRank are the main methods for computing the similarity of nodes in graphs in NLP (e.g., Hughes and Ramage (2007), Agirre et al. (2009) and other papers discussed in related work). Generally, these methods compute the Personalized PageRank for each node (see Eq. 1). When the computation has converged, the similarity of two nodes is given by the cosine similarity of the Personalized PageRank vectors. We implemented this method for our experiments and call it PPR+cos.

6.2 Synonym Extraction

We use *TS68*, a test set of 68 synonym pairs published by Minkov and Cohen (2012) for evaluation. This gold standard lists a single word as the

	P@1	P@10	MRR
one-synonym			
PPR+cos	20.6%	52.9%	0.32
SimRank	**25.0**%	61.8%	**0.37**
CoSimRank	**25.0**%	61.8%	**0.37**
Typed CoSimRank	23.5%	**63.2**%	**0.37**
extended			
PPR+cos	32.6%	73.5%	0.48
SimRank	**45.6**%	**83.8**%	**0.59**
CoSimRank	**45.6**%	**83.8**%	**0.59**
Typed CoSimRank	44.1%	**83.8**%	**0.59**

Table 2: Results for synonym extraction on TS68. Best result in each column in bold.

correct synonym even if there are several equally acceptable near-synonyms (see Table 3 for examples). We call this the one-synonym evaluation. Three native English speakers were asked to mark synonyms, that were proposed by a baseline or by CoSimRank, i.e. ranked in the top 10. If all three of them agreed on one word as being a synonym in at least one meaning, we added this as a correct answer to the test set. We call this the "extended" evaluation (see Table 2).

Synonym extraction is run on the English graph. To calculate PPR+cos, we computed 20 iterations with a decay factor of 0.8 and used the cosine similarity with the 2-norm in the denominator to compare two vectors. For the other three methods, we also used a decay factor of 0.8 and computed 5 iterations. Recall that CoSimRank uses the simple inner product $\langle \cdot, \cdot \rangle$ to compare vectors.

Our evaluation measures are proportion of words correctly translated by word in the top position (P@1), proportion of words correctly translated by a word in one of the top 10 positions (P@10) and Mean Reciprocal Rank (MRR). CoSimRank's MRR scores of 0.37 (one-synonym) and 0.59 (extended) are the same or better than all baselines (see Table 2). CoSimRank and SimRank have the same P@1 and P@10 accuracy (although they differed on some decisions). CoSimRank is better than PPR+cos on both evaluations, but as this test set is very small, the results are not significant. Table 3 shows a sample of synonyms proposed by CoSimRank.

Minkov and Cohen (2012) tested cosine and random-walk measures on grammatical relationships (similar to our setup) as well as on cooccurrence statistics. The MRR scores for these methods range from 0.29 to 0.59. (MRR is equivalent to MAP as reported by Minkov and Cohen (2012) when there is only one correct answer.) Their best number (0.59) is better than our one-synonym result; however, they performed manual postprocessing of results – e.g., discarding words that are morphologically or semantically related to other words in the list – so our fully automatic results cannot be directly compared.

keyword	expected	extracted
movie	film	**film**
modern	contemporary	**contemporary**
demonstrate	protest	**show**
attractive	appealing	beautiful
economic	profitable	financial
close	shut	open

Table 3: Examples for extracted synonyms. Correct synonyms according to extended evaluation in bold.

6.3 Lexicon Extraction

We evaluate lexicon extraction on TS1000, a test set of 1000 items, (Laws et al., 2010) each consisting of an English word and its German translations. For lexicon extraction, we use the same parameters as in the synonym extraction task for all four similarity measures. We use a seed dictionary of 12,630 word pairs to establish node-node correspondences between the two graphs. We remove a search keyword from the seed dictionary before calculating similarities for it, something that the architecture of CoSimRank makes easy because we can use a different seed dictionary $S^{(0)}$ for every keyword.

Both CoSimRank methods outperform SimRank significantly (see Table 4). The difference between CoSimRank with and without typed edges is not significant. (This observation was also made for SimRank on a smaller graph and test set (Laws et al., 2010).)

PPR+cos's performance at 14.8% correct translations is much lower than SimRank and CoSimRank. The disadvantage of this similarity measure is significant and even more visible on bilingual lexicon extraction than on synonym extraction (see Table 2). The reason might be that we are not comparing the whole PPR vector anymore,

	P@1	P@10
PPR+cos	14.8%†	45.7%†
SimRank MEE	48.0%†	76.0%†
CoSimRank	61.1%	**84.0%**
Typed CoSimRank	**61.4%**	83.9%

Table 4: Results for bilingual lexicon extraction (TS1000 EN → DE). Best result in each column in bold.

	synonym extraction (68 word pairs)	lexicon extraction (1000 word pairs)
PPR+cos	2,228	**2,195**
SimRank	23,423	14,418
CoSimRank	**524**	2,342
Typed CoSimRank	615	6,108

Table 5: Execution times in minutes for CoSimRank and the baselines. Best result in each column in bold.

but only entries which occur in the seed dictionary (see Eq. 9). As the seed dictionary contains 12,630 word pairs, this means that only every fourth entry of the PPR vector (the German graph has 47,439 nodes) is used for similarity calculation. This is also true for CoSimRank, but it seems that CoSimRank is more stable because we compare more than one vector.

We also experimented with the method of Fogaras and Rácz (2005). We tried a number of different ways of modifying it for weighted graphs: (i) running the random walks with the weighted adjacency matrix as Markov matrix, (ii) storing the weight (product of each edge weight) of a random walk and using it as a factor if two walks meet and (iii) a combination of both. We needed about 10,000 random walks in all three conditions. As a result, the computational time was approximately 30 minutes per test word, so this method is even slower than SimRank for our application. The accuracies P@1 and P@10 were worse in all experiments than those of CoSimRank.

6.4 Run time performance

Table 5 compares the run time performance of CoSimRank with the baselines. We ran all experiments on a 64-bit Linux machine with 64 Intel Xenon X7560 2.27Ghz CPUs and 1TB RAM. The calculated time is the sum of the time spent in user mode and the time spent in kernel mode. The actual wall clock time was significantly lower as we used up to 64 CPUs.

Compared to SimRank, CoSimRank is more than 40 times faster on synonym extraction and six times faster on lexicon extraction. SimRank is at a disadvantage because it computes all similarities in the graph regardless of the size of the test set; it is particularly inefficient on synonym extraction because the English graph contains a large number of edges (see Table 1).

Compared to PPR+cos, CoSimRank is roughly four times faster on synonym extraction and has comparable performance on lexicon extraction. We compute 20 iterations of PPR+cos to reach convergence and then calculate a single cosine similarity. For CoSimRank, we need only compute five iterations to reach convergence, but we have to compute a vector similarity in each iteration. The counteracting effects of fewer iterations and more vector similarity computations can give either CoSimRank or PPR+cos an advantage, as is the case for synonym extraction and lexicon extraction, respectively.

CoSimRank should generally be three times faster than typed CoSimRank since the typed version has to repeat the computation for each of the three types. This effect is only visible on the larger test set (lexicon extraction) because the general computation overhead is about the same on a smaller test set.

6.5 Comparison with WINTIAN

Here we address inducing a bilingual lexicon from a seed set based on grammatical relations found by a parser. An alternative approach is to induce a bilingual lexicon from Wikipedia's interwiki links (Rapp et al., 2012). These two approaches have different strengths and weaknesses; e.g., the interwiki-link-based approach does not require a seed set, but it can only be applied to comparable corpora that consist of corresponding – although not necessarily "parallel" – documents.

Despite these differences it is still interesting to compare the two algorithms. Rapp et al. (2012) kindly provided their test set to us. It contains 1000 English words and a single correct German translation for each. We evaluate on a subset we call TS774 that consists of the 774 test word pairs that are in the intersection of words covered by the

†significantly worse than CoSimRank ($\alpha = 0.05$, one-tailed Z-Test)

	P@1	P@10
Wintian	**43.8%**	55.4%†
CoSimRank	43.0%	**73.6%**

Table 6: Results for bilingual lexicon extraction (TS774 DE → EN). Best result in each column in bold.

WINTIAN Wikipedia data (Rapp et al., 2012) and words covered by our data. Most of the 226 missing word pairs are adverbs, prepositions and plural forms that are not covered by our graphs due to the construction algorithm we use: lemmatization, restriction to adjectives, nouns and verbs etc.

Table 6 shows that CoSimRank is slightly, but not significantly worse than WINTIAN on P@1 (43.0 vs 43.8), but significantly better on P@10 (73.6 vs 55.4).[4] The reason could be that CoSimRank is a more effective algorithm than WINTIAN; but the different initializations (seed set vs interwiki links) or the different linguistic representations (grammatical relations vs bag-of-words) could also be responsible.

6.6 Error Analysis

The results on TS774 can be considered conservative since only one translation is accepted as being correct. In reality other translations might also be acceptable (e.g., both *street* and *road* for *Straße*). In contrast, TS1000 accepts more than one correct translation. Additionally, TS774 was created by translating English words into German (using Google translate). We are now testing the reverse direction. So we are doomed to fail if the original English word is a less common translation of an ambiguous German word. For example, the English word *gulf* was translated by Google to *Golf*, but the most common sense of *Golf* is the sport. Hence our algorithm will incorrectly translate it back to *golf*.

As we can see in Table 7, we also face the problems discussed by Laws et al. (2010): the algorithm sometimes picks cohyponyms (which can still be seen as reasonable) and antonyms (which are clear errors).

Contrary to our intuition, the edge-typed variant of CoSimRank did not perform significantly better than the non-edge-typed version. Looking at Table 1, we see that there is only one edge type connecting adjectives. The same is true for verbs. The random surfer only has a real choice between different edge types when she is on a noun node. Combined with the fact that only the last edge type is important this has absolutely no effect for a random surfer meeting at adjectives or verbs.

Two possible solutions would be (i) to use more fine-grained edge types, (ii) to apply Eq. 12, in which the edge type of each step is important. However, this will increase the memory needed for calculation.

[4] We achieved better results for CoSimRank by optimizing the damping factor, but in this paper, we only present results for a fixed damping factor of 0.8.

keyword	gold standard	CoSimRank
arm	poor	**impoverished**
erreichen	reach	**achieve**
gehen	go	**walk**
direkt	directly	**direct**
weit	far	further
breit	wide	narrow
reduzieren	reduce	increase
Stunde	hour	second
Westen	west	southwest
Junge	boy	child

Table 7: Examples for CoSimRank translation errors on TS774. We counted translations as incorrect if they were not listed in the gold standard even if they were correct translations according to *www.dict.cc* (in bold).

7 Summary

We have presented *CoSimRank*, a new similarity measure that can be computed for a single node pair without relying on the similarities in the whole graph. We gave two different formalizations of CoSimRank: (i) a derivation from Personalized PageRank and (ii) a matrix representation that can take advantage of fast matrix multiplication algorithms. We also presented extensions of CoSimRank for a number of applications, thus demonstrating the flexibility of CoSimRank as a similarity measure.

We showed that CoSimRank is superior to SimRank in time and space complexity; and we demonstrated that CoSimRank performs better than PPR+cos on two similarity computation tasks.

Acknowledgments. This work was supported by DFG (SCHU 2246/2-2).

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 19–27.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *WWW*, pages 107–117.

Sung-Hyuk Cha. 2007. Comprehensive survey on distance/similarity measures between probability density functions. *Mathematical Models and Methods in Applied Sciences*, 1(4):300–307.

Ching-Yun Chang, Stephen Clark, and Brian Harrington. 2013. Getting creative with semantic similarity. In *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*, pages 330–333.

Gerard de Melo and Gerhard Weikum. 2012. Uwn: A large multilingual lexical knowledge base. In *ACL (System Demonstrations)*, pages 151–156.

Beate Dorow, Florian Laws, Lukas Michelbacher, Christian Scheible, and Jason Utt. 2009. A graph-theoretic algorithm for automatic extension of translation lexicons. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, GEMS '09, pages 91–95.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Intell. Res. (JAIR)*, 22:457–479.

Dániel Fogaras and Balázs Rácz. 2005. Scaling link-based similarity search. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 641–650.

Xianpei Han and Jun Zhao. 2010. Structural semantic relatedness: a knowledge-based method to named entity disambiguation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 50–59.

Taher H. Haveliwala. 2002. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, WWW '02, pages 517–526.

Cong Duy Vu Hoang and Min-Yen Kan. 2010. Towards automated related work summarization. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 427–435.

Thad Hughes and Daniel Ramage. 2007. Lexical semantic relatedness with random graph walks. In *EMNLP-CoNLL*, pages 581–589.

Tommi Jaakkola, David Haussler, et al. 1999. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493.

Tony Jebara, Risi Kondor, and Andrew Howard. 2004. Probability product kernels. *The Journal of Machine Learning Research*, 5:819–844.

Glen Jeh and Jennifer Widom. 2002. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 538–543.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

Florian Laws, Lukas ichelbacher, Beate Dorow, Christian Scheible, Ulrich Heid, and Hinrich Schütze. 2010. A linguistically grounded graph model for bilingual lexicon extraction. In *Coling 2010: Posters*, pages 614–622.

Elizabeth Leicht, Petter Holme, and Mark Newman. 2006. Vertex similarity in networks. *Physical Review E*, 73(2):026120.

Ronny Lempel and Shlomo Moran. 2000. The stochastic approach for link-structure analysis (salsa) and the tkc effect. *Computer Networks*, 33(1):387–401.

Pei Li, Zhixu Li, Hongyan Liu, Jun He, and Xiaoyong Du. 2009. Using link-based content analysis to measure document similarity effectively. In *Proceedings of the Joint International Conferences on Advances in Data and Web Management*, APWeb/WAIM '09, pages 455–467.

Cuiping Li, Jiawei Han, Guoming He, Xin Jin, Yizhou Sun, Yintao Yu, and Tianyi Wu. 2010. Fast computation of simrank for static and dynamic information networks. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 465–476.

Dmitry Lizorkin, Pavel Velikhov, Maxim Grinev, and Denis Turdakov. 2010. Accuracy estimate and optimization techniques for simrank computation. *The VLDB Journal—The International Journal on Very Large Data Bases*, 19(1):45–66.

Rada Mihalcea and Dragomir Radev. 2011. *Graph-based natural language processing and information retrieval*. Cambridge University Press.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Einat Minkov and William W. Cohen. 2008. Learning graph walk based similarity measures for parsed text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 907–916.

Einat Minkov and William W. Cohen. 2012. Graph based similarity measures for synonym extraction from parsed text. In *Workshop Proceedings of TextGraphs-7 on Graph-based Methods for Natural Language Processing*, TextGraphs-7 '12, pages 20–24.

Pradeep Muthukrishnan, Dragomir Radev, and Qiaozhu Mei. 2010. Edge weight regularization over multiple graphs for similarity learning. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 374–383.

Diarmuid Ó Séaghdha and Ann Copestake. 2008. Semantic classification with distributional kernels. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 649–656.

Delip Rao, David Yarowsky, and Chris Callison-Burch. 2008. Affinity measures based on the graph Laplacian. In *Proceedings of the 3rd Textgraphs Workshop on Graph-Based Algorithms for Natural Language Processing*, TextGraphs-3, pages 41–48.

Reinhard Rapp, Serge Sharoff, and Bogdan Babych. 2012. Identifying word translations from comparable documents without a seed lexicon. In *LREC*, pages 460–466.

Mehran Sahami and Timothy D. Heilman. 2006. A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 377–386.

Christian Scheible, Florian Laws, Lukas Michelbacher, and Hinrich Schütze. 2010. Sentiment translation through multi-edge graphs. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1104–1112.

Hinrich Schütze and Michael Walsh. 2008. A graph-theoretic model of lexical syntactic acquisition. In *EMNLP*, pages 917–926.

Wensi Xi, Edward A. Fox, Weiguo Fan, Benyu Zhang, Zheng Chen, Jun Yan, and Dong Zhuang. 2005. Simfusion: measuring similarity using unified relationship matrix. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 130–137.